From Selective IBE to Full IBE and Selective HIBE*

Nico Döttling[1] and Sanjam Garg[2]

[1] Friedrich-Alexander-University Erlangen-Nürnberg
[2] University of California, Berkeley

Abstract. Starting with any selectively secure identity-based encryption (IBE) scheme, we give generic constructions of fully secure IBE and selectively secure hierarchical IBE (HIBE) schemes. Our HIBE scheme allows for delegation arbitrarily many times.

1 Introduction

Identity-based encryption schemes [Sha84, Coc01, BF01] (IBE) are public key encryption schemes [DH76, RSA78] for which arbitrary strings can serve as valid public keys, given short public parameters. Additionally, in such a system, given the master secret key corresponding to the public parameters, one can efficiently compute secret keys corresponding to any string id. A popular use case for this type of encryption is certificate management for encrypted email: A sender Alice can send an encrypted email to Bob at `bob@iacr.org` by just using the string "`bob@iacr.org`" and the public parameters to encrypt the message. Bob can decrypt the email using a secret-key corresponding to "`bob@iacr.org`" which he can obtain from the setup authority that holds the master secret key corresponding to the public parameters.

Two main security notions for IBE have been considered in the literature — selective security and full security. In the *selective* security experiment of identity-based encryption [CHK04], the adversary is allowed to first choose a *challenge identity* and may then obtain the public parameters and the identity secret keys for identities different from the challenge identity. The adversary's goal is to distinguish messages encrypted under the challenge identity, for which he is not allowed to obtain a secret key. On the other hand, in the *fully secure* notion [BF01], the (adversarial) choice of the challenge identity may depend arbitrarily on the public parameters. That is, the adversary may choose the challenge identity after seeing the public parameters and any number of identity secret keys of its choice. It is straightforward to see that any scheme that features full security is also selectively secure. On the other hand, example IBE schemes

* Research supported in part from AFOSR YIP Award, DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, NSF CRII Award 1464397, and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley). The views expressed are those of the author and do not reflect the official policy or position of the funding agencies.

that are selectively secure but trivially insecure in the full security sense can be constructed without significant effort.

The first IBE scheme was realized by Boneh and Franklin [BF01] based on bilinear maps. Soon after, Cocks [Coc01] provided the first IBE scheme based on quadratic residuocity assumption. However, the security of these constructions was argued in the random oracle model [BR93]. Subsequently, substantial effort was devoted to realizing IBE schemes without random oracles. The first constructions of IBE without random oracles were only proved to be selectively secure [CHK04, BB04a] and achieving full security for IBE without the random oracle heuristic required significant research effort. In particular, the first IBE scheme meeting the full security definition in the standard model were constructed by Boneh and Boyen [BB04b] and Waters [Wat05] using bilinear maps. Later, several other IBE schemes based on the learning with errors assumption [Reg05] were proposed [GPV08, AB09, CHKP10, ABB10a]. Very recently, constructions based on the security of the Diffie-Hellman Assumption and Factoring have also be obtained [DG17].

Basic IBE does not support the capability of delegating the power to issue identity secret keys. This property is captured by the notion of *hierarchical identity-based encryption* (HIBE) [HL02, GS02]. In a HIBE scheme, the owner of a master secret key can issue delegated master secret keys that enable generating identity secret keys for identities that start with a certain prefix. For instance, Alice may use a delegated master secret key to issue an identity secret key to her secretary for the identity "`alice@iacr.org` ‖ `05-24-2017`", allowing the secretary to decrypt all her emails received on this day. While HIBE trivially implies IBE, the converse question has not been resolved yet. Abdalla, Fiore and Lyubashevsky [AFL12] provided constructions of fully secure HIBE from selective-pattern-secure *wildcarded identity-based encryption* (WIBE) schemes [ACD+06] and a construction of WIBE from HIBE schemes fulfilling the stronger notion of *security under correlated randomness*. Substantial effort has been devoted to realizing HIBE schemes based on specific assumptions [GS02, BB04b, BBG05, GH09, LW10, CHKP10, ABB10b, DG17].

The question whether selectively secure IBE generically implies fully secure IBE or HIBE remains open hitherto.

1.1 Our Results

In this work, we provide a generalization of the framework developed in [DG17]. Specifically, we replace the primitive chameleon encryption (or, chameleon hash function with encryption) from [DG17] with a weaker primitive which we call *one-time signatures with encryption* (OTSE). We show that this weaker primitive[3] also suffices for realizing fully secure IBE and selectively secure HIBE building on the techniques of [DG17]. We show that OTSE can be realized from

[3] Note that chameleon hash functions imply collision resistant hash functions which one-time signatures with encryption are not known to imply [AS15, MM16].

chameleon encryption, which, as shown in [DG17], can be based on the Computational Diffie-Hellman Assumption.

In the context of [DG17], OTSE can be seen as an additional layer of abstraction that further modularizes the IBE construction of [DG17]. More concretely, when plugging the construction of OTSE from chameleon encryption (Section 4) into the construction of HIBE from OTSE (Section 7), one obtains precisely the HIBE construction of [DG17][4].

The new insight in this work is that OTSE, unlike chameleon encryption, can be realized generically from any selectively secure IBE scheme. As a consequence, it follows that both fully secure IBE and selectively secure HIBE can also be constructed *generically* from any selectively secure IBE scheme. Prior works on broadening the assumptions sufficient for IBE and HIBE have focused on first realizing selectively secure IBE. Significant subsequent research has typically been needed for realizing fully secure IBE and HIBE. Having a generic construction immediately gives improvements over previously known results and makes it easier to achieve improvements in the future. For example, using the new IBE construction of Gaborit et al. [GHPT17] we obtain a new construction of HIBE from the rank-metric problem. As another example, we obtain a construction of selectively secure HIBE from LWE with compact public parameters, i.e. a HIBE scheme where the size of the public parameters does not depend on a maximum hierarchy depth [CHKP10, ABB10b].

1.2 Technical Outline

The results in this work build on a recent work of the authors [DG17], which provides an IBE scheme in groups without pairings. In particular, we will employ the tree-based bootstrapping technique of [DG17], which itself was inspired by the tree-based construction of *Laconic Oblivious Transfer*, a primitive recently introduced by Cho et al. [CDG+17]. Below, we start by recalling [DG17] and expand on how we generalize that technique to obtain our results.

Challenge in realizing the IBE schemes. The key challenge in realizing IBE schemes is the need to "somehow compress" public keys corresponding to all possible identities (which could be exponentially many) into small public parameters. Typically, IBE schemes resolve this challenge by generating the "identity specific" public keys in a correlated manner. Since these public keys are correlated they can all be described with succinct public parameters. However, this seems hard to do when relying on an assumption such as the Diffie-Hellman Assumption. Recently, [DG17] introduced new techniques for compressing multiple uncorrelated public keys into small public parameters allowing for a construction based on the Diffie-Hellman Assumption. Below we start by describing the notion of chameleon encryption and how the IBE scheme of [DG17] uses it.

[4] The IBE construction of [DG17] is optimized and does not fit nicely into the OTSE framework.

Chameleon Encryption at a high level. At the heart of the [DG17] construction is a new chameleon hash function [KR98] with some additional encryption and decryption functionality. A (keyed) chameleon hash function $\mathsf{H_k} : \{0,1\}^n \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ on input an n-bit string x (for $n > \lambda$) and random coins $r \in \{0,1\}^\lambda$ outputs a λ-bit string. The keyed hash function is such that a trapdoor t associated to k can be used to find collisions. In particular, given a trapdoor t for k, a pair of input and random coins (x, r) and an alternative preimage x' it is easy to compute coins r' such that $\mathsf{H_k}(\mathsf{x}; r) = \mathsf{H_k}(\mathsf{x}', r')$. Additionally, we require the following encryption and decryption procedures. The encryption function $\mathsf{Enc}(\mathsf{k}, (\mathsf{h}, i, b), \mathsf{m})$ outputs a ciphertext c such that decryption $\mathsf{Dec}(\mathsf{k}, c, (\mathsf{x}, r))$ yields the original message m back as long as

$$\mathsf{h} = \mathsf{H_k}(\mathsf{x}; r) \text{ and } \mathsf{x}_i = b,$$

where (h, i, b) are the values used in the generation of the ciphertext ct. In other words, the decryptor can use the knowledge of the preimage of h as the secret key to decrypt m as long as the i^{th} bit of the preimage it can supply is equal to the value b chosen at the time of encryption. Roughly, the security requirement of chameleon encryption is that

$$\{\mathsf{k}, \mathsf{x}, r, \mathsf{Enc}(\mathsf{k}, (\mathsf{h}, i, 1 - \mathsf{x}_i), 0)\} \overset{c}{\approx} \{\mathsf{k}, \mathsf{x}, r, \mathsf{Enc}(\mathsf{k}, (\mathsf{h}, i, 1 - \mathsf{x}_i), 1)\},$$

where $\overset{c}{\approx}$ denotes computational indistinguishability. In other words, if an adversary is given a preimage x of the hash value h, but the i^{th} bit of h is different from the value b used during encryption, then ciphertext indistinguishability holds.

Realization of Chameleon Encryption. [DG17] provide the following very natural realization of the Chameleon Encryption under the DDH assumption. Given a group $\mathbb{G}$ of prime order p with a generator g, the hash function H is computed as follows:

$$\mathsf{H_k}(\mathsf{x}; r) = g^r \prod_{j \in [n]} g_{j,\mathsf{x}_j},$$

where the key $\mathsf{k} = (g, \{g_{j,0}, g_{j,1}\}_{j \in [n]})$, $r \in \mathbb{Z}_p$ and x_j is the j^{th} bit of $\mathsf{x} \in \{0,1\}^n$.

Corresponding to this chameleon hash function the encryption procedure $\mathsf{Enc}(\mathsf{k}, (\mathsf{h}, i, b), \mathsf{m})$ proceeds as follows. Sample a random value $\rho \xleftarrow{\$} \mathbb{Z}_p$ and output the ciphertext $\mathsf{ct} = (e, c, c', \{c_{j,0}, c_{j,1}\}_{j \in [n] \setminus \{i\}})$, where $c := g^\rho$, $c' := \mathsf{h}^\rho$, $\forall j \in [n] \setminus \{i\}$, $c_{j,0} := g^\rho_{j,0}$, $c_{j,1} := g^\rho_{j,1}$, and $e := \mathsf{m} \oplus g^\rho_{i,b}$. It is easy to see that if $\mathsf{x}_i = b$, then decryption $\mathsf{Dec}(\mathsf{ct}, (\mathsf{x}, r))$ can be performed by computing

$$e \oplus \frac{c'}{c^r \prod_{j \in [n] \setminus \{i\}} c_{j,\mathsf{x}_j}}.$$

However, if $\mathsf{x}_i \neq b$ then the decryptor has access to the value g^ρ_{i,x_i} but not $g^\rho_{i,b}$, and this prevents him from learning the message m. This observation can be formalized as a security proof based on the DDH assumption[5] and we refer the reader to [DG17] for the details.

[5] In fact, [DG17] show that a variant of this scheme can be proven secure under the computational Diffie-Hellman assumption.

From Chameleon Encryption to Identity-Based Encryption [DG17]. As mentioned earlier, [DG17] provide a technique for compressing uncorrelated public keys. [DG17] achieve this compression using the above-mentioned hash function in a Merkle-hash-tree fashion. In particular, the public parameters of the [DG17] IBE scheme consist of the key of the hash function and the root of the Merkle-hash-tree hashing the public keys of all the parties. Note that the number of identities is too large (specifically it is exponential) to efficiently hash all the identity-specific public keys into short public parameters. Instead [DG17] use the chameleon property of the hash function to generate the tree top-down rather than bottom-up (as is typically done in a Merkle-tree hashing). We skip the details of this top-down Merkle tree generation and refer to [DG17].

A secret key for an identity id in the [DG17] scheme consists of the hash-values along the root-to-leaf path corresponding to the leaf node id in the Merkle-hash-tree. We also include the siblings of the hash-values provided and the random coins used. Moreover, it includes the secret key corresponding to the public key at the leaf-node id.

Encryption and decryption are based on the following idea. Let $\{Y_{j,0}, Y_{j,1}\}_{j\in[n]}$ be $2n$ *labels*. Given a hash-value h, an encryptor can compute the ciphertexts $c_{j,b} := \mathsf{Enc}(\mathsf{k}, (\mathsf{h}, j, b), Y_{j,b})$ for $j = 1, \ldots, n$ and $b \in \{0, 1\}$. Given the ciphertexts $\{c_{j,0}, c_{j,1}\}_{j\in[n]}$, a decryptor in possession of a message x and coins r with $\mathsf{H}_\mathsf{k}(\mathsf{x}; r) = \mathsf{h}$ can now decrypt the ciphertexts $\{c_{j,\mathsf{x}_j}\}_{j\in[n]}$ and obtain the labels $Y_{j,\mathsf{x}_j} := \mathsf{Dec}(\mathsf{k}, (\mathsf{x}, r), c_{j,\mathsf{x}_j})$ for $j = 1, \ldots, n$. Due to the security of the chameleon encryption scheme, the decryptor will learn nothing about the labels $\{Y_{j,1-\mathsf{x}_j}\}_{j\in[n]}$.

This technique can be combined with a *projective garbling scheme* to help an encryptor provide a value $\mathsf{C}(\mathsf{x})$ to the decryptor, where C is an arbitrary circuit that knows some additional secrets chosen by the encryptor. The key point here being that the encryptor does not need to know the value x, but only a hash-value $\mathsf{h} = \mathsf{H}_\mathsf{k}(\mathsf{x}; r)$. The encyptor garbles the circuit C and obtains a garbled circuit $\tilde{\mathsf{C}}$ and labels $\{Y_{j,0}, Y_{j,1}\}$ for the input-wires of C. Encrypting the labels in the above fashion, (i.e. computing $c_{j,b} := \mathsf{Enc}(\mathsf{k}, (\mathsf{h}, j + \mathsf{id}_i \cdot \lambda, b), Y_{j,b})$), we obtain a ciphertext $\mathsf{ct} := (\tilde{\mathsf{C}}, \{c_{j,0}, c_{j,1}\}_{j\in[n]})$.

Given such a ciphertext, by the above a decryptor can obtain the labels $\{Y_{j,\mathsf{x}_j}\}_{j\in[n]}$ corresponding to the input x and evaluate the garbled circuit $\tilde{\mathsf{C}}$ to obtain $\mathsf{C}(\mathsf{x})$. By the security property of the garbling scheme and the discussion above the decryptor will learn nothing about the circuit C but the output-value $\mathsf{C}(\mathsf{x})$.

The encryption procedure of the IBE scheme provided in [DG17] uses this technique as follows. It computes a sequence of garbled circuits $\tilde{\mathsf{Q}}^{(1)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$, where the circuit $\mathsf{Q}^{(i)}$ takes as input a hash-value h, and returns chameleon encryptions $\{c_{j,0}, c_{j,1}\}_{j\in[n]}$ of the input-labels $\{Y_{j,0}^{(i+1)}, Y_{j,1}^{(i+1)}\}_{j\in[n]}$ of $\mathsf{Q}^{(i+1)}$, where $c_{j,b} := \mathsf{Enc}(\mathsf{k}, (\mathsf{h}, j + \mathsf{id}_i \cdot \lambda, b), Y_{j,b}^{(i+1)})$. The last garbled circuit $\mathsf{Q}^{(n)}$ in this sequence outputs chameleon encryptions of the labels $\{T_{j,0}, T_{j,1}\}_{j\in[n]}$ of a garbled circuit T, where the circuit T takes as input a public key pk of a standard public key encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ and outputs and encryption $\mathsf{E}(\mathsf{pk}, \mathsf{m})$

of the message m. The IBE ciphertext consists of the chameleon encryptions $\{c^{(1)}_{j,0}, c^{(1)}_{j,1}\}_{j\in[n]}$ of the input labels of the first garbled circuit $\tilde{\mathsf{Q}}^{(1)}$, the garbled circuits $\tilde{\mathsf{Q}}^{(1)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$ and the garbled circuit $\tilde{\mathsf{T}}$.

The decryptor, who is in possession of the siblings along the root-to-leaf path for identity id, can now traverse the tree as follows. He starts by decrypting $\{c^{(1)}_{j,0}, c^{(1)}_{j,1}\}_{j\in[n]}$ to the labels corresponding the first pair of siblings, evaluating the garbled circuit $\tilde{\mathsf{Q}}^{(1)}$ on this input and thus obtain chameleon encryptions $\{c^{(2)}_{j,0}, c^{(2)}_{j,1}\}_{j\in[n]}$ of the labels of the next garbled circuit $\tilde{\mathsf{Q}}^{(2)}$. Repeating this process, the decryptor will eventually be able to evaluate the last garbled circuit $\tilde{\mathsf{T}}$ and obtain $\mathsf{E}(\mathsf{pk}_{\mathsf{id}}, \mathsf{m})$, an encryption of the message m under the leaf-public-key $\mathsf{pk}_{\mathsf{id}}$. Now this ciphertext can be decrypted using the corresponding leaf-secret-key $\mathsf{sk}_{\mathsf{id}}$.

Stated differently, the encryptor uses the garbled circuits $\tilde{\mathsf{Q}}^{(1)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$ to help the decryptor traverse the tree to the leaf corresponding to the identity id and obtain an encryption of m under the leaf-public key $\mathsf{pk}_{\mathsf{id}}$ (which is not know to the encryptor).

Security of this scheme follows, as sketched above, from the security of the chameleon encryption scheme, the garbling scheme and the security of the public key encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$.

Connection to a special signature scheme. It is well-known that IBE implies a signature scheme — specifically, by interpreting the secret key for an identity id as the signature on the message id. The starting point of our work is the observation that the [DG17] IBE scheme has similarities with the construction of a signature scheme from a one-time signature scheme [Lam79, NY89]. In particular, the chameleon hash function mimics the role of a one-time signature scheme which can then be used to obtain a signature scheme similar to the IBE scheme of [DG17]. Based on this intuition we next define a new primitive which we call *one-time signature with encryption* which is very similar to (though weaker than) chameleon encryption. Construction of one-time signature with encryption from chameleon encryption is provided in Section 4.

One-Time Signatures with Encryption. A one-time signature scheme [Lam79, NY89] is a signature scheme for which security only holds if a signing key is used at most once. In more detail, a one-time signature scheme consists of three algorithms $(\mathsf{SGen}, \mathsf{SSign}, \mathsf{Verify})$, where SGen produces a pair $(\mathsf{vk}, \mathsf{sk})$ of verification and signing keys, SSign takes a signing key sk and a message x and produces a signature σ, and Verify takes a message-signature pair (x, σ) and checks if σ is a valid signature for x. One-time security means that given a verification key vk and a signature σ on a message of its own choice, an efficient adversary will not be able to concoct a valid signature σ' on a different message x'.

As with chameleon encryption, we will supplement the notion of one-time signature schemes with an additional encryption functionality. More specifically, we require additional encryption and decryption algorithms SEnc and SDec with the following properties. SEnc encrypts a message m using parameters (vk, i, b),

i.e. a verification key vk, an index i and a bit b, and any message signature pair (x, σ) satisfying "$\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \sigma) = 1$ and $\mathsf{x}_i = b$" can be used with SDec to decrypt the plaintext m. In terms of security, we require that given a signature σ on a selectively chosen message x, it is infeasible to distinguish encryptions for which the bit b is set to $1 - \mathsf{x}_i$, i.e. $\mathsf{SEnc}((\mathsf{vk}, i, 1 - \mathsf{x}_i), \mathsf{m}_0)$ and $\mathsf{SEnc}((\mathsf{vk}, i, 1 - \mathsf{x}_i), \mathsf{m}_1)$ are indistinguishable for any pair of messages $\mathsf{m}_0, \mathsf{m}_1$.

Finally, we will have the additional requirement that the verification keys are succinct, i.e. the size of the verification keys does not depend on the length of the messages that can be signed.

In the following, we will omit the requirement of a verification algorithm Verify, as such an algorithm is implied by the SEnc and SDec algorithms[6].

Moreover, we remark that in the actual definition of OTSE in Section 3, we introduce additional public parameters pp that will be used to sample verification and signing keys.

In Section 4, we will provide a direct construction of an OTSE scheme from chameleon encryption [DG17]. We remark that the techniques used in this construction appear in the HIBE-from-chameleon-encryption construction of [DG17].

We will now sketch a construction of an OTSE scheme from any selectively secure IBE scheme. Assume henceforth that $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is a selectively secure IBE scheme. We will construct an OTSE scheme $(\mathsf{SGen}, \mathsf{SSign}, \mathsf{SDec})$ as follows. SGen runs the Setup algorithm and sets $\mathsf{vk} := \mathsf{mpk}$ and $\mathsf{sk} := \mathsf{msk}$, i.e. the master public key mpk will serve as verification key vk and the master secret key msk will serve as signing key sk. To sign a message $\mathsf{x} \in \{0,1\}^n$, compute identity secret keys for the identities $\mathsf{x}_j \| \mathsf{bin}(j)$ for $j \in [n]$. Here, x_j is the j-th bit of x, $\|$ is the string concatenation operator and $\mathsf{bin}(j)$ is a $\lceil \log_2(n) \rceil$ bits representation of the index $j \in [n]$. Thus, a signature σ of x is computed by

$$\sigma = \mathsf{SSign}(\mathsf{sk}, \mathsf{x}) := \{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{x}_j \| \mathsf{bin}(j))\}_{j \in [n]}.$$

It can be checked that this is a correct and secure one-time signature scheme. The encryption and decryption algorithms SEnc and SDec are obtained from the $\mathsf{Encrypt}$ and $\mathsf{Decrypt}$ algorithms of the IBE scheme. Namely, to encrypt a plaintext m using $\mathsf{vk} = \mathsf{mpk}, i, b$, compute the ciphertext

$$c = \mathsf{SEnc}((\mathsf{vk}, i, b), \mathsf{m}) := \mathsf{Encrypt}(\mathsf{mpk}, b \| \mathsf{bin}(i), \mathsf{m}),$$

i.e. we encrypt m to the identity $b \| \mathsf{bin}(i)$. Decryption using a signature σ on a message x is performed by computing

$$\mathsf{m} = \mathsf{SDec}((\sigma, \mathsf{x}), c) := \mathsf{Decrypt}(\mathsf{sk}_{\mathsf{x}_i \| \mathsf{bin}(i)}, c),$$

which succeeds if $\mathsf{x}_i = b$. The succinctness requirement is fulfilled, as the size of the verification keys (which are master public keys) depends only (polynomially) on the security parameter, but not on the actual number of identities.

[6] To verify a signature σ for a message x using SEnc and SDec, encrypt a random plaintext m using $(\mathsf{vk}, i, \mathsf{x}_i)$ for all indices i and test whether decryption using (x, σ) yields m

Security can be based on the selective security of the IBE scheme by noting that if the i-th bit of the message x for which a signature has been issued is different from b, then the identity secret key corresponding to the identity $b\|\mathsf{bin}(i)$ is not contained in σ and we can use the selective security of the IBE scheme.

Realizing fully secure IBE. We will now show how an OTSE scheme can be bootstrapped into a fully secure IBE scheme. As mentioned before, we will use the tree based approach of the authors [DG17]. For the sake of simplicity, we will describe a stateful scheme, i.e. the key-generation algorithm keeps a state listing the identity secret keys that have been issued so far. The actual scheme, described in Section 6, will be a stateless version of this scheme, which can be obtained via pseudorandom functions.

We will now describe how identity secret keys are generated. The key generation algorithm of our scheme can be seen as an instance of the tree-based construction of a signature scheme from one-time signatures and universal one-way hash functions [NY89]. In fact, our OTSE scheme serves as one-time signature scheme with short verification keys in the construction of [NY89]. In [NY89], one-time signature scheme with short verification keys are used implicitly via a combination of one-time signatures and universal one-way hash functions.

Assume that identities are of length n and that we have a binary tree of depth n. Nodes in this tree are labelled by binary strings v that correspond to the path to this node from the root, and the root itself is labelled by the empty string $\mathsf{v}_0 = \{\}$.

We will place public keys lpk_v of a standard $\mathsf{IND}^{\mathsf{CPA}}$-secure encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ into the leaf-nodes v of the tree and a verification key vk_v of the OTSE scheme into every node v. The nodes are connected in the following manner. If v is a node with two children $\mathsf{v}\|0$ and $\mathsf{v}\|1$, we will concatenate the keys $\mathsf{vk}_{\mathsf{v}\|0}$ and $\mathsf{vk}_{\mathsf{v}\|1}$ and sign them with the signing key sk_v (corresponding to the verification key vk_v), i.e. define $\mathsf{x}_\mathsf{v} := \mathsf{vk}_{\mathsf{v}\|0}\|\mathsf{vk}_{\mathsf{v}\|1}$ and compute

$$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{sk}_\mathsf{v}, \mathsf{x}).$$

If v is a leaf-node, compute

$$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{sk}_\mathsf{v}, \mathsf{lpk}_\mathsf{v}),$$

after padding lpk_v to the appropriate length.

The master public key mpk of our scheme consist of the verification key $\mathsf{vk}_{\mathsf{v}_0}$ at the root node v_0. The identity secret key for a root-to-leaf path $\mathsf{v}_0, \ldots, \mathsf{v}_n$ consists of the root verification key $\mathsf{vk}_{\mathsf{v}_0}$, the $\mathsf{x}_{\mathsf{v}_0}, \ldots, \mathsf{x}_{\mathsf{v}_n}$ (i.e. the verification keys for the siblings along the path), the signatures $\sigma_{\mathsf{v}_0}, \ldots, \sigma_{\mathsf{v}_n}$, and the leaf public and secrets keys $\mathsf{lpk}_{\mathsf{v}_n}$ and $\mathsf{lsk}_{\mathsf{v}_n}$.

We can think of the entire information in the identity secret key as public information, *except* the leaf secret key $\mathsf{lsk}_{\mathsf{v}_n}$. That is, from a security perspective they could as well be made publicly accessible (they are not, due to the succinctness constraint of the master public key).

Encryption and Decryption. We will now describe how a plaintext is encrypted to an identity id and how it is decrypted using the corresponding identity secret key $\mathsf{sk}_{\mathsf{id}}$. The basic idea is, as in [DG17], that the encryptor delegates encryption of the plaintext m to the decryptor. More specifically, while the encryptor only knows the root verification key, the decryptor is in possession of all verification keys and signatures along the root-to-leaf path for the identity.

This delegation task will be achieved using garbled circuits along with the OTSE scheme. The goal of this delegation task is to provide a garbled circuit $\tilde{\mathsf{T}}$ with the leaf public key $\mathsf{lpk}_{\mathsf{id}}$ for the identity id. To ensure that the proper leaf public key is provided to $\tilde{\mathsf{T}}$, a sequence of garbled circuits $\tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$ is used to *traverse* the tree from the root to the leaf id.

First consider a tree that consists of one single leaf-node v, i.e. in this case there is just one leaf public key $\mathsf{lpk}_{\mathsf{v}}$ and one verification key vk_{v}. The signature σ is given by

$$\sigma := \mathsf{SSign}(\mathsf{sk}_{\mathsf{v}}, \mathsf{lpk}_{\mathsf{v}})$$

The encryptor wants to compute an encryption of a plaintext m under $\mathsf{lpk}_{\mathsf{v}}$, while only in possession of the verification key vk_{v}. It will do so using a garbled cricuit $\tilde{\mathsf{T}}$. The garbled circuit $\tilde{\mathsf{T}}$ has the plaintext m hardwired, takes as input a local public key lpk and outputs an encryption of the plaintext m under lpk, i.e. $\mathsf{E}(\mathsf{lpk}, \mathsf{m})$. Let $\{T_{j,0}, T_{j,1}\}_{j\in[\ell]}$ be the set of input labels for the garbled circuit $\tilde{\mathsf{T}}$. In this basic case, the ciphertext consists of the garbled circuit $\tilde{\mathsf{T}}$ and encryptions of the labels $\{T_{j,0}, T_{j,1}\}_{j\in[\ell]}$ under the OTSE scheme. More specifically, for all $j \in [\ell]$ and $b \in \{0,1\}$ the encryptor computes $c_{j,b} := \mathsf{SEnc}((\mathsf{vk}_{\mathsf{v}}, j, b), T_{j,b})$ and sets the ciphertext to $\mathsf{ct} := (\tilde{\mathsf{T}}, \{c_{j,b}\}_{j,b})$.

To decrypt such a ciphertext ct given $\mathsf{lsk}_{\mathsf{v}}$, $\mathsf{lpk}_{\mathsf{v}}$ and a signature σ_{v} of $\mathsf{lpk}_{\mathsf{v}}$ we proceed as follows. First, the decryptor recovers the labels $\{T_{j,(\mathsf{lpk}_{\mathsf{v}})_j}\}_j$ (where $(\mathsf{lpk}_{\mathsf{v}})_j$ is the j-th bit of $\mathsf{lpk}_{\mathsf{v}}$) by computing

$$T_{j,(\mathsf{lpk}_{\mathsf{v}})_j} := \mathsf{SDec}((\sigma, \mathsf{lpk}_{\mathsf{v}}), c_{j,(\mathsf{lpk}_{\mathsf{v}})_j}).$$

By the correctness of the OTSE scheme it follows that these are indeed the correct labels corresponding to $\mathsf{lpk}_{\mathsf{v}}$. Evaluating the garbled circuit $\tilde{\mathsf{T}}$ on these labels yields an encryption $\mathsf{f} = \mathsf{E}(\mathsf{lpk}_{\mathsf{v}}, \mathsf{m})$ of the plaintext m. Now the secret key $\mathsf{lsk}_{\mathsf{v}}$ can be used to decrypt f to the plaintext m.

For larger trees, the encryptor is not in possession of the verification key vk_{v} of the leaf-node v, and can therefore not compute the encryptions $c_{j,b} := \mathsf{SEnc}((\mathsf{vk}_{\mathsf{v}}, j, b), T_{j,b})$ by herself. This task will therefore be delegated to a sequence of garbled circuits $\tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$. For $i = 0, \ldots, n-1$, the garbled circuit $\tilde{\mathsf{Q}}^{(i)}$ has the bit id_{i+1} and the labels $\{X_{j,b}\}_{j,b}$ of the next garbled circuit $\tilde{\mathsf{Q}}^{(i+1)}$ hardwired, takes as input a verification key vk_{v} and outputs $\{c_{j,b}\}_{j,b}$, where $c_{j,b} := \mathsf{SEnc}((\mathsf{vk}_{\mathsf{v}}, \mathsf{id}_{i+1} \cdot \ell + j, b), X_{j,b})$. The garbled circuit $\tilde{\mathsf{Q}}^{(n)}$ has the labels $\{T_{j,b}\}_{j,b}$ of the garbled circuit $\tilde{\mathsf{T}}$ hardwired, takes as input a verification key vk_{v} and outputs $\{c_{j,b}\}_{j,b}$, where $c_{j,b} := \mathsf{SEnc}((\mathsf{vk}_{\mathsf{v}}, j, b), T_{j,b})$.

Thus, a decryptor who knows input labels for $\tilde{\mathsf{Q}}^{(i)}$ corresponding to vk_{v} will be able to evaluate $\tilde{\mathsf{Q}}^{(i)}$ and obtain the encrypted labels $\{c_{j,b}\}_{j,b}$, where $c_{j,b} =$

$\mathsf{SEnc}((\mathsf{vk_v}, \mathsf{id}_{i+1} \cdot \ell + j, b), X_{j,b})$. If the decryptor is in possession of the values $\mathsf{x_v} = \mathsf{vk}_{\mathsf{v}_i\|0} \| \mathsf{vk}_{\mathsf{v}_i\|1}$ and a valid signature σ_v of $\mathsf{x_v}$ that verifies with respect to $\mathsf{vk_v}$, he will be able to compute

$$X_{j,(\mathsf{vk}_{\mathsf{v}\|\mathsf{id}_i})_j} := \mathsf{SDec}((\sigma_\mathsf{v}, \mathsf{x_v}), c_{j,(\mathsf{vk}_{\mathsf{v}\|\mathsf{id}_i})_j}).$$

These are the input labels of $\tilde{\mathsf{Q}}^{(i+1)}$ corresponding to the input $\mathsf{vk}_{\mathsf{v}\|\mathsf{id}_{i+1}}$. Consequently, the decryptor will be able to evaluate $\tilde{\mathsf{Q}}^{(i+1)}$ on input $\mathsf{vk}_{\mathsf{v}\|\mathsf{id}_{i+1}}$ and so forth.

Thus, in the full scheme a ciphertext ct consists of the input-labels of the garbled circuit $\tilde{\mathsf{Q}}^{(0)}$, the sequence of garbled circuits $\tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}$ and a garbed circuit $\tilde{\mathsf{T}}$. To decrypt this ciphertext, proceed as above starting with the garbled circuit $\tilde{\mathsf{Q}}^{(0)}$ and traversing the tree to the leaf-node id, where $\tilde{\mathsf{T}}$ can be evaluated and the plaintext m be recovered as above.

In the security proof, we will replace the garbled circuits with simulated garbled circuits and change the encryptions to only encrypt labels for the next verification key in the sequence of nodes. One key idea here is that the security reduction knows all the verification keys and signatures in the tree, which as mentioned above is not private but not accessible to the real encryptor due to succinctness requirements of the public parameters. See Section 6 for details.

Hierarchical IBE. To upgrade the above scheme into a HIBE scheme, we will associate a local public key $\mathsf{lpk_v}$ with each node v of the tree, i.e. each node of the tree may serve as a *leaf* in the above scheme if needed. This means each node will contain a signature of the verification keys of the two child nodes and the local public key, i.e. we set $\mathsf{x} := \mathsf{vk}_{\mathsf{v}\|0} \| \mathsf{vk}_{\mathsf{v}\|1} \| \mathsf{lpk_v}$ and compute

$$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{sk_v}, \mathsf{x})$$

Moreover, we can make this scheme stateless using a pseudorandom function that supports the delegation of keys. In particular, the classic GGM construction [GGM86] supports delegation of PRF keys for subtrees when instantiated appropriately. We are only able to prove selective security of the obtained HIBE scheme, as in the HIBE experiment the delegation keys include PRF keys, something that was not needed to be done for the case of IBE.

2 Preliminaries

Let λ denote the security parameter. We use the notation $[n]$ to denote the set $\{1, \ldots, n\}$. By PPT we mean a probabilistic polynomial time algorithm. For any set S, we use $x \xleftarrow{\$} S$ to denote that x is sampled uniformly at random from the set S.[7] Alternatively, for any distribution D we use $x \xleftarrow{\$} D$ to denote

[7] We use this notion only when the sampling can be done by a PPT algorithm and the sampling algorithm is implicit.

that x is sampled from the distribution D. We use the operator $:=$ to represent assignment and $=$ to denote an equality check. For two strings x and x', we denote the concatenation of x and x' by $\mathsf{x}\|\mathsf{x}'$. For an integer $j \in [n]$, let $\mathsf{bin}(j)$ be the $\lceil \log_2(n) \rceil$ bits representation of j.

2.1 Public Key Encryption

Definition 1 (Public Key Encryption). *A public key encryption scheme consists of three PPT algorithms* $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *with the following syntax.*

- $\mathsf{KG}(1^\lambda)$ *takes as input a security parameter* 1^λ *and outputs a pair of public and secret keys* $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{E}(\mathsf{pk}, \mathsf{m})$ *takes as input a public key* pk *and a plaintext* m *and outputs a ciphertext* c.
- $\mathsf{D}(\mathsf{sk}, c)$ *takes as input a secret key* sk *and a ciphertext* c *and outputs a plaintext* m.

We require the following properties to hold.

- ***Completeness:*** *For every security parameter* λ *and for all messages* m*, it holds that*
$$\mathsf{D}(\mathsf{sk}, \mathsf{E}(\mathsf{pk}, \mathsf{m})) = \mathsf{m},$$
where $(\mathsf{pk}, \mathsf{sk}) := \mathsf{KG}(1^\lambda)$.
- $\mathsf{IND}^{\mathsf{CPA}}$ ***Security:*** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*
$$\Pr[\mathsf{IND}^{\mathsf{CPA}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$
where $\mathsf{IND}^{\mathsf{CPA}}(\mathcal{A})$ *is shown in Figure 1.*

Experiment $\mathsf{IND}^{\mathsf{CPA}}(\mathcal{A})$*:*

1. $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{KG}(1^\lambda)$
2. $(\mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}_1(\mathsf{pk})$.
3. $b^* \xleftarrow{\$} \{0, 1\}$.
4. $\mathsf{m}^* := \mathsf{m}_b$
5. $c^* \xleftarrow{\$} \mathsf{E}(\mathsf{pk}, \mathsf{m}^*)$
6. $b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{pk}, c^*)$
7. *Output* 1 *if* $b^* = b'$ *and* 0 *otherwise.*

Fig. 1: The $\mathsf{IND}^{\mathsf{CPA}}(\mathcal{A})$ Experiment

This notion easily extends to multiple challenge-ciphertexts. A simple hybrid argument shows that if a PPT-adversary $\mathcal{A}$ breaks the $\mathsf{IND}^{\mathsf{CPA}}$-security in the k ciphertext setting with advantage ϵ, then there exists a PPT adversary $\mathcal{A}'$ that breaks single challenge-ciphertext $\mathsf{IND}^{\mathsf{CPA}}$-security with advantage ϵ/k.

2.2 Identity-Based Encryption

Below we provide the definition of identity-based encryption (IBE).

Definition 2 (Identity-Based Encryption (IBE) [Sha84, BF01]). *An* identity-based encryption *scheme consists of four PPT algorithms* (Setup, KeyGen, Encrypt, Decrypt) *defined as follows:*

- $\mathsf{Setup}(1^\lambda)$*: given the security parameter, it outputs a master public key* mpk *and a master secret key* msk*.*
- $\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$*: given the master secret key* msk *and an identity* $\mathsf{id} \in \{0,1\}^n$*, it outputs the identity secret key* $\mathsf{sk}_{\mathsf{id}}$*.*
- $\mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}, \mathsf{m})$*: given the master public key* mpk*, an identity* $\mathsf{id} \in \{0,1\}^n$*, and a message* m*, it outputs a ciphertext* c*.*
- $\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, c)$*: given a secret key* $\mathsf{sk}_{\mathsf{id}}$ *for identity* id *and a ciphertext* c*, it outputs a plaintext* m*.*

The following completeness and security properties must be satisfied:

- ***Completeness:*** *For all security parameters* λ*, identities* $\mathsf{id} \in \{0,1\}^n$ *and messages* m*, the following holds:*

$$\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}, \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}, \mathsf{m})) = \mathsf{m}$$

  where $\mathsf{sk}_{\mathsf{id}} \leftarrow \mathsf{KeyGen}(\mathsf{msk}, \mathsf{id})$ *and* $(\mathsf{mpk}, \mathsf{msk}) \leftarrow \mathsf{Setup}(1^\lambda)$*.*
- ***Selective Security [CHK04]:*** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\Pr[\mathsf{sel\text{-}IND}^{\mathsf{IBE}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

  where $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}(\mathcal{A})$ *is shown in Figure 2, and for each key query* id *that* $\mathcal{A}$ *sends to the* KeyGen *oracle, it must hold that* $\mathsf{id} \neq \mathsf{id}^*$*.*
- ***Full Security:*** *For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\Pr[\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

  where $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$ *is shown in Figure 3, and for each key query* id *that* $\mathcal{A}$ *sends to the* KeyGen *oracle, it must hold that* $\mathsf{id} \neq \mathsf{id}^*$*.*

Experiment $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}(\mathcal{A})$*:*

1. $\mathsf{id}^* := \mathcal{A}_1(1^\lambda)$
2. $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
3. $(\mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{msk},.)}(\mathsf{mpk})$ *where* $|\mathsf{m}_0| = |\mathsf{m}_1|$ *and for each query* id *by* $\mathcal{A}_1$ *to* $\mathsf{KeyGen}(\mathsf{msk}, .)$ *we have that* $\mathsf{id} \neq \mathsf{id}^*$*.*

4. $b^* \xleftarrow{\$} \{0,1\}$.
5. $\mathsf{m}^* := \mathsf{m}_b$
6. $c^* \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{m}^*)$
7. $b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{msk},.)}(\mathsf{mpk}, c^*)$ *where for each query* id *by* $\mathcal{A}_2$ *to* $\mathsf{KeyGen}(\mathsf{msk},.)$ *we have that* $\mathsf{id} \neq \mathsf{id}^*$.
8. *Output* 1 *if* $b^* = b'$ *and* 0 *otherwise.*

Fig. 2: The $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}(\mathcal{A})$ Experiment

The selective security notion easily extends to multiple challenge ciphertexts with multiple challenge identities. A simple hybrid argument shows that if an PPT adversary $\mathcal{A}$ *break* $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$ *security in the* k *ciphertext setting with advantage* ϵ*, there there exists a PPT adversary* $\mathcal{A}'$ *that breaks single challenge ciphertext* $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$ *with advantage* ϵ/k.

Experiment $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$*:*

1. $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$.
2. $(\mathsf{id}^*, \mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{msk},.)}(\mathsf{mpk})$ *where* $|\mathsf{m}_0| = |\mathsf{m}_1|$ *and for each query* id *by* $\mathcal{A}_1$ *to* $\mathsf{KeyGen}(\mathsf{msk},.)$ *we have that* $\mathsf{id} \neq \mathsf{id}^*$.
3. $b^* \xleftarrow{\$} \{0,1\}$.
4. $\mathsf{m}^* := \mathsf{m}_b$
5. $c^* \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{m}^*)$
6. $b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{msk},.)}(\mathsf{mpk}, c^*)$ *where for each query* id *by* $\mathcal{A}_2$ *to* $\mathsf{KeyGen}(\mathsf{msk},.)$ *we have that* $\mathsf{id} \neq \mathsf{id}^*$.
7. *Output* 1 *if* $b^* = b'$ *and* 0 *otherwise.*

Fig. 3: The $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$ Experiment

2.3 Hierarchical Identity-Based Encryption (HIBE)

In a HIBE scheme, there exists an additional algorithm $\mathsf{Delegate}$ which allows to generate hierarchical secret-keys $\mathsf{msk}_{\mathsf{id}}^{\mathsf{HIBE}}$ for any input identity id. The hierarchical key for an identity id allows a user holding it to generate regular (or hierarchical keys) for any identity with prefix id. The syntax of $\mathsf{Delegate}$ is as follows.

– $\mathsf{Delegate}(\mathsf{msk}, \mathsf{id})$ takes as input a master secret key (or a delegated key) msk and an identity id and outputs a HIBE key $\mathsf{msk}_{\mathsf{id}}^{\mathsf{HIBE}}$.

In terms of correctness, we require that our HIBE additionally has the property that identity secret keys computed from delegated master secret keys are identical to identity secret keys computed by the original master secret key, i.e. for all identities id and id' it holds that

$$\mathsf{KeyGen}(\mathsf{msk}, \mathsf{id}\|\mathsf{id}') = \mathsf{KeyGen}(\mathsf{msk}_{\mathsf{id}}^{\mathsf{HIBE}}, \mathsf{id}'),$$
$$\mathsf{Delegate}(\mathsf{msk}, \mathsf{id}\|\mathsf{id}') = \mathsf{Delegate}(\mathsf{msk}_{\mathsf{id}}^{\mathsf{HIBE}}, \mathsf{id}'),$$

where $\mathsf{msk}_{\mathsf{id}}^{\mathsf{HIBE}} := \mathsf{Delegate}(\mathsf{msk}, \mathsf{id})$. This correctness condition is stronger than what is typically defined for HIBE and we use this definition as it simplifies our correctness analysis and the security definition. We note that if the distribution of the secret-key queries obtained via first computing delegation keys is different from the distribution of the secret-keys obtained directly, then a "complete" model of HIBE security is needed. This was introduced by [SW08].

The security property is analogous to the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$ except that now $\mathcal{A}$ is also allowed to ask for any hierarchical secret-key queries as long as they are not sufficient for decrypting the challenge ciphertext. We only consider the notion of *selective security* for HIBE; namely, the adversary $\mathcal{A}$ is required to announce the challenge identity id^* before it can make any secret-key or hierarchical secret-key queries.

Selective Security: For any PPT adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that the following holds:

$$\Pr[\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}(\mathcal{A})$ is shown in Figure 4. For each identity key query id that $\mathcal{A}$ sends to the KeyGen oracle, it must hold that $\mathsf{id} \neq \mathsf{id}^*$. Moreover, for each HIBE key query id that $\mathcal{A}$ sends to the $\mathsf{Delegate}$ oracle, it must hold that id is not a prefix of id^*.

Experiment $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}(\mathcal{A})$:

1. $\mathsf{id}^* := \mathcal{A}_1(1^\lambda)$
2. $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda)$
3. $(\mathsf{m}_0, \mathsf{m}_1) \xleftarrow{\$} \mathcal{A}_1^{\mathsf{KeyGen}(\mathsf{msk},.),\mathsf{Delegate}(\mathsf{msk},\cdot)}(\mathsf{mpk})$ where $|\mathsf{m}_0| = |\mathsf{m}_1|$ and for each query id by $\mathcal{A}_1$ to $\mathsf{KeyGen}(\mathsf{msk}, .)$ we have that $\mathsf{id} \neq \mathsf{id}^*$ and for each $\mathsf{Delegate}(\mathsf{msk}, .)$ query we have that id is not a prefix of id^*.
4. $b^* \xleftarrow{\$} \{0, 1\}$.
5. $\mathsf{m}^* := \mathsf{m}_b$
6. $c^* \xleftarrow{\$} \mathsf{Encrypt}(\mathsf{mpk}, \mathsf{id}^*, \mathsf{m}^*)$
7. $b' \xleftarrow{\$} \mathcal{A}_2^{\mathsf{KeyGen}(\mathsf{msk},.),\mathsf{Delegate}(\mathsf{msk},\cdot)}(\mathsf{mpk}, c^*)$ where for each query id by $\mathcal{A}_2$ to $\mathsf{KeyGen}(\mathsf{msk}, .)$ we have that $\mathsf{id} \neq \mathsf{id}^*$ and for each $\mathsf{Delegate}(\mathsf{msk}, .)$ query we have that id is not a prefix of id^*.
8. Output 1 if $b^* = b'$ and 0 otherwise.

Fig. 4: The $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}(\mathcal{A})$ Experiment

2.4 Chameleon Encryption

Definition 3 (Chameleon Encryption [DG17]). *A chameleon encryption scheme consists of five PPT algorithms* CGen, CHash, CHash^{-1}, CEnc, *and* CDec *with the following syntax.*

- $\mathsf{CGen}(1^\lambda, n)$*: Takes the security parameter* λ *and a message-length* n *(with* $n = \mathsf{poly}(\lambda)$*) as input and outputs a key* k *and a trapdoor* t.
- $\mathsf{CHash}(\mathsf{k}, \mathsf{x}; r)$*: Takes a key* k*, a message* $\mathsf{x} \in \{0,1\}^n$*, and coins* r *and outputs a hash value* h*, where the size of* h *is* λ *bits.*
- $\mathsf{CHash}^{-1}(\mathsf{t}, (\mathsf{x}, r), \mathsf{x}')$*: Takes a trapdoor* t*, previously used message* $\mathsf{x} \in \{0,1\}^n$ *and coins* r*, and a message* $\mathsf{x}' \in \{0,1\}^n$ *as input and returns* r'.
- $\mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, i, b), \mathsf{m})$*: Takes a key* k*, a hash value* h*, an index* $i \in [n]$*,* $b \in \{0,1\}$*, and a message* $\mathsf{m} \in \{0,1\}^*$ *as input and outputs a ciphertext* ct.[8]
- $\mathsf{CDec}(\mathsf{k}, (\mathsf{x}, r), \mathsf{ct})$*: Takes a key* k*, a message* x*, coins* r *and a ciphertext* ct*, as input and outputs a value* m *(or* $\perp$*).*

We require the following properties

- ***Uniformity****: For* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$ *the two distributions* $\mathsf{CHash}(\mathsf{k}, \mathsf{x}; r)$ *and* $\mathsf{CHash}(\mathsf{k}, \mathsf{x}'; r')$ *are statistically close (when* r, r' *are chosen uniformly at random).*
- ***Trapdoor Collisions****: For every choice of* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^n$ *and* r *it holds that if* $(\mathsf{k}, \mathsf{t}) \xleftarrow{\$} \mathsf{CGen}(1^\lambda, n)$ *and* $r' := \mathsf{CHash}^{-1}(\mathsf{t}, (\mathsf{x}, r), \mathsf{x}')$*, then it holds that*

$$\mathsf{CHash}(\mathsf{k}, \mathsf{x}; r) = \mathsf{CHash}(\mathsf{k}, \mathsf{x}'; r'),$$

  i.e. $\mathsf{CHash}(\mathsf{k}, \mathsf{x}; r)$ *and* $\mathsf{CHash}(\mathsf{k}, \mathsf{x}'; r')$ *generate the same hash* h*. Moreover, if* r *is chosen uniformly at random, then* r' *is also statistically close to uniform.*
- ***Correctness****: For any choice of* $\mathsf{x} \in \{0,1\}^n$*, coins* r*, index* $i \in [n]$ *and message* m *it holds that if* $(\mathsf{k}, \mathsf{t}) \xleftarrow{\$} \mathsf{CGen}(1^\lambda, n)$*,* $\mathsf{h} := \mathsf{CHash}(\mathsf{k}, \mathsf{x}; r)$*, and* $\mathsf{ct} \xleftarrow{\$} \mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, i, \mathsf{x}_i), \mathsf{m})$ *then* $\mathsf{CDec}(\mathsf{k}, (\mathsf{x}, r), \mathsf{ct}) = \mathsf{m}$.
- ***Security****: For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\Pr[IND^{CE}_{\mathcal{A}}(1^\lambda) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

  where $IND^{CE}_{\mathcal{A}}$ *is shown in Figure 5.*

[8] ct is assumed to contain (h, i, b).

Experiment $IND^{CE}_{\mathcal{A}=(\mathcal{A}_1,\mathcal{A}_2)}(1^\lambda)$*:*

1. $(\mathsf{k}, \mathsf{t}) \xleftarrow{\$} \mathsf{CGen}(1^\lambda, n)$.
2. $(\mathsf{x}, r, i \in [n]) \xleftarrow{\$} \mathcal{A}_1(\mathsf{k})$.
3. $b^* \xleftarrow{\$} \{0,1\}$.
4. $\mathsf{ct} \xleftarrow{\$} \mathsf{CEnc}(\mathsf{k}, (\mathsf{CHash}(\mathsf{k}, \mathsf{x}; r), i, 1 - \mathsf{x}_i), b)$.
5. $b' \xleftarrow{\$} \mathcal{A}_2(\mathsf{k}, \mathsf{ct}, (\mathsf{x}, r))$.
6. *Output* 1 *if* $b^* = b'$ *and* 0 *otherwise.*

Fig. 5: The $\mathrm{IND}^{\mathrm{CE}}_{\mathcal{A}}$ Experiment

2.5 Garbled Circuits

Garbled circuits were first introduced by Yao [Yao82] (see Lindell and Pinkas [LP09] and Bellare et al. [BHR12] for a detailed proof and further discussion). A projective circuit garbling scheme is a tuple of PPT algorithms $(\mathsf{Garble}, \mathsf{Eval})$ with the following syntax.

- $\mathsf{Garble}(1^\lambda, \mathsf{C})$ takes as input a security parameter λ and a circuit C and outputs a *garbled circuit* $\tilde{\mathsf{C}}$ and labels $\mathsf{e}_C = \{X_{\iota,0}, X_{\iota,1}\}_{\iota\in[n]}$, where n is the number of input wires of C.
- Projective Encoding: To encode an $\mathsf{x} \in \{0,1\}^n$ with the input labels $\mathsf{e}_C = \{X_{\iota,0}, X_{\iota,1}\}_{\iota\in[n]}$, we compute $\tilde{\mathsf{x}} := \{X_{\iota,\mathsf{x}_\iota}\}_{\iota\in[n]}$.
- $\mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{\mathsf{x}})$: takes as input a garbled circuit $\tilde{\mathsf{C}}$ and a garbled input $\tilde{\mathsf{x}}$, represented as a sequence of input labels $\{X_{\iota,\mathsf{x}_\iota}\}_{\iota\in[n]}$, and outputs an output y.

We will denote hardwiring of an input s into a circuit C by $\mathsf{C}[s]$. The garbling algorithm Garble treats the hardwired input as a regular input and additionally outputs the garbled input corresponding to s (instead of all the labels of the input wires corresponding to s). If a circuit C uses additional randomness, we will implicitly assume that appropriate random coins are hardwired in this circuit during garbling.

Correctness. For correctness, we require that for any circuit C and input $\mathsf{x} \in \{0,1\}^n$ we have that

$$\Pr\left[\mathsf{C}(\mathsf{x}) = \mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{\mathsf{x}})\right] = 1$$

where $(\tilde{\mathsf{C}}, \mathsf{e}_C = \{X_{\iota,0}, X_{\iota,1}\}_{\iota\in[n]}) \xleftarrow{\$} \mathsf{Garble}(1^\lambda, \mathsf{C})$ and $\tilde{\mathsf{x}} := \{X_{\iota,\mathsf{x}_\iota}\}$.

Security. For security, we require that there is a PPT simulator GCSim such that for any circuit C and any input x, we have that

$$(\tilde{\mathsf{C}}, \tilde{\mathsf{x}}) \overset{\text{comp}}{\approx} \mathsf{GCSim}(\mathsf{C}, \mathsf{C}(\mathsf{x}))$$

where $(\tilde{\mathsf{C}}, \mathsf{e}_C = \{X_{\iota,0}, X_{\iota,1}\}_{\iota\in[n]}) := \mathsf{Garble}(1^\lambda, \mathsf{C})$ and $\tilde{\mathsf{x}} := \{X_{\iota,\mathsf{x}_\iota}\}$.

2.6 Delegatable Pseudorandom Functions

In our HIBE construction we will need a PRF for which the inputs can be binary strings of unrestricted length and which supports the delegation of seeds for inputs that start with certain prefixes.

Definition 4. *A delegatable pseudorandom function consists of two algorithms F and $F.\mathsf{Delegate}$ with the following syntax.*

- $F(s, \mathsf{x})$ *takes as input a seed* $s \in \{0,1\}^\lambda$ *and a string* $\mathsf{x} \in \{0,1\}^*$ *and outputs a value* $u \in \{0,1\}^\lambda$.
- $F.\mathsf{Delegate}(s, \mathsf{x})$ *takes as input a seed* s *and an input* x *and outputs a seed* s_x.

We require the following properties of a delegatable pseudorandom function.

- ***Delegatability****: It holds for all inputs* $\mathsf{x}, \mathsf{x}' \in \{0,1\}^*$ *that*

$$F(s, \mathsf{x} \| \mathsf{x}') = F(s_\mathsf{x}, \mathsf{x}'),$$

  where $s_\mathsf{x} := F.\mathsf{Delegate}(s, \mathsf{x})$.
- ***Pseudorandomness****: It holds for all PPT distinguishers* $\mathcal{D}$ *and every* $\mathsf{x} \in \{0,1\}^*$ *of size at most polynomial in* λ *that*

$$|\Pr[\mathcal{D}^{F(s,\cdot),\mathsf{Delegate}(s,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{D}^{H(\cdot),\mathsf{Delegate}(s,\cdot)}(1^\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

  where $s \xleftarrow{\$} \{0,1\}^\lambda$ *is chosen uniformly at random,* H *is a function which is uniformly random on all prefixes of* x *(including* x*) and identical to* $F(s, \cdot)$ *on all other inputs, and* $\mathsf{Delegate}(s, \cdot)$ *delegates seeds for all inputs* $\mathsf{x}' \in \{0,1\}^*$ *that are not a prefix of* x.

We will briefly sketch a simple variant of the GGM construction [GGM84] which satisfies the above definition. Let $\mathsf{G} : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be a length-tripling pseudorandom generator and $\mathsf{G}_0, \mathsf{G}_1$ and G_2 be the $1 \dots \lambda$, $\lambda + 1 \dots 2\lambda$ and $2\lambda + 1 \dots 3\lambda$ bits of the output of G, respectively. Now define a GGM-type pseudo-random function $F : \{0,1\}^\lambda \times \{0,1\}^* \to \{0,1\}^\lambda$ such that $F(s, \mathsf{x}) := \mathsf{G}_2(\mathsf{G}_{\mathsf{x}_n}(\mathsf{G}_{\mathsf{x}_{n-1}}(\dots(\mathsf{G}_{\mathsf{x}_1}(s))\dots)))$, where for each $i \in [n]$ x_i is the i^{th} bit of $\mathsf{x} \in \{0,1\}^n$. $F.\mathsf{Delegate}(s, \mathsf{x})$ computes and outputs $\mathsf{G}_{\mathsf{x}_n}(\mathsf{G}_{\mathsf{x}_{n-1}}(\dots(\mathsf{G}_{\mathsf{x}_1}(s))\dots))$.

3 One-Time Signatures with Encryption

In this Section, we will introduce a primitive we call *One-Time Signatures with Encryption* (OTSE). Syntactically, we will not require the existence of a verification algorithm for such signature schemes, but instead require the existence of accompanying encryption and decryption algorithms. Details follow.

Definition 5. *A One-Time Signature with Encryption (OTSE) scheme consists of five algorithms* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *with the following syntax.*

- $\mathsf{SSetup}(1^\lambda, \ell)$*: Takes as input an unary encoding of the security parameter* 1^λ *and a message length parameter* ℓ *and outputs public parameters* pp*.*
- $\mathsf{SGen}(\mathsf{pp})$*: Takes as input public parameters* pp *and outputs a pair* $(\mathsf{vk}, \mathsf{sk})$ *of verification and signing keys.*
- $\mathsf{SSign}(\mathsf{pp}, \mathsf{sk}, \mathsf{x})$*: Takes as input public parameters* pp*, a signing key* sk *and a message* x *and outputs a signature* σ*.*
- $\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i, b), \mathsf{m})$*: Takes as input public parameters* pp*, a verification key* vk*, an index* i*, a bit* b *and a plaintext* m *and outputs a ciphertext* c*. We will generally assume that the index* i *and the bit* b *are included in* c*.*
- $\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}, \sigma, \mathsf{x}), c)$*: Takes as input public parameters* pp*, a verification key* vk*, a signature* σ*, a message* x *and a ciphertext* c *and returns a plaintext* m*.*

We require the following properties.

- ***Succinctness****: For* $\mathsf{pp} := \mathsf{SSetup}(1^\lambda, \ell)$ *and* $(\mathsf{vk}, \mathsf{sk}) := \mathsf{SGen}(\mathsf{pp}, \ell)$ *it holds that the size of* vk *is independent of* ℓ*, only depending on* λ*.*
- ***Correctness****: It holds for all security parameters* λ*, every message* x *and every plaintext* m *that if* $\mathsf{pp} := \mathsf{Setup}(1^\lambda, \ell)$*,* $(\mathsf{vk}, \mathsf{sk}) := \mathsf{SGen}(\mathsf{pp})$ *and* $\sigma := \mathsf{SSign}(\mathsf{sk}, \mathsf{x})$ *then*

$$\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}, \sigma, \mathsf{x}), \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i.b), \mathsf{m})) = \mathsf{m}.$$

- ***Selective Security****: For any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$*, there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that the following holds:*

$$\Pr[\mathsf{IND}^{\mathsf{OTSE}}(\mathcal{A}) = 1] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

where $\mathsf{IND}^{\mathsf{OTSE}}(\mathcal{A})$ *is shown in Figure 6.*

Experiment $\mathsf{IND}^{\mathsf{OTSE}}(\mathcal{A})$*:*

1. $\mathsf{pp} := \mathsf{SSetup}(1^\lambda, \ell)$
2. $\mathsf{x} := \mathcal{A}_1(\mathsf{pp})$
3. $(\mathsf{vk}, \mathsf{sk}) := \mathsf{SGen}(\mathsf{pp})$
4. $\sigma := \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}, \mathsf{x})$
5. $(i, \mathsf{m}_0, \mathsf{m}_1) := \mathcal{A}_2(\mathsf{pp}, \mathsf{vk}, \sigma)$
6. $b^* \xleftarrow{\$} \{0, 1\}$
7. $\mathsf{m}^* := \mathsf{m}_{b^*}$
8. $c^* := \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, i, 1 - \mathsf{x}_i), \mathsf{m}^*)$
9. $b' := \mathcal{A}_3(\mathsf{pp}, \mathsf{vk}, \sigma, c^*)$
10. *Output* 1 *if* $b' = b^*$ *and* 0 *otherwise.*

Fig. 6: The $\mathsf{IND}^{\mathsf{OTSE}}(\mathcal{A})$ Experiment

Again, we remark that multi-challenge security follows via a hybrid argument.

4 One-Time Signatures with Encryption from Chameleon Encryption

In this Section we provide a construction of an OTSE scheme from chameleon encryption.

$\mathsf{SSetup}(1^\lambda, \ell)$: Compute $(K, \cdot) := \mathsf{CGen}(1^\lambda, \ell)$ and output $\mathsf{pp} := K$.

$\mathsf{SGen}(\mathsf{pp})$: Compute $(\mathsf{k}, \mathsf{t}) := \mathsf{CGen}(1^\lambda, \lambda)$, sample $r' \xleftarrow{\$} \{0,1\}^\lambda$, compute $\mathsf{h} := \mathsf{CHash}(\mathsf{k}, 0^\lambda; r')$. Set $\mathsf{vk} := (\mathsf{k}, h)$, $\mathsf{sk} := (\mathsf{t}, r')$ and output $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{SSign}(\mathsf{pp}, \mathsf{sk} = (\mathsf{t}, r'), \mathsf{x})$: Compute $\mathsf{y} := \mathsf{CHash}(K, \mathsf{x})$ and $r{:=}\mathsf{CHash}^{-1}(\mathsf{t},(0^\lambda,r'),\mathsf{y})$, output $\sigma := r$.

$\mathsf{SEnc}(\mathsf{pp} = K, (\mathsf{vk} = (\mathsf{k}, \mathsf{h}), i, b), \mathsf{m})$: Let C be the following circuit.

- $\mathsf{C}[K, i, b](\mathsf{y})$: Compute and output $\mathsf{CEnc}(K, (\mathsf{y}, i, b), \mathsf{m})$.

$(\tilde{\mathsf{C}}, \mathsf{e}_C) := \mathsf{Garble}(1^\lambda, \mathsf{C}[K, i, b])$
Parse $\mathsf{e}_C = \{Y_{\iota,0}, Y_{\iota,1}\}_{\iota \in [\lambda]}$
$\mathsf{f}_C := \{\mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, \iota, b'), Y_{\iota,b'})\}_{\iota \in [\lambda], b' \in \{0,1\}}$
Output $\mathsf{ct} := (\tilde{\mathsf{C}}, \mathsf{f}_C)$.

$\mathsf{SDec}(\mathsf{pp} = K, (\mathsf{vk} = (\mathsf{k}, \mathsf{h}), \sigma = r, \mathsf{x}), \mathsf{ct} = (\tilde{\mathsf{C}}, \mathsf{f}_C))$:

Parse $\mathsf{f}_C = \{c_{\iota,b'}\}_{\iota \in [\lambda], b' \in \{0,1\}}$
$\mathsf{y} := \mathsf{CHash}(K, \mathsf{x})$
$\tilde{\mathsf{y}} := \{\mathsf{CDec}(\mathsf{k}, (\mathsf{y}, r), c_{\iota,\mathsf{y}_\iota})\}_{\iota \in [\lambda]}$
$c' := \mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{\mathsf{y}})$
$\mathsf{m} := \mathsf{CDec}(K, \mathsf{x}, c')$
Output m

Succinctness and Correctness By construction the size of $\mathsf{vk} = (\mathsf{k}, \mathsf{h})$ depends only on λ, so we have established the succinctness property. To see that the construction is correct, note that since the hash value $\mathsf{h} = \mathsf{CHash}(\mathsf{k}, \mathsf{y}; r)$ and $c_{\iota,b'} = \mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, \iota, b'), Y_{\iota,b'})$, it holds by the correctness property of the chameleon encryption scheme $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$ that

$$\tilde{\mathsf{y}} = \{\mathsf{CDec}(\mathsf{k}, (\mathsf{y}, r), c_{\iota,\mathsf{y}_\iota})\}_{\iota \in [\lambda]} = \{Y_{\iota,\mathsf{y}_\iota}\}.$$

Therefore, as $(\tilde{\mathsf{C}}, \mathsf{e}_C) = \mathsf{Garble}(1^\lambda, \mathsf{C}[K, i, b])$, it holds by the correctness of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$ that

$$c' = \mathsf{Eval}(\tilde{\mathsf{C}}, \tilde{\mathsf{y}}) = \mathsf{C}[K, i, b](\mathsf{y}) = \mathsf{CEnc}(K, (\mathsf{y}, i, b), \mathsf{m}).$$

Finally, as $\mathsf{y} = \mathsf{CHash}(K, \mathsf{x})$, it holds by the correctness of the of the chameleon encryption scheme $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$ that

$$\mathsf{CDec}(K, \mathsf{x}, c') = \mathsf{m}.$$

Security We will now establish the $\mathsf{IND}^{\mathsf{OTSE}}$ security of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ from the $\mathsf{IND}^{\mathsf{CE}}$-security of $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$ and the security of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.

Theorem 1. *Assume that* $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$ *is* $\mathsf{IND}^{\mathsf{CE}}$*-secure and* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme. Then* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure.*

Proof. Let $\mathcal{A}$ be a PPT-adversary against $\mathsf{IND}^{\mathsf{OTSE}}$. Consider the following hybrid experiments.

Hybrid $\mathcal{H}_0$ This experiment is identical to $\mathsf{IND}^{\mathsf{OTSE}}(\mathcal{A})$.

Hybrid $\mathcal{H}_1$ This experiment is identical to $\mathcal{H}_0$, except that f_C is computed by $\mathsf{f}_C := \{\mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, \iota, b'), Y_{\iota,\mathsf{y}_\iota})\}_{\iota\in[\lambda],b'\in\{0,1\}}$ instead of by the expression $\mathsf{f}_C := \{\mathsf{CEnc}(\mathsf{k}, (\mathsf{h}, \iota, b'), Y_{\iota,b'})\}_{\iota\in[\lambda],b'\in\{0,1\}}$. Computational indistinguishability between hybrids $\mathcal{H}_0$ and $\mathcal{H}_1$ follows by the $\mathsf{IND}^{\mathsf{CE}}$-security of $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$. Note that the security reduction has no access to the collision-trapdoor t. However, as the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment is defined selectively, the reduction gets to see x before it has to provide vk. Consequently, it can set $\mathsf{h} := \mathsf{CHash}(\mathsf{k}, \mathsf{CHash}(K, \mathsf{x}); r)$, $\mathsf{vk} := (\mathsf{k}, \mathsf{h})$ and present $\sigma := r$ as a valid signature to the adversary without the need of a collision trapdoor.

Hybrid $\mathcal{H}_2$ This experiment is identical to $\mathcal{H}_1$, except that we compute $\tilde{\mathsf{C}}$ and $\tilde{y}$ by $(\tilde{\mathsf{C}}, \tilde{y}) := \mathsf{GCSim}(\mathsf{C}, c)$, where $c := \mathsf{CEnc}(K, (\mathsf{y}, i, b), \mathsf{m})$ instead of $(\tilde{\mathsf{C}}, \mathsf{e}_C) := \mathsf{Garble}(1^\lambda, \mathsf{C}[K, i, b])$, where $\mathsf{e}_C = \{Y_{\iota,b'}\}_{\iota\in[\lambda],b'\in\{0,1\}}$ and $\tilde{\mathsf{y}} = \{Y_{\iota,\mathsf{y}_\iota}\}$. Computational indistinguishability between hybrids $\mathcal{H}_1$ and $\mathcal{H}_2$ follows by the security of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$. By the $\mathsf{IND}^{\mathsf{CE}}$-security of $(\mathsf{CGen}, \mathsf{CHash}, \mathsf{CHash}^{-1}, \mathsf{CEnc}, \mathsf{CDec})$ it follows that the advantage of $\mathcal{A}$ in $\mathcal{H}_2$ is negligible.

5 One-Time Signatures with Encryption from Selectively Secure IBE

We will now provide a construction of an OTSE scheme from selectively secure IBE. Let therefore $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be an IBE scheme.

$\mathsf{SSetup}(1^\lambda, \ell)$: Output $\mathsf{pp} := \ell$.

$\mathsf{SGen}(\mathsf{pp})$: Compute $(\mathsf{mpk}, \mathsf{msk}) := \mathsf{Setup}(1^\lambda)$, set $\mathsf{vk} := \mathsf{mpk}$ and $\mathsf{sk} := \mathsf{msk}$ and output $(\mathsf{vk}, \mathsf{sk})$.

$\mathsf{SSign}(\mathsf{pp}, \mathsf{sk} = \mathsf{msk}, \mathsf{x})$: Compute and output $\sigma := \{\mathsf{KeyGen}(\mathsf{msk}, \mathsf{x}_\iota \| \mathsf{bin}(\iota))\}_{\iota\in[\ell]}$.

$\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk} = \mathsf{mpk}, i, b), \mathsf{m})$: Compute and output $c := \mathsf{Encrypt}(\mathsf{mpk}, b \| \mathsf{bin}(i), \mathsf{m})$.

$\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}, \sigma, \mathsf{x}), c)$: Parse $\sigma = \{\mathsf{sk}_{\mathsf{x}_\iota \| \mathsf{bin}(\iota)}\}_{\iota\in[\ell]}$. Compute and output $\mathsf{m} := \mathsf{Decrypt}(\mathsf{sk}_{\mathsf{x}_i \| \mathsf{bin}(i)}, c)$.

Succinctness and Correctness The succinctness property follows directly form the fact the size of the master public key mpk does not depend on this length of the identities, but is a fixed polynomial in the security parameter λ.

On the other hand, correctness follows from the correctness of the IBE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

Security We will now show that the $\mathsf{IND}^{\mathsf{OTSE}}$-security of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ follows from the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$-security of the IBE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

Theorem 2. *Assume that* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$ *secure. Then* $(\cdot, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure.*

Proof. Let $\mathcal{A}$ be a PPT adversary that breaks the $\mathsf{IND}^{\mathsf{OTSE}}$-security of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ with advantage ϵ. We will provide a reduction $\mathcal{R}$ such that $\mathcal{R}^{\mathcal{A}}$ breaks the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$-security of $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ with advantage ϵ. $\mathcal{R}$ proceeds as follows. $\mathcal{R}$ first guesses a random index $i^* \xleftarrow{\$} [\ell]$. It then simulates the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment with $\mathcal{A}$ until $\mathcal{A}$ outputs a message x (that is, $\mathcal{R}$ runs $\mathcal{A}_1(1^\lambda)$). $\mathcal{R}$ now declares its challenge identity $\mathsf{id}^* := (1 - \mathsf{x}_{i^*}) \| \mathsf{bin}(i^*)$ to the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$ experiment and also asks for identity secret keys corresponding to the identities $\{\mathsf{x}_\iota \| \mathsf{bin}(\iota)\}_{\iota \in [\ell]}$. $\mathcal{R}$ now receives the master public key mpk and the identity secret keys $\{\mathsf{sk}_{\mathsf{x}_\iota \| \mathsf{bin}(\iota)}\}_{\iota \in [\ell]}$. Next, $\mathcal{R}$ sets $\mathsf{vk} := \mathsf{mpk}$ and $\sigma := \{\mathsf{sk}_{\mathsf{x}_\iota \| \mathsf{bin}(\iota)}\}_{\iota \in [\ell]}$ and provides vk and σ to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation until $\mathcal{A}$ outputs a triple $(i, \mathsf{m}_0, \mathsf{m}_1)$. If $i \neq i^*$, $\mathcal{R}$ aborts the simulation and outputs a random bit. Otherwise, $\mathcal{R}$ sends $(\mathsf{m}_0, \mathsf{m}_1)$ to the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$-experiment, obtains a challenge ciphertext c^* and forwards c^* to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation and outputs whatever $\mathcal{A}$ outputs.

We will now analyze the advantage of $\mathcal{R}^{\mathcal{A}}$. Clearly, if $i^* \neq i$, then the advantage of $\mathcal{R}^{\mathcal{A}}$ is 0. On the other hand, if $i^* = i$, then from the view of $\mathcal{A}$ the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment is simulated perfectly, where the challenge bit of the simulated $\mathsf{IND}^{\mathsf{OTSE}}$-experiment is identical to the challenge bit b^* of the $\mathsf{sel\text{-}IND}^{\mathsf{IBE}}$-experiment. Consequently, in this case the advantage of $\mathcal{R}^{\mathcal{A}}$ is identical to the advantage of $\mathcal{A}$. Since i^* is chosen uniformly at random, it holds $i^* = i$ with probability $1/\ell$. We can conclude that the advantage of $\mathcal{R}^{\mathcal{A}}$ is

$$\mathsf{Adv}_{\mathsf{sel\text{-}IND}^{\mathsf{IBE}}}(\mathcal{R}^{\mathcal{A}}) = \frac{1}{\ell} \cdot \mathsf{Adv}_{\mathsf{IND}^{\mathsf{OTSE}}}(\mathcal{A}) = \frac{\epsilon}{\ell},$$

which concludes the proof.

6 Achieving Fully Secure IBE

Let in the following $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ be an OTSE scheme. Without loss of generality, we will assume that the signing algorithm SSign is deterministic. This can always be achieved by making an additional pseudorandom function seed part of the signing key and generating random coins for

the signing algorithm as needed. Let F be a pseudorandom function. We assume for convenience that the pseudorandom function F has two output registers, F_1 and F_2. Moreover, let $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ be a standard public key encryption scheme. Without loss of generality we assume that the verification keys of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ and the public keys of the public-key encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ have the same length ℓ. This can always be achieved by an appropriate padding.

As we are working with an exponentially sized tree, we will define two functions $\mathsf{NodeGen}$ and $\mathsf{LeafGen}$ that provide access to the keys and thus implicitly define the tree. The $\mathsf{NodeGen}$ function generates keys for the root node and all internal nodes, whereas the $\mathsf{LeafGen}$ function generates public and private keys for the leaf nodes. More specifically, the $\mathsf{NodeGen}$ function takes as input a node identifier v and a pseudorandom function seed s and outputs a verification key $\mathsf{vk_v}$ for this node, a signature σ_v authenticating the verification keys of its children and an auxiliary value $\mathsf{x_v}$ which is the concatenation of the verification keys of the children of v.

Recall that $\|$ is the concatenation operator. In the rest of this Section and the next Section we will use the following convention. The variable ι will always run over the range $[\ell]$ and b will always run over $\{0, 1\}$.

$\mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}, s)$:
$(\mathsf{vk_v}, \mathsf{sk_v}) := \mathsf{SGen}(\mathsf{pp}; F_1(s, \mathsf{v}))$
Compute $\mathsf{vk}_{\mathsf{v}\|0}$ and $\mathsf{vk}_{\mathsf{v}\|1}$ in the same way.
$\mathsf{x_v} := \mathsf{vk}_{\mathsf{v}\|0} \| \mathsf{vk}_{\mathsf{v}\|1}$
$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{pp}, \mathsf{sk_v}, \mathsf{x})$
Output $(\mathsf{vk_v}, \sigma_\mathsf{v}, \mathsf{x_v})$

The function $\mathsf{LeafGen}$ takes as input public parameters pp, a node-identifier v of a leaf-node and a pseudorandom function seed s and outputs the verification key $\mathsf{vk_v}$ of the leaf, a signature σ_v authenticating the leaf public key, a leaf public key $\mathsf{lpk_v}$ and a leaf secret key $\mathsf{lsk_v}$.

$\mathsf{LeafGen}(\mathsf{pp}, \mathsf{v}, s)$
$(\mathsf{vk_v}, \mathsf{sk_v}) := \mathsf{SGen}(\mathsf{pp}; F_1(s, \mathsf{v}))$
$(\mathsf{lpk_v}, \mathsf{lsk_v}) := \mathsf{KG}(1^\lambda; F_2(s, \mathsf{v}))$
$\mathsf{x_v} := \mathsf{lpk_v}$
$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{pp}, \mathsf{sk_v}, \mathsf{x_v})$
Output $(\mathsf{vk_v}, \sigma_\mathsf{v}, \mathsf{lpk_v}, \mathsf{lsk_v})$

We will now provide the construction of our IBE scheme $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

$\mathsf{Setup}(1^\lambda, n)$: Choose a random seed s for the PRF F. Compute the public parameters $\mathsf{pp} := \mathsf{SSetup}(1^\lambda, 2\ell)$ and $(\mathsf{vk}_{\mathsf{v}_0}, \cdot, \cdot) := \mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}_0, s)$. Output $\mathsf{mpk} := (\mathsf{pp}, \mathsf{vk}_{\mathsf{v}_0})$ and $\mathsf{msk} := s$.

$\mathsf{KeyGen}(\mathsf{msk} = s, \mathsf{id} \in \{0,1\}^n)$: Let $\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_n$ be the root-to-leaf path for the identity id, i.e. all the prefixes of id. For $j = 0, \ldots, n-1$ compute $(\cdot, \sigma_{\mathsf{v}_j}, \mathsf{x}_{\mathsf{v}_j}) := \mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}_j, s)$. Further compute $(\cdot, \sigma_{\mathsf{id}}, \mathsf{lpk}_{\mathsf{id}}, \mathsf{lsk}_{\mathsf{id}}) := \mathsf{LeafGen}(\mathsf{pp}, \mathsf{v}_n, s)$. Output $\mathsf{sk}_{\mathsf{id}} := ((\sigma_{\mathsf{v}_0}, \mathsf{x}_{\mathsf{v}_0}), \ldots, (\sigma_{\mathsf{v}_n}, \mathsf{x}_{\mathsf{v}_n}), \sigma_{\mathsf{id}}, \mathsf{lpk}_{\mathsf{id}}, \mathsf{lsk}_{\mathsf{id}})$.

$\mathsf{Encrypt}(\mathsf{mpk} = (\mathsf{pp}, \mathsf{vk}_{\mathsf{v}_0}), \mathsf{id} \in \{0,1\}^n, \mathsf{m})$: We will first describe two circuits that will be used by the encryption algorithm.

- $\mathsf{Q}[\mathsf{pp}, \beta \in \{0,1\}, \mathsf{e}_Q = \{(Y_{\iota,0}, Y_{\iota,1})\}_\iota](\mathsf{vk})$:
  Compute and output $\{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, \beta \cdot \ell + \iota, b), Y_{\iota,b})\}_{\iota,b}$
- $\mathsf{T}[\mathsf{m}](\mathsf{lpk})$: Compute and output $\mathsf{E}(\mathsf{lpk}, \mathsf{m})$.

$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}])$
$(\tilde{\mathsf{Q}}^{(n)}, \mathsf{e}_Q^{(n)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \mathsf{e}_T])$
For $j = n-1, \ldots, 0$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}_{j+1}, \mathsf{e}_Q^{(j+1)}])$
Parse $\mathsf{e}_Q^{(0)} = \{Y_{\iota,0}, Y_{\iota,1}\}_\iota$
$\mathsf{y} := \mathsf{vk}_{\mathsf{v}_0}$
$\tilde{\mathsf{y}}^{(0)} := \{Y_{\iota,\mathsf{y}_\iota}\}_\iota$
Output $c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

$\mathsf{Decrypt}(\mathsf{sk}_{\mathsf{id}}=((\sigma_{\mathsf{v}_0},\mathsf{x}_{\mathsf{v}_0}),\ldots,(\sigma_{\mathsf{v}_n},\mathsf{x}_{\mathsf{v}_n}),\sigma_{\mathsf{id}},\mathsf{lpk}_{\mathsf{id}},\mathsf{lsk}_{\mathsf{id}}), c=(\tilde{\mathsf{y}}^{(0)},\tilde{\mathsf{Q}}^{(0)},\ldots,\tilde{\mathsf{Q}}^{(n)},\tilde{\mathsf{T}}))$
For $j = 0, \ldots, n-1$:
  $\{c_{\iota,b}^{(j)}\}_{\iota,b} := \mathsf{Eval}(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)})$
  $\tilde{\mathsf{y}}^{(j+1)} := \{\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \sigma_{\mathsf{v}_j}, \mathsf{x}_{\mathsf{v}_j}), c^{(j)}_{\iota,(\mathsf{x}_{\mathsf{v}_j})_\iota})\}_\iota$
$\{c_{\iota,b}^{(n)}\}_{\iota,b} := \mathsf{Eval}(\tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{y}}^{(n)})$
$\mathsf{z} := \mathsf{lpk}_{\mathsf{id}}$
$\tilde{\mathsf{z}} := \{\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_n}, \sigma_{\mathsf{id}}, \mathsf{z}), c^{(n)}_{\iota,\mathsf{z}_\iota})\}_\iota$
$\mathsf{f} := \mathsf{Eval}(\tilde{\mathsf{T}}, \tilde{\mathsf{z}})$
Output $\mathsf{m} := \mathsf{D}(\mathsf{lsk}_{\mathsf{id}}, \mathsf{f})$

6.1 Correctness

We will first show that our scheme is correct. Note that by correctness of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$, we have that the evaluation of $\tilde{\mathsf{Q}}^{(0)}$ on the labels $\tilde{\mathsf{y}}^{(0)}$ yields correctly formed ciphertexts of the OTSE scheme $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$. Next, by the correctness of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$, we get that the decrypted values $\tilde{\mathsf{y}}^{(1)}$ are correct labels for the next garbled circuit $\tilde{\mathsf{Q}}^{(1)}$. Repeating this argument, we can argue that all $\tilde{\mathsf{Q}}^{(j)}$ output correct encryptions that are subsequently decrypted to correct input labels of the next garbled circuit in the sequence. Finally, the circuit $\tilde{\mathsf{Q}}^{(n)}$ outputs correct encryptions of the input labels of $\tilde{\mathsf{T}}$, which are again correctly decrypted to input labels for $\tilde{\mathsf{T}}$. Finally, the correctness of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$ guarantees that $\tilde{\mathsf{T}}$ outputs a correct encryption of the plaintext m under the leaf public key $\mathsf{lpk}_{\mathsf{id}}$, and the correctness of the public-key-encryption scheme $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ ensures that the decryption function D outputs the correct plaintext m.

6.2 Proof of Security

We will now show that our scheme is fully secure.

Theorem 3. *Assume that* $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *is an* $\mathsf{IND}^{\mathsf{CPA}}$*-secure public key encryption scheme,* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is a* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure OTSE scheme and that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a garbling scheme. Then the scheme* $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a fully secure IBE scheme.*

We will split the proof of Theorem 3 into several lemmas. Let $\mathcal{A}$ be a PPT adversary with advantage ϵ against the fully secure $\mathsf{IND}^{\mathsf{IBE}}$-experiment and let in the following $\mathsf{v}_0, \ldots, \mathsf{v}_n$ always denote the root-to-leaf path for the challenge identity id^*. Consider the following hybrids.

Hybrid $\mathcal{H}_0$ is identical to the real experiment $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$, except that we replace the pseudorandom function F used for the generation of the identity keys by a lazily evaluated truly random function. In particular, each time we visit a new node during key generation we generate fresh keys for this node and store them. If these keys are needed later on, we retrieve them from a table of stored keys instead of generating new ones. By a standard argument it follows that the $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$-experiment and $\mathcal{H}_0$ are computationally indistinguishable, given that F is a pseudorandom function.

In the remaining hybrids we will only change the way the challenge ciphertext c^* is computed. First consider the computation of the challenge ciphertext c^* in the extremal hybrids $\mathcal{H}_0$ and $\mathcal{H}_{2n+3}$ (Figure 7). While in $\mathcal{H}_0$ all garbled circuits are computed by the garbling algorithm Garble, in $\mathcal{H}_{2n+3}$ all garbled circuits are simulated. Moreover, in $\mathcal{H}_{2n+3}$ the messages encrypted in the ciphertexts computed by the garbled circuits do not depend on the bit b, i.e. decryption of these ciphertexts always yields the same labels, regardless of which message-signature pair has been used to decrypt the encrypted labels. Notice that in $\mathcal{H}_{2n+3}$ the garbled circuit $\tilde{\mathsf{T}}$ is simulated using $\mathsf{f} := \mathsf{E}(\mathsf{lpk}_{\mathsf{id}^*}, \mathsf{m}^*)$, the encryption of the challenge message m^* under the leaf public key $\mathsf{lpk}_{\mathsf{id}^*}$.

$\mathcal{H}_0$:
$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n)}, \mathsf{e}_Q^{(n)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \mathsf{e}_T])$
For $j = n-1, \ldots, 0$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}_{j+1}, \mathsf{e}_Q^{(j+1)}])$
$\mathsf{y} := \mathsf{vk}_{\mathsf{v}_0}$
$\tilde{\mathsf{y}}^{(0)} := \{Y_{\iota, \mathsf{y}_\iota}\}_\iota$
$c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

$\mathcal{H}_{2n+3}$:
$\mathsf{f} := \mathsf{E}(\mathsf{lpk}_{\mathsf{id}^*}, \mathsf{m}^*)$
$(\tilde{\mathsf{T}}, \tilde{\mathsf{z}}) := \mathsf{GCSim}(\mathsf{T}, \mathsf{f})$
$\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{id}^*}, \iota, b), \tilde{\mathsf{z}}_\iota)\}_{\iota, b}$
$(\tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{y}}^{(n)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_T)$
For $j = n-1, \ldots, 0$
  $\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \mathsf{id}^*_{j+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota, b}$
  $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
Output $c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

Fig. 7: The extremal hybrids $\mathcal{H}_0$ and $\mathcal{H}_{2n+3}$

We will show indistinguishability of $\mathcal{H}_0$ and $\mathcal{H}_{2n+3}$ via the following hybrids. For $i = 0, \ldots, n-1$ define:

Hybrid $\mathcal{H}_{2i+1}$ This hybrid is the same as $\mathcal{H}_{2i}$, except that we change the way $\tilde{\mathsf{Q}}^{(i)}$ and $\tilde{\mathsf{y}}^{(i)}$ are computed. Compute $\tilde{\mathsf{Q}}^{(i)}$ and $\tilde{\mathsf{y}}^{(i)}$ by $(\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(i)})$.

Hybrid $\mathcal{H}_{2(i+1)}$ This hybrid is identical to $\mathcal{H}_{2i+1}$, except for the following change. Instead of computing $\mathsf{f}_Q^{(i+1)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{i+1} \cdot \ell + \iota, b), Y_{\iota,b})\}_{\iota,b}$ we compute $\mathsf{f}_Q^{(i+1)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{i+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(i+1)})\}_{\iota,b}$

The final 3 hybrids are given as follows.

Hybrid $\mathcal{H}_{2n+1}$ This hybrid is the same as $\mathcal{H}_{2n}$, except that we change the way $\tilde{\mathsf{Q}}^{(n)}$ and $\tilde{\mathsf{y}}^{(n)}$ are computed. Compute $\tilde{\mathsf{Q}}^{(n)}$ and $\tilde{\mathsf{y}}^{(n)}$ by $(\tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{y}}^{(n)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_T)$, where $\mathsf{f}_T = \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{id}^*}, \iota, b), Z_{\iota,b})\}_{\iota,b}$.

Hybrid $\mathcal{H}_{2n+2}$ This hybrid is the same as $\mathcal{H}_{2n+1}$, except that we change how f_T is computed. Let $\mathsf{e}_T = \{Z_{\iota,0}, Z_{\iota,1}\}_\iota$. Instead of computing f_T by $\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{id}^*}, \iota, b), Z_{\iota,b})\}_{\iota,b}$ we compute $\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{id}^*}, \iota, b), \tilde{\mathsf{z}}_\iota^{(n)})\}_{\iota,b}$.

Hybrid $\mathcal{H}_{2n+3}$ This hybrid is the same as $\mathcal{H}_{2n+2}$, except that we change the way $\tilde{\mathsf{T}}$ and $\tilde{\mathsf{z}}$ are computed. Compute $\tilde{\mathsf{T}}$ and $\tilde{\mathsf{z}}$ by $(\tilde{\mathsf{T}}, \tilde{\mathsf{z}}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f})$, where $\mathsf{f} := \mathsf{E}(\mathsf{lpk}_{\mathsf{id}}, \mathsf{m})$.

Lemma 1. *We claim that for* $i = 0, \ldots, n-1$ *the hybrids* $\mathcal{H}_{2i}$ *and* $\mathcal{H}_{2i+1}$ *are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme.*

Proof. Assume towards contradiction that $\mathcal{A}$ distinguishes between $\mathcal{H}_{2i}$ and $\mathcal{H}_{2i+1}$ with non-negligible advantage ϵ. We will construct a distinguisher $\mathcal{R}^{\mathcal{A}}$ that breaks the security of the garbling scheme with advantage ϵ. $\mathcal{R}$ simulates the $\mathcal{H}_{2i}$ experiment faithfully with the adversary $\mathcal{A}$ until $\mathcal{A}$ requests a challenge ciphertext. Once $\mathcal{A}$ does request the challenge ciphertext, $\mathcal{R}$ computes

$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n)}, \mathsf{e}_Q^{(n)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \mathsf{e}_T])$
For $j = n-1, \ldots, i+1$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{j+1}, \mathsf{e}_Q^{(j+1)}])$.
$(\tilde{\mathsf{Q}}^{(i)}, \mathsf{e}_Q^{(i)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{i+1}, \mathsf{e}_Q^{(i+1)}])$

and sends the circuit $\mathsf{Q}[\mathsf{e}_Q^{(i)}]$ and the input $\mathsf{y}^{(i)}$ to the experiment. Once the experiment returns $\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}$, $\mathcal{R}$ computes

For $j = i-1, \ldots, 0$
  $\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \mathsf{id}^*_{j+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota,b}$
  $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
$c^* := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

and returns c^* to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation of the $\mathcal{H}_{2i}$ experiment and outputs whatever the simulated $\mathcal{H}_{2i}$ experiment outputs.

Clearly, if $\mathcal{R}$'s challenge $\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}$ is distributed according to the real distribution, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is identical to the view of $\mathcal{A}$ in $\mathcal{H}_{2i}$. On the other hand, if $\mathcal{R}$'s challenge is distributed according to the simulated distribution, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is identical to the view of $\mathcal{A}$ in $\mathcal{H}_{2i+1}$. We conclude that

$$\mathsf{Adv}(\mathcal{R}^{\mathcal{A}}) = |\Pr[\mathcal{H}_{2i}(\mathcal{A}) = 1] - \Pr[\mathcal{H}_{2i+1}(\mathcal{A}) = 1]| \leq \epsilon,$$

which contradicts the security of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.

Lemma 2. *We claim that for $i = 0, \ldots, n-1$ the hybrids $\mathcal{H}_{2i+1}$ and $\mathcal{H}_{2(i+1)}$ are computationally indistinguishable, given that $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ is a selectively $\mathsf{IND}^{\mathsf{OTSE}}$-secure OTSE scheme.*

Proof. Let q be the number of queries by $\mathcal{A}$ (including the challenge query), which gives us an upper bound for the number of distinct nodes visited at level i. We will construct an adversary $\mathcal{R}^{\mathcal{A}}$ that breaks the $\mathsf{IND}^{\mathsf{OTSE}}$-security of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ in the multi-challenge setting with advantage ϵ/q. $\mathcal{R}$ first guesses an index $k^* \in [q]$. $\mathcal{R}$ then generates keys

$$(\mathsf{vk}_0^*, \mathsf{sk}_0^*) := \mathsf{SGen}(\mathsf{pp})$$
$$(\mathsf{vk}_1^*, \mathsf{sk}_1^*) := \mathsf{SGen}(\mathsf{pp})$$

and sets $\mathsf{x}^* := \mathsf{vk}_{\mathsf{v}_i\|0} \| \mathsf{vk}_{\mathsf{v}_i\|1}$ and sends the challenge message x^* to the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment and receives a verification key vk and a signature σ from the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment.

$\mathcal{R}$ continues simulating the $\mathcal{H}_{2i+1}$ experiment. Once the k^*-th distinct node v^* on level i is visited, $\mathcal{R}$ modifies the $\mathsf{NodeGen}$ function for this node as follows.

$\mathsf{vk}_{\mathsf{v}^*} := \mathsf{vk}$
$\mathsf{vk}_{\mathsf{v}^*\|0} := \mathsf{vk}_0^*$
$\mathsf{sk}_{\mathsf{v}^*\|0} := \mathsf{sk}_0^*$
$\mathsf{vk}_{\mathsf{v}^*\|1} := \mathsf{vk}_1^*$
$\mathsf{sk}_{\mathsf{v}^*\|1} := \mathsf{sk}_1^*$
$\mathsf{x}_{\mathsf{v}^*} := \mathsf{vk}_{\mathsf{v}^*\|0} \| \mathsf{vk}_{\mathsf{v}^*\|1}$
$\sigma_{\mathsf{v}^*} := \sigma$
Output $(\mathsf{vk}_{\mathsf{v}^*}, \sigma_{\mathsf{v}^*}, \mathsf{x}_{\mathsf{v}^*})$

When the corresponding signing keys are required for the $\mathsf{NodeGen}$ procedure on $\mathsf{v}^*\|0$ and $\mathsf{v}^*\|1$, use the corresponding signing keys $\mathsf{sk}_{\mathsf{v}^*\|0}$ and $\mathsf{sk}_{\mathsf{v}^*\|1}$ computed in the modified procedure above.

$\mathcal{R}$ now continues the simulation. Once $\mathcal{A}$ requests a challenge-ciphertext for an identity id^*, $\mathcal{R}$ checks if v^* is on the root-to-leaf path for id^* (i.e. if v^* is a prefix of id^*), and if not aborts and outputs a random bit. Otherwise, $\mathcal{R}$ generates the challenge ciphertext c^* for $\mathcal{A}$ in the following way.

$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n)}, \mathsf{e}_Q^{(n)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 0, \mathsf{e}_T])$
For $j = n-1, \ldots, i+1$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{j+1}, \mathsf{e}_Q^{(j+1)}])$
Parse $\mathsf{e}_Q^{(i+1)} = \{(Y_{\iota,0}, Y_{\iota,1})\}_\iota$

$\mathcal{R}$ now computes the messages $M_0^* := \{Y_{\iota,1-\mathsf{x}_\iota^{(i+1)}}\}_\iota$ and $M_1^* := \{Y_{\iota,\mathsf{x}_\iota^{(i+1)}}\}_\iota$, sends the challenge messages (M_0^*, M_1^*) to the $\mathsf{IND}^{\mathsf{OTSE}}$ experiment and receives a challenge ciphertext $C^* = (C_1^*, \ldots, C_\ell^*)$. Now $\mathcal{R}$ computes $\mathsf{f}_Q^{(i+1)}$ by $\mathsf{f}_Q^{(i+1)} := \{C_{\iota,b}\}_\iota$, where $C_{\iota,\mathsf{x}_\iota} := \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{i+1} \cdot \ell + \iota, \mathsf{x}_\iota^{(i+1)}), Y_{\iota,\mathsf{x}_\iota^{(i+1)}})$ and $C_{\iota,1-\mathsf{x}_\iota} := C_\iota^*$. $\mathcal{R}$ continues the computation of the challenge ciphertext as follows.

$(\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(i)})$
For $j = i-1, \ldots, 0$
  $\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \mathsf{id}^*_{j+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota,b}$
  $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
$c^* := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

and returns c^* to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation of the $\mathcal{H}_{2i+1}$ experiment and outputs whatever the simulated $\mathcal{H}_{2i+1}$ experiment outputs.

We will now compute the advantage of $\mathcal{R}^{\mathcal{A}}$. First notice that the keys provided by $\mathcal{R}$ to $\mathcal{A}$ are distributed exactly as in $\mathcal{H}_{2i+1}$ (and therefore do not depend on k^*). If $\mathcal{R}$ guesses k^* wrongly its advantage is 0. Let E be the event that k^* has been guessed correctly. It clearly holds that $\Pr[\mathsf{E}] \geq 1/q$. Assume now that the event E holds. If the challenge bit b^* of the $\mathsf{IND}^{\mathsf{OTSE}}$ experiment is 0, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is distributed exactly as in experiment $\mathcal{H}_{2i+1}$. On the other hand, if $b^* = 1$ then the view of $\mathcal{A}$ is distributed exactly as in experiment $\mathcal{H}_{2(i+1)}$. Thus we can conclude

$$\begin{aligned}\mathsf{Adv}(\mathcal{R}^{\mathcal{A}}) &= \Pr[\mathsf{E}] \cdot |\Pr[\mathcal{H}_{2i+1}(\mathcal{A}) = 1] - \Pr[\mathcal{H}_{2(i+1)}(\mathcal{A}) = 1]| \\ &\geq \Pr[\mathsf{E}] \cdot \epsilon \\ &\geq \epsilon/q.\end{aligned}$$

Lemma 3. *We claim that the hybrids $\mathcal{H}_{2n}$ and $\mathcal{H}_{2n+1}$ are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme.*

The proof proceeds analogous to the proof of Lemma 1.

Lemma 4. *We claim that the hybrids $\mathcal{H}_{2n+1}$ and $\mathcal{H}_{2n+2}$ are computationally indistinguishable, given that the OTSE-scheme* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure.*

The proof follows analogous to the proof of Lemma 2.

Lemma 5. *We claim that the hybrids $\mathcal{H}_{2n+2}$ and $\mathcal{H}_{2n+3}$ are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval}, \mathsf{GCSim})$ *is a secure garbling scheme.*

Again, the proof follows analogous to the proof of Lemma 1.

Lemma 6. *The advantage of $\mathcal{A}$ in $\mathcal{H}_{2n+3}$ is negligible, given that* $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *is* $\mathsf{IND}^{\mathsf{CPA}}$*-secure.*

Proof. We will construct an adversary $\mathcal{R}^{\mathcal{A}}$ that breaks the $\mathsf{IND}^{\mathsf{CPA}}$ security of $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ with advantage ϵ. $\mathcal{R}$ simulates $\mathcal{H}_{2n+3}$ faithfully, with the exception that it uses its own challenge public key pk^* as public key for the leaf id^*, i.e. it sets $\mathsf{lpk}_{\mathsf{id}^*} := \mathsf{pk}^*$. It forwards $\mathcal{A}$'s challenge messages m_0 and m_1 to the $\mathsf{IND}^{\mathsf{CPA}}$ experiment and uses its own challenge ciphertext c^* as the ciphertext f in the computation of the challenge ciphertext c^*. It follows that $\mathcal{R}$ simulates $\mathcal{H}_{4n+3}$ perfectly and therefore $\mathsf{Adv}_{\mathsf{IND}^{\mathsf{CPA}}}(\mathcal{R}^{\mathcal{A}}) = \mathsf{Adv}_{\mathcal{H}_{4n+3}}(\mathcal{A})$.

This concludes the proof of Theorem 3.

7 Achieving Selectively Secure HIBE

We will now add a delegation mechanism to the IBE scheme constructed in the last Section, yielding the construction of a hierarchical IBE scheme. The basic idea is as follows. Instead of putting the public keys of the $\mathsf{IND}^{\mathsf{CPA}}$-secure scheme only into the leaf nodes of the tree, we will put such public keys into every node of the tree. This means that every node of the (unbounded size) tree can effectively be used in the same way we used the leaf nodes in the scheme of the last Section.

Since we want to be able to delegate the ability to delegate HIBE keys for entire sub-trees, we need to work with a pseudorandom function supporting this kind of delegation. We therefore use the *delegatable pseudorandom functions* defined in Section 2.6 for this task.

In our scheme, the delegated master secret key for an identity id consist of the identity secret key for id and a delegated PRF seed s_{id}. This enables the delegator to compute identity secret keys for all the nodes in the sub-tree of id.

Let $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ be an $\mathsf{IND}^{\mathsf{OTSE}}$-secure OTSE scheme, $(F, F.\mathsf{Delegate})$ be a delegatable pseudorandom function and $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ be a standard public key encryption scheme. We assume for convenience that the pseudorandom function F has two output registers, F_1 and F_2. Assume that both the verification keys of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ and the public keys of $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ have length ℓ and let $d = 3\ell$.

Again, we will first define a function $\mathsf{NodeGen}$ that provides access to the keys stored in the tree. As mentioned above, we do not make distinctions between leaf nodes and non-leaf nodes anymore but store a local public key $\mathsf{lpk}_{\mathsf{v}}$ at every node v. $\mathsf{NodeGen}$ takes as input a node identifier v and a pseudorandom function seed s and outputs a verification key vk_{v}, signatures σ_{v}, auxiliary information x_{v} and a secret key $\mathsf{lsk}_{\mathsf{v}}$. Again, we use the convention that the variable ι runs over $[\ell]$ and b over $\{0, 1\}$.

$\mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}, s)$:
$(\mathsf{vk}_{\mathsf{v}}, \mathsf{sk}_{\mathsf{v}}) := \mathsf{SGen}(\mathsf{pp}; F_1(s, \mathsf{v}))$

Compute $\mathsf{vk}_{\mathsf{v}\|0}$ and $\mathsf{vk}_{\mathsf{v}\|1}$ in the same way.
$(\mathsf{lpk}_\mathsf{v}, \mathsf{lsk}_\mathsf{v}) := \mathsf{KG}(1^\lambda; F_2(s, \mathsf{v}))$
$\mathsf{x}_\mathsf{v} := \mathsf{vk}_{\mathsf{v}\|0} \| \mathsf{vk}_{\mathsf{v}\|1} \| \mathsf{lpk}_\mathsf{v}$
$\sigma_\mathsf{v} := \mathsf{SSign}(\mathsf{pp}, \mathsf{sk}_\mathsf{v}, \mathsf{x}_\mathsf{v})$
Output $(\mathsf{vk}_\mathsf{v}, \sigma_\mathsf{v}, \mathsf{x}_\mathsf{v}, \mathsf{lsk}_\mathsf{v})$

The HIBE scheme $(\mathsf{Setup}, \mathsf{Delegate}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is given by the following algorithms.

$\mathsf{Setup}(1^\lambda)$: Let v_0 be the root-node. Choose a random seed s for the pseudo-random function F. Compute $\mathsf{pp} := \mathsf{SSetup}(1^\lambda, 3 \cdot \ell)$ and $(\mathsf{vk}_{\mathsf{v}_0}, \cdot, \cdot, \cdot) := \mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}_0, s)$. Output $\mathsf{mpk} := \mathsf{vk}_{\mathsf{v}_0}$ and $\mathsf{msk} := s$.

$\mathsf{Delegate}(\mathsf{msk} = s, \mathsf{id} \in \{0,1\}^*)$: Set $n := |\mathsf{id}|$. Let $\mathsf{v}_0, \mathsf{v}_1, \ldots, \mathsf{v}_n$ be the root-to-leaf path for the identity id, i.e. all the prefixes of id. For $j = 0, \ldots, n-1$ compute $(\cdot, \sigma_{\mathsf{v}_j}, \mathsf{x}_\mathsf{v}, \cdot) := \mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}_j, s)$. Compute $s_\mathsf{id} := F.\mathsf{Delegate}(s, \mathsf{id})$. Output $((\sigma_{\mathsf{v}_0}, \mathsf{x}_{\mathsf{v}_0}), \ldots, (\sigma_{\mathsf{v}_n}, \mathsf{x}_{\mathsf{v}_n}), s_\mathsf{id})$[9]

$\mathsf{KeyGen}(\mathsf{msk}^{\mathsf{HIBE}}_{\mathsf{id}'} = ((\sigma_{\mathsf{v}_0}, \mathsf{x}_{\mathsf{v}_0}), \ldots, (\sigma_{\mathsf{v}_{|\mathsf{id}'|}}, \mathsf{x}_{\mathsf{v}_{|\mathsf{id}'|}}), s_{\mathsf{id}'}), \mathsf{id} \in \{0,1\}^*)$: Set $n := |\mathsf{id}|$. Let $\mathsf{v}_{|\mathsf{id}'|}, \ldots, \mathsf{v}_{|\mathsf{id}'|+|\mathsf{id}|}$ be the path from id' to $\mathsf{id}' \| \mathsf{id}$, i.e. id' concatenated with all the prefixes of id. For $j = |\mathsf{id}'|, \ldots, |\mathsf{id}'|+|\mathsf{id}|-1$ compute $(\cdot, \sigma_{\mathsf{v}_j}, \mathsf{x}_\mathsf{v}, \mathsf{lsk}_\mathsf{v}) := \mathsf{NodeGen}(\mathsf{pp}, \mathsf{v}_j, s_{\mathsf{id}'})$. Output $\mathsf{sk}_\mathsf{id} := ((\sigma_{\mathsf{v}_0}, \mathsf{x}_{\mathsf{v}_0}), \ldots, (\sigma_{\mathsf{v}_{|\mathsf{id}'|+|\mathsf{id}|}}, \mathsf{x}_{\mathsf{v}_{|\mathsf{id}'|+|\mathsf{id}|}}), \sigma_\mathsf{id}, \mathsf{lsk}_\mathsf{id})$

$\mathsf{Encrypt}(\mathsf{mpk} = \mathsf{vk}_{\mathsf{v}_0}, \mathsf{id} \in \{0,1\}^*, \mathsf{m})$: We will first describe two circuits that will be used by the encryption algorithm. The mode $\beta = 2$ of the circuit Q targets a local public key.

- $\mathsf{Q}[\mathsf{pp}, \beta \in \{0,1,2\}, \mathsf{e}_Q = \{(Y_{\iota,0}, Y_{\iota,1})\}_\iota](\mathsf{vk})$: Compute and then output $\{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}, \beta \cdot \ell + \iota, b), Y_{\iota,b})\}_{\iota,b}$
- $\mathsf{T}[\mathsf{m}](\mathsf{lpk})$: Compute and output $\mathsf{E}(\mathsf{lpk}, \mathsf{m})$.

$n := |\mathsf{id}|$
$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}])$
$(\tilde{\mathsf{Q}}^{(n)}, \mathsf{e}_Q^{(n)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 2, \mathsf{e}_T])$
For $j = n-1, \ldots, 0$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}_{j+1}, \mathsf{e}_Q^{(j+1)}])$
Parse $\mathsf{e}_Q^{(0)} = \{Y_{\iota,0}, Y_{\iota,1}\}_\iota$
$\mathsf{y} := \mathsf{mpk}_{\mathsf{v}_0}$
$\tilde{\mathsf{y}}^{(0)} := \{Y_{\iota,\mathsf{y}_\iota}\}_\iota$
Output $c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}})$

$\mathsf{Decrypt}(\mathsf{sk}_\mathsf{id} = ((\sigma_{\mathsf{v}_0}, \mathsf{x}_{\mathsf{v}_0}), \ldots, (\sigma_{\mathsf{v}_n}, \mathsf{x}_{\mathsf{v}_n}), \sigma_\mathsf{id}, \mathsf{lpk}_\mathsf{id}, \mathsf{lsk}_\mathsf{id}), c = (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{T}}))$
For $i = 0, \ldots, n-1$:
  $\{c_{\iota,b}^{(i)}\}_{\iota,b} := \mathsf{Eval}(\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)})$

[9] To delegate keys from delegated keys at an identity id, treat id as a root node, compute the delegated keys, and the concatenate the root-to-node paths.

$$\tilde{\mathsf{y}}^{(i+1)} := \{\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \sigma_{\mathsf{v}_i}, \mathsf{x}_{\mathsf{v}_i}), c^{(i)}_{\iota,(\mathsf{x}_{\mathsf{v}_i})_\iota})\}_\iota$$
$$\{c^{(n)}_{\iota,b}\}_{\iota,b} := \mathsf{Eval}(\tilde{\mathsf{Q}}^{(n)}, \tilde{\mathsf{y}}^{(n)})$$
$$\mathsf{z} := \mathsf{lpk}_{\mathsf{id}}$$
$$\tilde{\mathsf{z}} := \{\mathsf{SDec}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_n}, \sigma_{\mathsf{id}}, \mathsf{z}), c^{(n)}_{\iota,\mathsf{z}_\iota})\}_\iota$$
$$c^\dagger := \mathsf{Eval}(\tilde{\mathsf{T}}, \tilde{\mathsf{z}})$$
$$\text{Output } \mathsf{m} := \mathsf{D}(\mathsf{lsk}_{\mathsf{id}}, c^\dagger)$$

7.1 Correctness

Correctness of the scheme follows by the same argument as for the scheme in Section 6. Moreover, correctness of the delegation mechanism follows directly from the the correctness of the delegation mechanism $F.\mathsf{Delegate}$.

7.2 Proof of Security

We will now show that our scheme is $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}$-secure.

Theorem 4. *Assume that* $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *is an* $\mathsf{IND}^{\mathsf{CPA}}$*-secure public-key-encryption scheme,* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is an* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure one-time signature with encryption scheme and that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a garbling scheme. Then* $(\mathsf{Setup}, \mathsf{Delegate}, \mathsf{KeyGen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a* $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}$*-secure HIBE scheme.*

We will split the proof of Theorem 4 into several lemmas. Let $\mathcal{A}$ be a PPT adversary with advantage ϵ against the $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}$-experiment, let id^* be the challenge identity, which is selectively chosen by $\mathcal{A}$ at the beginning of the experiment and let $n^* := |\mathsf{id}^*|$ be the length of the challenge identity. Let in the following $\mathsf{v}_0, \ldots, \mathsf{v}_{n^*}$ always denote the root-to-leaf path for the challenge identity id^*.

We will start by providing an overview of the hybrids.

Hybrid $\mathcal{H}_0$ This hybrid is identical to the real experiment $\mathsf{sel\text{-}IND}^{\mathsf{HIBE}}_{\mathcal{A}}$, except that on the challenge-path $\mathsf{v}_0, \ldots, \mathsf{v}_n$ we replace the pseudorandom function F used for the generation of the identity keys by a function H, which is truly random on the path from the root to the challenge identity and identical to $F(s, \cdot)$ everywhere else. This means, in particular, that we can choose the all the keys on the path from the root to the challenge identity in advance and with truly random coins. It follows directly from the pseudorandomness property of the delegatable pseudorandom function $(F, F.\mathsf{Delegate})$ that the experiments $\mathsf{IND}^{\mathsf{IBE}}(\mathcal{A})$ and $\mathcal{H}_0$ are computationally indistinguishable.

In the remaining hybrids, we will only change the way the challenge ciphertext c^* is computed. For $i = 0, \ldots, n^* - 1$ we define the hybrids $\mathcal{H}_0, \ldots, \mathcal{H}_{2n^*+3}$. As in the last Section, we will first provide an overview of the extremal hybrids $\mathcal{H}_0$ and $\mathcal{H}_{2n^*+3}$ in Figure 8.

For $i = 0, \ldots, n^* - 1$ define the following hybrids.

$\mathcal{H}_0$:
$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n^*)}, \mathsf{e}_Q^{(n^*)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 2, \mathsf{e}_T])$
For $j = n^* - 1, \ldots, 0$
$(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{j+1}, \mathsf{e}_Q^{(j+1)}])$
$\mathsf{y} := \mathsf{pk}_{\mathsf{v}_0}$
$\tilde{\mathsf{y}}^{(0)} := \{Y_{\iota, \mathsf{y}_\iota}\}_\iota$
$c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{T}})$

$\mathcal{H}_{2n^*+3}$:
$\mathsf{f} := \mathsf{E}(\mathsf{lpk}_{\mathsf{id}^*}, \mathsf{m}^*)$
$(\tilde{\mathsf{T}}, \tilde{\mathsf{z}}) := \mathsf{GCSim}(\mathsf{T}, \mathsf{f})$
$\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{id}^*}, \iota, b), \tilde{\mathsf{z}}_\iota)\}_{\iota, b}$
$(\tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{y}}^{(n^*)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_T)$
For $j = n^* - 1, \ldots, 0$
$\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \mathsf{id}^*_{j+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota, b}$
$(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
Output $c := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{T}})$

Fig. 8: The extremal hybrids $\mathcal{H}_0$ and $\mathcal{H}_{2n^*+3}$

Hybrid $\mathcal{H}_{2i+1}$ This hybrid is the same as $\mathcal{H}_{2i}$, except that we change the way $\tilde{\mathsf{Q}}^{(i)}$ and $\tilde{\mathsf{y}}^{(i)}$ are computed. Compute $\tilde{\mathsf{Q}}^{(i)}$ and $\tilde{\mathsf{y}}^{(i)}$ by $(\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(i)})$.

Hybrid $\mathcal{H}_{2(i+1)}$ This hybrid is identical to $\mathcal{H}_{2i+1}$, except for the following change. Instead of computing $\mathsf{f}_Q^{(i+1)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{i+1} \cdot \ell + \iota, b), Y_{\iota,b})\}_{\iota,b}$ we compute $\mathsf{f}_Q^{(i+1)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{i+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(i+1)})\}_{\iota,b}$

The final 3 hybrids are given as follows.

Hybrid $\mathcal{H}_{2n^*+1}$ This hybrid is the same as $\mathcal{H}_{2n^*}$, except that we change the way $\tilde{\mathsf{Q}}^{(n^*)}$ and $\tilde{\mathsf{y}}^{(n^*)}$ are computed. Compute $\tilde{\mathsf{Q}}^{(n^*)}$ and $\tilde{\mathsf{y}}^{(n^*)}$ by $(\tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{y}}^{(n^*)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_T)$.

Hybrid $\mathcal{H}_{2n^*+2}$ This hybrid is the same as $\mathcal{H}_{2n^*+1}$, except that we change how f_T is computed. Let $\mathsf{e}_T = \{Z_{\iota,0}, Z_{\iota,1}\}_\iota$. Instead of computing f_T by $\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_{n^*}}, \iota, b), Z_{\iota,b})\}_{\iota,b}$ we compute $\mathsf{f}_T := \{(\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{n^*}, \iota, b), \tilde{\mathsf{z}}_\iota^{(n^*)})\}_{\iota,b}$.

Hybrid $\mathcal{H}_{2n^*+3}$ This hybrid is the same as $\mathcal{H}_{2n^*+2}$, except that we change the way $\tilde{\mathsf{T}}$ and $\tilde{\mathsf{z}}$ are computed. Compute $\tilde{\mathsf{T}}$ and $\tilde{\mathsf{z}}$ by $(\tilde{\mathsf{T}}, \tilde{\mathsf{z}}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f})$, where $\mathsf{f} := \mathsf{E}(\mathsf{lpk}_{\mathsf{id}^*}, \mathsf{m}^*)$.

Lemma 7. *We claim that for $i = 0, \ldots, n^* - 1$ the hybrids $\mathcal{H}_{2i}$ and $\mathcal{H}_{2i+1}$ are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme.*

Proof. Assume towards contradiction that $\mathcal{A}$ distinguishes between $\mathcal{H}_{2i}$ and $\mathcal{H}_{2i+1}$ with non-negligible advantage ϵ. We will construct a distinguisher $\mathcal{R}^{\mathcal{A}}$ that breaks the security of the garbling scheme with advantage ϵ. $\mathcal{R}$ simulates the $\mathcal{H}_{2i}$ experiment faithfully with the adversary $\mathcal{A}$ until $\mathcal{A}$ requests a challenge ciphertext. Once $\mathcal{A}$ does request the challenge ciphertext, $\mathcal{R}$ computes

$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n^*)}, \mathsf{e}_Q^{(n^*)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 2, \mathsf{e}_T])$
For $j = n^* - 1, \ldots, i+1$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{j+1}, \mathsf{e}_Q^{(j+1)}])$.
$(\tilde{\mathsf{Q}}^{(i)}, \mathsf{e}_Q^{(i)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}^*_{i+1}, \mathsf{e}_Q^{(i+1)}])$

and sends the circuit $\mathsf{Q}[\mathsf{e}_Q^{(i)}]$ and the input $\mathsf{y}^{(i)}$ to the experiment. Once the experiment returns $\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}$, $\mathcal{R}$ computes

For $j = i-1, \ldots, 0$
  $\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_i}, \mathsf{id}^*_{j+1} \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota,b}$
  $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
$c^* := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{T}})$

and returns c^* to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation of the $\mathcal{H}_{2i}$ experiment and outputs whatever the simulated $\mathcal{H}_{2i}$ experiment outputs.

Clearly, if $\mathcal{R}$'s challenge $\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}$ is distributed according to the real distribution, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is identical to the view of $\mathcal{A}$ in $\mathcal{H}_{2i}$. On the other hand, if $\mathcal{R}$'s challenge is distributed according to the simulated distribution, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is identical to the view of $\mathcal{A}$ in $\mathcal{H}_{2i+1}$. We conclude that

$$\mathsf{Adv}(\mathcal{R}^{\mathcal{A}}) = |\Pr[\mathcal{H}_{2i}(\mathcal{A}) = 1] - \Pr[\mathcal{H}_{2i+1}(\mathcal{A}) = 1]| \leq \epsilon,$$

which contradicts the security of the garbling scheme $(\mathsf{Garble}, \mathsf{Eval})$.

Lemma 8. *We claim that for $i = 0, \ldots, n^*-1$ the hybrids $\mathcal{H}_{2i+1}$ and $\mathcal{H}_{2(i+1)}$ are computationally indistinguishable, given that $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ is an $\mathsf{IND}^{\mathsf{OTSE}}$-secure IBE scheme.*

Proof. We will construct an adversary $\mathcal{R}^{\mathcal{A}}$ that breaks the multi-challenge $\mathsf{IND}^{\mathsf{OTSE}}$-security of $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ with advantage ϵ. Let $\mathsf{v}^* = \mathsf{v}_i$ be the i-th node on the challenge-path. Let pp be the public parameters passed to $\mathcal{R}$. $\mathcal{R}$ first generates keys for the children $\mathsf{v}^*\|0$ and $\mathsf{v}^*\|1$ of v^* by

$$(\mathsf{vk}_b^*, \mathsf{sk}_b^*) := \mathsf{SGen}(\mathsf{pp})$$

if $\mathsf{v}^*\|b$ is on the challenge path and

$$(\mathsf{vk}_b^*, \mathsf{sk}_b^*) := \mathsf{SGen}(\mathsf{pp}; F(s, \mathsf{v}^*\|b))$$

otherwise. Next, $\mathcal{R}$ generates the local key $\mathsf{lpk}_{\mathsf{v}^*}$ by $(\mathsf{lpk}^*, \mathsf{lsk}^*) := \mathsf{KeyGen}(1^\lambda)$. Now $\mathcal{R}$ sets $\mathsf{x}^* := \mathsf{vk}_{\mathsf{v}^*\|0}\|\mathsf{vk}_{\mathsf{v}^*\|1}\|\mathsf{lpk}_{\mathsf{v}^*}$, sends the challenge message x^* to the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment and receives a verification key vk and a signature σ.

$\mathcal{R}$ now chooses the keys for all nodes on the root-to-leaf path as in $\mathcal{H}_{2i+1}$, except for the keys of v^*, which are chosen as follows.

$\mathsf{vk}_{\mathsf{v}^*} := \mathsf{vk}$
$\mathsf{vk}_{\mathsf{v}^*\|0} := \mathsf{vk}_0^*$
$\mathsf{sk}_{\mathsf{v}^*\|0} := \mathsf{sk}_0^*$
$\mathsf{vk}_{\mathsf{v}^*\|1} := \mathsf{vk}_1^*$
$\mathsf{sk}_{\mathsf{v}^*\|1} := \mathsf{sk}_1^*$
$\mathsf{x}_{\mathsf{v}^*} := \mathsf{vk}_{\mathsf{v}^*\|0}\|\mathsf{vk}_{\mathsf{v}^*\|1}\|\mathsf{lpk}_{\mathsf{v}^*}$
$\sigma_{\mathsf{v}^*} := \sigma$
Output $(\mathsf{vk}_{\mathsf{v}^*}, \sigma_{\mathsf{v}^*}, \mathsf{x}_{\mathsf{v}^*}, \mathsf{lsk}_{\mathsf{v}^*})$

When the corresponding secret keys are required for the $\mathsf{NodeGen}$ procedure on $\mathsf{v}^*\|0$ and $\mathsf{v}^*\|1$, use the corresponding secret keys $\mathsf{sk}_{\mathsf{v}^*\|0}$ and $\mathsf{sk}_{\mathsf{v}^*\|1}$ set above in the modified procedure above.

$\mathcal{R}$ now continues the simulation. Once $\mathcal{A}$ requests a challenge-ciphertext for the identity id^*, $\mathcal{R}$ generates the challenge ciphertext c^* for $\mathcal{A}$ in the following way.

$(\tilde{\mathsf{T}}, \mathsf{e}_T) := \mathsf{Garble}(1^\lambda, \mathsf{T}[\mathsf{m}^*])$
$(\tilde{\mathsf{Q}}^{(n^*)}, \mathsf{e}_Q^{(n^*)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, 2, \mathsf{e}_T])$
For $j = n^* - 1, \ldots, i+1$
  $(\tilde{\mathsf{Q}}^{(j)}, \mathsf{e}_Q^{(j)}) := \mathsf{Garble}(1^\lambda, \mathsf{Q}[\mathsf{pp}, \mathsf{id}_{j+1}^*, \mathsf{e}_Q^{(j+1)}])$
Parse $\mathsf{e}_Q^{(i+1)} = \{(Y_{\iota,0}, Y_{\iota,1})\}_\iota$

$\mathcal{R}$ now computes the messages $M_0^* := \{Y_{\iota,1-\mathsf{x}_\iota^{(i+1)}}\}_\iota$ and $M_1^* := \{Y_{\iota,\mathsf{x}_\iota^{(i+1)}}\}_\iota$, sends the challenge messages (M_0^*, M_1^*) to the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment and receives a challenge ciphertext $C^* = (C_1^*, \ldots, C_\ell^*)$. Now $\mathcal{R}$ computes $\mathsf{f}_Q^{(i+1)}$ by $\mathsf{f}_Q^{(i+1)} := \{C_{\iota,b}\}_{\iota\in[\ell]}$, where $C_{\iota,\mathsf{x}_\iota} := \mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}^*}, \beta\cdot\ell+\iota, \mathsf{x}_\iota^{(i+1)}), Y_{\iota,\mathsf{x}_\iota^{(i+1)}})$ and $C_{\iota,1-\mathsf{x}_\iota} := C_\iota^*$.

$(\tilde{\mathsf{Q}}^{(i)}, \tilde{\mathsf{y}}^{(i)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(i)})$
For $j = i-1, \ldots, 0$
  $\mathsf{f}_Q^{(j)} := \{\mathsf{SEnc}(\mathsf{pp}, (\mathsf{vk}_{\mathsf{v}_j}, \mathsf{id}_{j+1}^* \cdot \ell + \iota, b), \tilde{\mathsf{y}}_\iota^{(j+1)})\}_{\iota,b}$
  $(\tilde{\mathsf{Q}}^{(j)}, \tilde{\mathsf{y}}^{(j)}) := \mathsf{GCSim}(\mathsf{Q}, \mathsf{f}_Q^{(j)})$
$c^* := (\tilde{\mathsf{y}}^{(0)}, \tilde{\mathsf{Q}}^{(0)}, \ldots, \tilde{\mathsf{Q}}^{(n^*)}, \tilde{\mathsf{T}})$

and returns c^* to $\mathcal{A}$. $\mathcal{R}$ now continues the simulation of the $\mathcal{H}_{2i+1}$ experiment and outputs whatever the simulated $\mathcal{H}_{2i+1}$ experiment outputs.

We will now compute the advantage of $\mathcal{R}^{\mathcal{A}}$. First notice that the keys provided by $\mathcal{R}$ to $\mathcal{A}$ are distributed exactly as in $\mathcal{H}_{2i+1}$ (and therefore do not depend on i^*). If the challenge bit b^* of the $\mathsf{IND}^{\mathsf{OTSE}}$-experiment is 0, then the view of $\mathcal{A}$ in $\mathcal{R}$'s simulation is distributed exactly as in experiment $\mathcal{H}_{2i+1}$. On the other hand, if $b^* = 1$ then the view of $\mathcal{A}$ is distributed exactly as in experiment $\mathcal{H}_{2(i+1)}$. Thus we can conclude

$$\mathsf{Adv}(\mathcal{R}^{\mathcal{A}}) = |\Pr[\mathcal{H}_{2i+1}(\mathcal{A}) = 1] - \Pr[\mathcal{H}_{2(i+1)}(\mathcal{A}) = 1]| \geq \epsilon$$

Lemma 9. *We claim that the hybrids $\mathcal{H}_{2n}$ and $\mathcal{H}_{2n+1}$ are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme.*

The proof proceeds analogous to the proof of Lemma 7.

Lemma 10. *We claim that the hybrids $\mathcal{H}_{2n+1}$ and $\mathcal{H}_{2n+2}$ are computationally indistinguishable, given that the OTSE scheme* $(\mathsf{SSetup}, \mathsf{SGen}, \mathsf{SSign}, \mathsf{SEnc}, \mathsf{SDec})$ *is* $\mathsf{IND}^{\mathsf{OTSE}}$*-secure.*

The proof follows analogous to the proof of Lemma 8.

Lemma 11. *We claim that the hybrids $\mathcal{H}_{2n+2}$ and $\mathcal{H}_{2n+3}$ are computationally indistinguishable, given that* $(\mathsf{Garble}, \mathsf{Eval})$ *is a secure garbling scheme.*

Again, the proof follows analogous to the proof of Lemma 7.

Lemma 12. *The advantage of $\mathcal{A}$ in $\mathcal{H}_{2n+3}$ is negligible, given that* $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ *is* $\mathsf{IND}^{\mathsf{CPA}}$*-secure.*

Proof. We will construct an adversary $\mathcal{R}^{\mathcal{A}}$ that breaks the $\mathsf{IND}^{\mathsf{CPA}}$-security of $(\mathsf{KG}, \mathsf{E}, \mathsf{D})$ with advantage ϵ. $\mathcal{R}$ simulates $\mathcal{H}_{2n^*+3}$ faithfully, with the exception that it uses its own challenge public key pk^* as public key $\mathsf{lpk}_{\mathsf{id}^*}$ for the identity id^*, i.e. it sets $\mathsf{lpk}_{\mathsf{id}^*} := \mathsf{pk}^*$. It forwards $\mathcal{A}$'s challenge messages m_0 and m_1 to the $\mathsf{IND}^{\mathsf{CPA}}$-experiment and uses its own challenge ciphertext c^* as the ciphertext f in the computation of the challenge ciphertext c^*. It follow directly that $\mathcal{R}$ simulates $\mathcal{H}_{4n^*+3}$ perfectly and therefore $\mathsf{Adv}_{\mathsf{IND}^{\mathsf{CPA}}}(\mathcal{R}^{\mathcal{A}}) = \mathsf{Adv}_{\mathcal{H}_{4n^*+3}}(\mathcal{A})$.

This concludes the proof of Theorem 4.

References

AB09. Shweta Agrawal and Xavier Boyen. Identity-based encryption from lattices in the standard model. 2009.

ABB10a. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Efficient lattice (H)IBE in the standard model. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 553–572, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

ABB10b. Shweta Agrawal, Dan Boneh, and Xavier Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical IBE. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 98–115, Santa Barbara, CA, USA, August 15–19, 2010. Springer, Heidelberg, Germany.

ACD+06. Michel Abdalla, Dario Catalano, Alex Dent, John Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006: 33rd International Colloquium on Automata, Languages and Programming, Part II*, volume 4052 of *Lecture Notes in Computer Science*, pages 300–311, Venice, Italy, July 10–14, 2006. Springer, Heidelberg, Germany.

AFL12. Michel Abdalla, Dario Fiore, and Vadim Lyubashevsky. From selective to full security: Semi-generic transformations in the standard model. In Marc Fischlin, Johannes Buchmann, and Mark Manulis, editors, *PKC 2012: 15th International Conference on Theory and Practice of Public Key Cryptography*, volume 7293 of *Lecture Notes in Computer Science*, pages 316–333, Darmstadt, Germany, May 21–23, 2012. Springer, Heidelberg, Germany.

AS15. Gilad Asharov and Gil Segev. Limits on the power of indistinguishability obfuscation and functional encryption. In Venkatesan Guruswami, editor, *56th Annual Symposium on Foundations of Computer Science*, pages 191–209, Berkeley, CA, USA, October 17–20, 2015. IEEE Computer Society Press.

BB04a. Dan Boneh and Xavier Boyen. Efficient selective-ID secure identity based encryption without random oracles. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 223–238, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

BB04b. Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 443–459, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

BBG05. Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size ciphertext. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 440–456, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

BF01. Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

BHR12. Mihir Bellare, Viet Tung Hoang, and Phillip Rogaway. Foundations of garbled circuits. In Ting Yu, George Danezis, and Virgil D. Gligor, editors, *ACM CCS 12: 19th Conference on Computer and Communications Security*, pages 784–796, Raleigh, NC, USA, October 16–18, 2012. ACM Press.

BR93. Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In V. Ashby, editor, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

CDG+17. Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, pages 33–65, 2017.

CHK04. Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

CHKP10. David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. In Henri Gilbert, editor, *Advances in Cryp-*

tology – EUROCRYPT 2010, volume 6110 of *Lecture Notes in Computer Science*, pages 523–552, French Riviera, May 30 – June 3, 2010. Springer, Heidelberg, Germany.

Coc01. Clifford Cocks. An identity based encryption scheme based on quadratic residues. In Bahram Honary, editor, *8th IMA International Conference on Cryptography and Coding*, volume 2260 of *Lecture Notes in Computer Science*, pages 360–363, Cirencester, UK, December 17–19, 2001. Springer, Heidelberg, Germany.

DG17. Nico Döttling and Sanjam Garg. Identity-based encryption from the diffie-hellman assumption. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, pages 537–569, 2017.

DH76. Whitfield Diffie and Martin E. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, 22(6):644–654, 1976.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science*, pages 464–479, Singer Island, Florida, October 24–26, 1984. IEEE Computer Society Press.

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

GH09. Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *TCC 2009: 6th Theory of Cryptography Conference*, volume 5444 of *Lecture Notes in Computer Science*, pages 437–456. Springer, Heidelberg, Germany, March 15–17, 2009.

GHPT17. Philippe Gaborit, Adrien Hauteville, Duong Hieu Phan, and Jean-Pierre Tillich. Identity-based encryption from codes with rank metric. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part III*, volume 10403 of *Lecture Notes in Computer Science*, pages 194–224, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

GPV08. Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In Richard E. Ladner and Cynthia Dwork, editors, *40th Annual ACM Symposium on Theory of Computing*, pages 197–206, Victoria, British Columbia, Canada, May 17–20, 2008. ACM Press.

GS02. Craig Gentry and Alice Silverberg. Hierarchical ID-based cryptography. In Yuliang Zheng, editor, *Advances in Cryptology – ASIACRYPT 2002*, volume 2501 of *Lecture Notes in Computer Science*, pages 548–566, Queenstown, New Zealand, December 1–5, 2002. Springer, Heidelberg, Germany.

HL02. Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In Lars R. Knudsen, editor, *Advances in Cryptology – EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 466–481, Amsterdam, The Netherlands, April 28 – May 2, 2002. Springer, Heidelberg, Germany.

KR98. Hugo Krawczyk and Tal Rabin. Chameleon hashing and signatures. Cryptology ePrint Archive, Report 1998/010, 1998. http://eprint.iacr.org/1998/010.

Lam79. L. Lamport. Constructing digital signatures from a one-way function. Technical report, October 1979.

LP09. Yehuda Lindell and Benny Pinkas. A proof of security of Yao's protocol for two-party computation. *Journal of Cryptology*, 22(2):161–188, April 2009.

LW10. Allison B. Lewko and Brent Waters. New techniques for dual system encryption and fully secure HIBE with short ciphertexts. In Daniele Micciancio, editor, *TCC 2010: 7th Theory of Cryptography Conference*, volume 5978 of *Lecture Notes in Computer Science*, pages 455–479, Zurich, Switzerland, February 9–11, 2010. Springer, Heidelberg, Germany.

MM16. Mohammad Mahmoody and Ameer Mohammed. On the power of hierarchical identity-based encryption. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 243–272, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

NY89. Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *21st Annual ACM Symposium on Theory of Computing*, pages 33–43, Seattle, WA, USA, May 15–17, 1989. ACM Press.

Reg05. Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In Harold N. Gabow and Ronald Fagin, editors, *37th Annual ACM Symposium on Theory of Computing*, pages 84–93, Baltimore, MA, USA, May 22–24, 2005. ACM Press.

RSA78. Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. A method for obtaining digital signature and public-key cryptosystems. *Communications of the Association for Computing Machinery*, 21(2):120–126, 1978.

Sha84. Adi Shamir. Identity-based cryptosystems and signature schemes. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

SW08. Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In Luca Aceto, Ivan Damgård, Leslie Ann Goldberg, Magnús M. Halldórsson, Anna Ingólfsdóttir, and Igor Walukiewicz, editors, *ICALP 2008: 35th International Colloquium on Automata, Languages and Programming, Part II*, volume 5126 of *Lecture Notes in Computer Science*, pages 560–578, Reykjavik, Iceland, July 7–11, 2008. Springer, Heidelberg, Germany.

Wat05. Brent R. Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, volume 3494 of *Lecture Notes in Computer Science*, pages 114–127, Aarhus, Denmark, May 22–26, 2005. Springer, Heidelberg, Germany.

Yao82. Andrew Chi-Chih Yao. Protocols for secure computations (extended abstract). In *23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, Chicago, Illinois, November 3–5, 1982. IEEE Computer Society Press.