Adaptively Secure Hierarchical Attribute-Based Encryption from Witness Encryption

Rishab Goyal
UW-Madison*

Saikumar Yadugiri
UW-Madison†

Abstract

Hierarchical attribute-based encryption (HABE), also called delegatable ABE, augments ABE with a public delegation algorithm that lets any user holding a secret key for a predicate f locally derive a key for a more restrictive predicate $f \wedge g$. Since keys are no longer produced only by the master authority, an attacker may adaptively corrupt keys generated by honest users, which makes security far more delicate than for plain ABE. For general predicates, HABE was previously known in the standard model only with selective security from lattices, where the number of delegations had to be bounded a-priori and secret key size grew quadratically with it, or from obfuscation-flavored assumptions. Beyond identity-based predicates no construction achieved adaptive security without complexity leveraging. We revisit HABE through the lens of witness encryption (WE) and achieve the following.

- A **selectively-secure** HABE scheme for all polynomial-size predicates from witness encryption, statistically-sound NIZKs, and statistically-binding commitments, supporting an *unbounded* number of key delegations with secret key size growing only *linearly* with each delegation.
- An **adaptively-secure** HABE scheme for the same class, again supporting an *unbounded* number of key delegations, assuming in addition equivocal commitments and a mixed hierarchical functional encryption scheme, a new primitive that we introduce.

Our constructions are in the standard model, avoid random oracles, complexity leveraging, and reduce black box to the polynomial hardness of the underlying primitives. On the technical front, we extend the witness encryption based ABE template of [GGSW13] to support public delegation. Encoding a delegated key as a proof that verifies its parent's proof makes proof size grow exponentially with the depth of the hierarchy, capping delegations at $O(1)$. So, we instead *chain* proofs rather than compose them. Adaptive security then calls for a genuinely hierarchical form of the dual-systems methodology, which is what mixed hierarchical functional encryption abstracts.

*Email: rishab@cs.wisc.edu.
†Email: saikumar@cs.wisc.edu.

Contents

1 Introduction **1**

2 Technical Overview **4**
2.1 Going Beyond $O(1)$ Delegations 6
2.2 Adaptive Security 8

3 Preliminaries **11**
3.1 Statistically Binding Commitments 11
3.2 Non-Interactive Zero-Knowledge Proofs 12
3.3 Witness Encryption 13

4 Hierarchical Attribute-Based Encryption **13**

5 Selectively Secure HABE from Witness Encryption **15**
5.1 Security Analysis 17

6 Mixed Hierarchical Functional Encryption **25**

7 Adaptively Secure HABE from Witness Encryption and Mixed Hierarchical Functional Encryption **27**
7.1 Preliminary: Equivocal Commitments 27
7.2 Construction 28
7.3 Security Analysis 30

1 Introduction

Attribute-Based Encryption (ABE) [SW05, GPSW06] is a powerful generalization of public-key encryption. It enables fine-grained access control over encrypted data, by allowing senders to specify a policy at the time of encryption (or attaching attributes to ciphertexts) and tying decryption capabilities of users to a matching credential. Beyond access-control applications [PRV12, GKP+13, GM15], ABE has become a versatile cryptographic building block. It has been instrumental in the design of reusable garbled circuits [GKP+13], traitor-tracing systems [GKW18, CVW+18, GKW19], broadcast encryption [AY20, AWY20, Wee25], succinct witness encryption for P [AMR25], and a host of other primitives. Over two decades, the community has come up with many diverse ABE designs from a wide range of cryptographic assumptions [GPSW06, BSW07, KSW08, Wat09, LOS+10, LW10, Wat11, GVW13, GGSW13, BGG+14, CGW15, Tsa19, KW20, KNYY20, GLW21, Wee22, CW23, HLL23, WW24, AKY24, CW24, Wee24, Wee25].

In a (key-policy) ABE scheme, a central authority sets up a master public-secret key pair $(\mathsf{PP}, \mathsf{MSK})$. Anyone can encrypt a message μ under an attribute x, producing a ciphertext CT_x. The authority issues a secret key SK_f to each user for their policy f, and SK_f decrypts CT_x if and only if $f(x) = 1$. The most desirable security goal is *adaptive security*, where an adversary may interleave secret-key queries with the challenge query, picking the challenge attribute only after receiving system parameters and secret keys. In contrast, a weaker but more tractable notion is *selective security*, in which the adversary must commit to the challenge ahead of time. Selective security is largely a technical convenience for deriving a formal security reduction, and the gap between the two has historically been quite large to close, especially when working outside the realm of bilinear pairings.

Hierarchical/Delegatable ABE. A hierarchical ABE (also called delegatable ABE, henceforth HABE) [GPSW06, GVW13, BGG+14, BCG+17] generalizes ABE by adding a public delegation algorithm $\mathsf{Delegate}$ that lets any user holding a secret key SK_f for a policy f locally derive a new secret key $\mathsf{SK}_{f \wedge g}$ for a more restrictive policy $f \wedge g$, without contacting the master authority. This mirrors organizational hierarchies and reduces the burden on the central authority, since it no longer has to issue a key to every receiver. For instance, a department chair holding a secret key for '*dept:CS*' can locally derive keys for the instructors of individual courses, and an instructor can in turn derive further restricted keys for the teaching assistants of their own course. The most popular special case of HABE is hierarchical identity-based encryption (HIBE) [HL02, GS02], where the policy class is prefix matching.

There is a long line of HABE designs, but they all sit in one of three regimes. For HIBE-style prefix policies, constructions are known from pairings, lattices, and a variety of other assumptions [HL02, GS02, GH09, LW10, ABB10a, ABB10b, CHKP12, DG17b, DG17a, BLSV18, GKR25]. For HABE beyond prefix policies, that is supporting general circuits as policies, the only known constructions in the standard model are those of Gorbunov-Vaikuntanathan-Wee [GVW13] and Boneh et al. [BGG+14] from LWE, and both are selectively secure. From obfuscation-flavored assumptions, hierarchical functional encryption and HABE for general circuits can be designed via [BCG+17].

Why adaptive security for HABE is hard. Because of the hierarchical nature of HABE, security has to be defined with care. Unlike vanilla ABE, where every secret key is sampled by the master authority, secret keys in HABE can be computed by any honest user. An attacker can therefore corrupt not only the keys issued by the master authority, but also the keys that honest

users produced as part of delegation. This considerably widens the scope of the attacker and makes both defining and proving adaptive security significantly more challenging than for vanilla ABE.

This gap was first pointed out by Shi and Waters [SW08], who studied the definitional issues in the context of HIBE. Informally, they observed that the right way to capture general adversaries in delegatable systems is to let the attacker initialize an arbitrary number of honest users and adaptively decide which of them to corrupt, where initialization, delegation, and corruption may be arbitrarily interleaved. As a concrete example, any reasonable definition should capture the following attacker. The attacker asks the challenger to initialize a key SK_{f_1}, then asks it to delegate that key to $\mathsf{SK}_{f_1 \wedge f_2}$, and later to $\mathsf{SK}_{f_1 \wedge f_2 \wedge f_3}$ or to $\mathsf{SK}_{f_1 \wedge f_4}$, and meanwhile it may ask to corrupt any subset of the keys initialized so far. We carry this style of definition over to HABE.

While a real-world adversary should certainly be allowed to behave this way, it may not be obvious why this makes adaptive security so much harder to prove. Briefly, the reason is that in vanilla ABE an attacker never initializes user keys, it only corrupts them directly. Hence, whenever the challenger has to create a secret key for a policy f in the pre-challenge phase, it knows for a fact that $f(x^*) = 0$ for the eventual challenge attribute x^*. This guarantee is simply not available in HABE. The attacker may ask the challenger to initialize SK_f for an arbitrary policy f, ask it to delegate on top of SK_f, and even corrupt some of those delegated keys, while the challenger still has no idea whether $f(x^*) = 0$ or $f(x^*) = 1$. In vanilla ABE a reduction can always set up the system parameters in a trapdoor way that lets it answer every secret key query. In HABE, precisely because of the extra power of asking for delegated keys derived from a distinguishing key, it is very difficult to generalize the usual ABE proof techniques. For the special case of HIBE, Lewko and Waters [LW14] additionally showed that adaptive security cannot be proven via black-box reductions for schemes with certain checkability properties, and since HABE is far more expressive than HIBE, adaptive security for HABE faces even stronger barriers.

To the best of our knowledge, beyond identity-based policies, there is no construction of HABE in any model that achieves adaptive security without relying on 'complexity leveraging', with its associated sub-exponential security loss and stronger assumptions. This raises the natural question.

Can we design adaptively-secure HABE for general policies without complexity leveraging?

This Work. We address this question under the assumption of *witness encryption* (WE) [GGSW13]. Unlike indistinguishability obfuscation, witness encryption is a fundamentally and significantly weaker primitive than iO, and there is a growing body of work designing advanced cryptographic primitives from WE that we do not know how to build from iO-free assumptions otherwise [GLW14, GVW19, FWW23, WW24]. Moreover, the candidate-construction landscape for WE is richer than for iO as there are obfuscation-free candidates from pairing-free groups [BIOW20], lattice-based candidates from evasive LWE [Tsa22, VWW22], and other recent proposals. Concretely, we show the following.

Theorem (Informal) 1 (HABE from WE)**.** Assuming witness encryption, statistically-sound NIZKs, and statistically-binding commitments, there exists a **selectively-secure** HABE scheme for all polynomial-size policies, supporting an unbounded number of key delegations. Assuming in addition equivocal commitments and a mixed hierarchical functional encryption scheme, there exists an **adaptively-secure** HABE scheme for all polynomial-size policies, again supporting an unbounded number of key delegations.

Both of our schemes are in the standard model, do not rely on Random Oracle [BR93] heuristics or complexity leveraging, and provide black-box reductions to the polynomial hardness of the

underlying primitives. Both support an *unbounded* number of key delegations. Neither Setup nor KGen nor Delegate takes any bound on the depth of the hierarchy, so a key may be delegated as many times as its holders wish. In the selectively secure scheme the secret key size grows only linearly with each delegation operation. We do not claim the same for the adaptively secure scheme.

The one place a bound appears is at encryption time. The encryptor specifies the maximum hierarchy depth of the secret keys that may decrypt the resulting ciphertext, and both ciphertext size and encryption time grow with that choice. This is not a limitation of our design but of witness encryption itself, which we only know how to build for circuit languages rather than uniform ones, so the size of a witness has to be fixed in advance. Designing HABE where a single ciphertext is decryptable by keys of unbounded depth is an interesting open problem, and it would follow immediately from witness encryption for uniform models of computation.

It is worth comparing this to what was previously known. Beyond HIBE, the only prior work that could support an unbounded number of delegations was that of Brakerski et al. [BCG+17], which required collusion-resistant general-purpose functional encryption [BSW11] and proved only selective security. The scheme of Boneh et al. [BGG+14] supported an a-priori bounded number of delegations, fixed once and for all at setup, and its secret key size grew quadratically with that bound. Ours therefore improves on the former in the assumption it needs and, in the adaptive case, in the security it achieves, and on the latter in the delegation guarantee, and in the selectively secure case in the growth of the key as well.

Mixed hierarchical functional encryption is a new primitive that we introduce in this work. It augments mixed functional encryption with a public delegation algorithm on ciphertexts, and it is what allows the dual-systems methodology to be carried down a delegation chain. We give a full definition of it, and we regard instantiating it as the main problem left open by our work.

Our techniques. Our starting point is the witness-encryption-based ABE template of Garg-Gentry-Sahai-Waters [GGSW13], and our high-level approach to adaptivity follows the recent work of Waters and Wichs [WW24], who generalized the classic dual-systems methodology of Waters [Wat09] beyond bilinear pairings. Extending either template to a hierarchy is not immediate, and we develop two new ideas.

The first idea concerns efficiency. The natural way to encode a delegated key in the GGSW template is as a NIZK proof that verifies the proof carried by its parent. Composing proofs in this way makes the proof size grow with the depth of the hierarchy, so that after d levels a proof has size $\mathsf{poly}(\lambda)^d$, which caps the number of delegations at $O(1)$. We instead *chain* proofs rather than compose them. Each secret key carries a sequence of fresh commitments, and the proof at each level speaks only about the openings of two consecutive commitments rather than about the previous proof. Proof size then becomes independent of the depth, and in our selectively secure scheme the secret key grows only *linearly* with the number of delegations.

The second idea concerns adaptivity, and it is where the hierarchy greatly departs from the flat case. The dual-systems based approach of [WW24] attaches to every key a mixed functional encryption ciphertext, which can be generated in a public mode or in a secret mode, and the analysis moves from one to the other. This breaks down under delegation. We therefore introduce mixed hierarchical functional encryption, in which the secret mode is injected once at the root of a delegation chain and then transported downwards by a public delegation operation on ciphertexts, so that a ciphertext deep in a chain still testifies to the entire chain above it.

Concurrent Work. In an independent work, Goyal-Koppula-Rajasree [GKR25] resolved adaptive security of HIBE, restricting to the much less expressive class of prefix policies, under the much

weaker assumption of selectively-secure IBE. Their techniques are largely orthogonal to ours. They exploit the structural specialness of prefix policies over a selective IBE template, while we design *general-circuit* HABE from witness encryption.

Related Work. Since ABE was proposed in [SW05, GPSW06], it has grasped significant attention from the community and has been a versatile tool in constructing encrypted access-control systems [PRV12, GKP+13]. Numerous works have constructed ABE with varying levels of security and functionality from a range of standard and post-quantum assumptions, a non-exhaustive list being [LOS+10, GVW13, BGG+14, Tsa19, Wee22, LLL22, HLL23, AKY24, HLL24, Wee24, Wee25].

HABE is a hierarchical variant of ABE and has seen limited success through the decades, with [GVW13, BGG+14] constructing selectively secure variants. A stronger version, hierarchical functional encryption, is nearly equivalent to functional encryption [BCG+17, BS15], and a weaker version, hierarchical IBE, is known to be equivalent to selective IBE [DG17b, DG17a, GKR25].

Adaptive security has been a challenging problem for ABE, with only a few known approaches. These include those based on general-purpose functional encryption [GGH+13, Wat15, ABSV15, AJS15, AJ15, BV15], the pairing-based dual-systems methodology [Wat09, LOS+10], and subset functionalities [Tsa19, GLW21]. Recently, Waters and Wichs [WW24] constructed adaptively secure ABE for any polynomial-size policy class from witness encryption, which is weaker than functional encryption and obfuscation. Our work develops new techniques that lift the [WW24] template through a delegation hierarchy.

2 Technical Overview

We start by recalling the notion of HABE [HL02, GS02, GVW13, BGG+14] as an extension of (key-policy) ABE.

Hierarchical ABE. An ABE scheme consists of four algorithms, $\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}$. Setup generates public parameters PP and a master secret key MSK. KGen uses MSK and the description of a policy f to privately create a secret key SK_f. Enc encrypts a message μ under an attribute x to create a ciphertext CT_x, and Dec using SK_f and CT_x outputs μ if and only if $f(x) = 1$.

A hierarchical ABE (HABE) scheme adds a public delegation facility on top of the private one provided by KGen. This is done using a $\mathsf{Delegate}$ algorithm that uses a secret key for a policy f and the description of a policy g to generate $\mathsf{SK}_{f \wedge g}$. Note that MSK is not used in this process, so any user holding a key may delegate on top of it. Naturally, Dec now reveals μ if and only if for $f = f_1 \wedge \ldots \wedge f_\ell$, we have $f(x) = 1$, that is $f_1(x) = 1 \wedge \ldots \wedge f_\ell(x) = 1$.

Adaptive security for HABE. The idea behind adaptive security for HABE is to protect ciphertexts from active attackers who can create and delegate secret keys for arbitrary policies but can only corrupt keys that do not satisfy the challenge attribute. To formulate this, we consider a game between a challenger and an attacker where the attacker asks the challenger to generate and Store a secret key for a policy f. The attacker receives a handle h from the challenger in response. The attacker can use h and a policy g to $\mathsf{Delegate}$ the key SK_f with g to create $\mathsf{SK}_{f \wedge g}$, and it receives a new handle to access this secret key. The attacker can also $\mathsf{Corrupt}$ the secret key for any policy p using the corresponding handle, provided $p(x^*) = 0$, and it receives SK_p. In addition, the attacker makes a challenge query with (x^*, μ_0, μ_1) adaptively, that is at any point in the game. Note that there is no admissibility criterion on the policies sent to Store and $\mathsf{Delegate}$. They could

very well be satisfying, that is $f(x^*) = 1$. The challenger encrypts (x^*, μ_b) for a randomly chosen b and the attacker wins if it can guess b with non-negligible probability. This behaviour of the attacker, sending x^* after making any of the previous queries, is what makes adaptive security hard to achieve. In the weaker *selective* notion, the attacker declares x^* before it even receives PP.

Witness encryption (WE). The major tool we use in our constructions is WE [GGSW13] (GGSW henceforth). A WE scheme for an NP language $\mathcal{L}$ can be thought of as a worst-case public-key encryption where the relation between the public and the secret key is determined by an NP relation R. In particular, in a WE scheme we encrypt a message μ using an instance $\mathtt{inst}$ that may or may not be in $\mathcal{L}$. The resulting ciphertext can only be decrypted using a witness $\mathtt{wit}$ such that $\mathsf{R}(\mathtt{inst}, \mathtt{wit}) = 1$, that is if and only if $\mathtt{wit}$ is a valid witness for the membership of $\mathtt{inst}$ in $\mathcal{L}$. Semantic security of WE can be argued as long as $\mathtt{inst} \notin \mathcal{L}$.

Reviewing GGSW. The public and master secret key of their system are the verification and signing key of a special dual-mode constrained signature scheme, which is instantiated using perfectly binding commitments and perfectly sound NIZKs[1]. Setup samples two commitments of zero, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$ and $\mathsf{com}' = \mathsf{Com}(0^n; r^{(1)})$, sets $\mathsf{PP} := (\mathsf{com}_0, \mathsf{com}')$ and $\mathsf{MSK} := r^{(0)}$. A secret key for a policy f is a signature σ_f, which concretely is a NIZK proof π_f for the instance (PP, f) in the language

$$\mathcal{L} = \left\{ (\mathsf{PP}, f) \ : \begin{array}{c} \mathtt{wit} := r^{(0)} \text{ such that } \mathsf{com}_0 = \mathsf{Com}(0; r^{(0)}) \\ (\text{OR}) \\ \mathtt{wit} := (r^{(1)}, x^*) \text{ such that } \mathsf{com}' = \mathsf{Com}(x^*; r^{(1)}) \wedge f(x^*) = 0 \end{array} \right\}$$

To encrypt (x, μ), GGSW creates a WE ciphertext for the language that asks for a pair (g, π_g) such that π_g verifies with respect to $\mathcal{L}$ and $g(x) = 1$. Correctness is immediate. The first branch of $\mathcal{L}$, which is the normal mode (indicated in green above) is satisfied using MSK, so (f, π_f) is a valid WE witness whenever $f(x) = 1$, and the WE ciphertext reveals μ.

The selective security argument is a sequence of three steps. An attacker that commits to x^* up front lets the challenger gradually move to a trapdoor mode (indicated in red above). (1) Program $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$, (2) for every secret key query, which is necessarily for a policy with $f(x^*) = 0$, generate π_f using the trapdoor witness $(r^{(1)}, x^*)$ rather than $r^{(0)}$, (3) program $\mathsf{com}_0 = \mathsf{Com}(1; r^{(0)})$. At this point, consider any candidate WE witness (g, π_g). Since com_0 perfectly binds to 1, the first branch of $\mathcal{L}$ cannot be satisfied, so perfect soundness of the NIZK tells us that π_g must have been produced in the trapdoor mode, and since com' perfectly binds to x^*, this certifies $g(x^*) = 0$. But the WE language also demands $g(x^*) = 1$. Here, we reach a contradiction as we require $g(x^*) = 0$ and 1 simultaneously, and thus the WE instance is unsatisfiable. Semantic security of WE finishes the argument.

Delegation via metaproofs. A natural way to add delegation to this template is to let a delegated key be a proof that verifies its parent's proof. The master authority issues $\mathsf{SK}_{f_1} = (f_1, \pi_1)$ as in GGSW, where π_1 is a proof for a language $\mathcal{L}_1$. The public delegation algorithm then uses (f_1, π_1) as a witness to a new language $\mathcal{L}_2$ asserting that π_1 verifies with respect to $\mathcal{L}_1$, which yields $\mathsf{SK}_{f_1 \wedge f_2} = (f_1, f_2, \pi_2)$. More generally, after d delegations we work with d languages $\mathcal{L}_1, \ldots, \mathcal{L}_d$, where $\mathcal{L}_i$ in the normal mode checks the previous level proof and in the trapdoor mode checks the GGSW trapdoor condition for $f_1 \wedge \ldots \wedge f_i$. The WE language is adjusted to ask that for some

[1] Statistically binding commitments and statistically sound NIZKs suffice, and that is what we use in the main body. Perfect binding and perfect soundness make our arguments easier to follow.

$\ell \leq d$, the proof π_ℓ verifies with respect to $\mathcal{L}_\ell$ and $f_1 \wedge \ldots \wedge f_\ell(x) = 1$. Selective security follows from the GGSW argument augmented with a recursive simulation step, where the challenger delays proof generation until a $\mathsf{Corrupt}$ query is made and then simulates each proof in turn.

The fundamental limitation of this approach is its inefficiency under composition. If the size of a NIZK proof is $|\pi| = \mathsf{poly}(\lambda, |\mathtt{inst}|, |\mathtt{wit}|)$, then the running time of $\mathsf{NIZK.Verify}$ is at least $\mathsf{poly}(\lambda, |\mathtt{inst}|, |\mathtt{wit}|)$. When we place another proof on top of it, the size and the running time of the prover grow proportionally to $|\pi|$. Hence, after d layers of delegation the size of a proof is $\mathsf{poly}(\lambda)^d$, and we cannot delegate more than $O(1)$ many times with this strategy. Rate-1 NIZKs could help, but the known constructions [GGI+15, ACG+24, BDS25] either lack statistical soundness or have prover time growing with the large instance, neither of which suffices here.

2.1 Going Beyond $O(1)$ Delegations

The core issue with the metaproof strategy is that we compose proofs. We did so to make sure that there is a connection between the key generated by KGen and any key delegated from it. Our idea is to instead *chain* proofs, so that a new proof is generated from the components of the previous key rather than from the previous proof itself.

Concretely, we let every secret key carry a fresh commitment of zero that the delegation algorithm can use to generate the next proof. That is, a key at depth i carries $\mathsf{com}_i = \mathsf{Com}(0; r_i^{(0)})$ along with the randomness $r_i^{(0)}$, and the delegation algorithm treats the pair $(\mathsf{com}_i, r_i^{(0)})$ as the *new master secret key* with which it delegates. Delegation samples a fresh $\mathsf{com}_{i+1} = \mathsf{Com}(0; r_{i+1}^{(0)})$ and proves a statement about the pair $(\mathsf{com}_i, \mathsf{com}_{i+1})$ alone, never about π_i. A secret key for $f_1 \wedge \ldots \wedge f_\ell$ therefore looks like

$$\mathsf{SK}_{f_1 \wedge \ldots \wedge f_\ell} = \left(f_1, \pi_1, \mathsf{com}_1, \ \ldots \ , f_\ell, \pi_\ell, \mathsf{com}_\ell, \underline{r_\ell^{(0)}}\right)$$

where the underlined component is the only part that does not survive delegation. The randomness $r_\ell^{(0)}$ is exactly what lets its holder delegate once more, and it is discarded the moment it is used, so a delegated key can extend the chain but can never speak for a shorter prefix of it.

For technical reasons, we use two trapdoor modes rather than one. The first is the GGSW trapdoor mode, which certifies $f_\ell(x^*) = 0$. The second certifies that the chain is not corruptible yet, and it is expressed by having both commitments of the level be commitments of 1. Writing com_0 for the commitment sampled by Setup and $r_0^{(0)}$ for MSK, so that every level speaks about a consecutive pair, the language is

$$\mathcal{L} = \left\{ \left(\begin{array}{c} \mathsf{PP}, \\ (f_i, \pi_i, \mathsf{com}_i)_{i<\ell}, f_\ell, \mathsf{com}_\ell \end{array} \right) : \begin{array}{c} \mathtt{wit} := (r_{\ell-1}^{(0)}, r_\ell^{(0)}) \text{ such that} \\ \mathsf{com}_{\ell-1} = \mathsf{Com}(0; r_{\ell-1}^{(0)}) \wedge \mathsf{com}_\ell = \mathsf{Com}(0; r_\ell^{(0)}) \\ (\text{OR}) \\ \mathtt{wit} := (r_{\ell-1}^{(0)}, r_\ell^{(0)}, r^{(1)}, x^*) \text{ such that} \\ \mathsf{com}_{\ell-1} = \mathsf{Com}(1; r_{\ell-1}^{(0)}) \wedge \mathsf{com}_\ell = \mathsf{Com}(0; r_\ell^{(0)}) \wedge \\ \mathsf{com}' = \mathsf{Com}(x^*; r^{(1)}) \wedge f_\ell(x^*) = 0 \\ (\text{OR}) \\ \mathtt{wit} := (r_{\ell-1}^{(0)}, r_\ell^{(0)}) \text{ such that} \\ \mathsf{com}_{\ell-1} = \mathsf{Com}(1; r_{\ell-1}^{(0)}) \wedge \mathsf{com}_\ell = \mathsf{Com}(1; r_\ell^{(0)}) \end{array} \right\}$$

Reading the three branches as transitions on the values committed by $(\mathsf{com}_{\ell-1}, \mathsf{com}_\ell)$, the language

permits $0 \to 0$, permits $1 \to 0$ only when $f_\ell(x^*) = 0$, and permits $1 \to 1$ freely. It never permits $0 \to 1$, and this is what drives the entire security argument.

The WE language changes a little more than one might expect, and the change matters. A witness is a full chain $(f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}$ together with the randomness $r_\ell^{(0)}$, and the relation checks three things. First, that every proof π_j in the chain verifies, and not merely the last one. Second, that $f_1(x) \wedge \ldots \wedge f_\ell(x) = 1$. Third, that $\mathsf{com}_\ell = \mathsf{Com}(0; r_\ell^{(0)})$. The third check is what makes the hidden component of the key meaningful. Without it, an attacker holding a key for $f_1 \wedge f_2$ where $f_1(x^*) = 1$ and $f_2(x^*) = 0$ could simply drop the last link and present the prefix $(f_1, \pi_1, \mathsf{com}_1)$ as a witness, which would satisfy the first two checks. Requiring an opening of the last commitment in the chain blocks this, because $r_1^{(0)}$ was discarded when the key was delegated.

Unbounded delegations. Since a proof at level ℓ speaks only about openings of two commitments and about f_ℓ, its witness has size $\mathsf{poly}(\lambda, n)$ and its size is independent of ℓ. Nothing in Setup, KGen, or $\mathsf{Delegate}$ takes a bound on the depth of the hierarchy, so a key may be delegated an unbounded number of times. Using perfectly sound NIZKs for unbounded size instances, which follow from fully homomorphic encryption [GGI+15, Gen09, BV11, GSW13], a secret key for a chain of length ℓ has size $\ell \cdot \mathsf{poly}(\lambda, n, s)$, so it grows only *linearly* with each delegation. This compares favourably with the scheme of Boneh et al. [BGG+14], which supports only an a-priori bounded number of delegations and whose keys grow quadratically with that bound.

The one place a bound does appear is in the WE language, whose witness is a chain and therefore has size proportional to the depth of delegation. Since we only know how to build witness encryption for circuit languages rather than for uniform models of computation, the size of a witness must be fixed before encrypting. Hence the encryptor declares a bound d, and the resulting ciphertext is decryptable by any key that was delegated at most d times.

Selective security. Our analysis relies on the fact that x^* is available before Setup is run. The challenger therefore knows, for every chain the attacker will ever build, exactly the depth at which a key becomes corruptible. It uses this to decide what each commitment holds.

It sets $\mathsf{com}_0 = \mathsf{Com}(1)$. For a key whose chain satisfies x^*, that is $f_1(x^*) \wedge \ldots \wedge f_i(x^*) = 1$, the attacker is not allowed to corrupt it, and the challenger sets $\mathsf{com}_i = \mathsf{Com}(1)$. For the first policy in the chain with $f_i(x^*) = 0$, the key may have to be handed over, and the challenger sets $\mathsf{com}_i = \mathsf{Com}(0)$, as it does for every commitment below that point. The two trapdoor modes now line up exactly with this assignment. The links above the descent are $1 \to 1$ and are covered by the second trapdoor mode, the unique link at the descent is $1 \to 0$ and is covered by the first trapdoor mode, which certifies $f_i(x^*) = 0$ just as in GGSW, and the links below are $0 \to 0$ and are covered by the normal mode. The transition $0 \to 1$ never arises, because once a prefix fails to satisfy x^* no extension of it can satisfy x^* either. To move from the real experiment to this one, the challenger first simulates every proof, then switches all the commitments at once, and then generates the proofs honestly again using whichever branch matches the pair at that level.

It remains to see that the WE instance is unsatisfiable. Consider any candidate witness, a chain $(f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}$ together with $r_\ell^{(0)}$, and read off the value β_i that each com_i perfectly binds to. We know $\beta_0 = 1$ because Setup committed 1, and we know $\beta_\ell = 0$ because the third check of the WE language exhibits an opening of com_ℓ to 0. Since the language never permits $0 \to 1$, the sequence $\beta_0, \ldots, \beta_\ell$ is non-increasing, so there must exist a point i^* in the chain where the previous commitment is a commitment of 1 and the current one is a commitment of 0. By perfect soundness, the proof at that level was produced in the first trapdoor mode, which certifies $f_{i^*}(x^*) = 0$. But

the second check of the WE language demands $f_1(x^*) \wedge \ldots \wedge f_\ell(x^*) = 1$, which gives $f_{i^*}(x^*) = 1$. Here, we reach a contradiction as we require $f_{i^*}(x^*) = 0$ and 1 simultaneously, and thus the WE instance is unsatisfiable. Semantic security of WE now applies, and we obtain a selectively secure HABE for all polynomial-size policies.

2.2 Adaptive Security

Everything before rests on the challenger knowing x^* before it sets up the system. Making the scheme adaptive requires additional technical tools. To this end we look at the work of Waters-Wichs [WW24], which relied on a special type of functional encryption system called Mixed-FE[2] [GKW18, CVW+18] that plays an important part in executing the dual-systems paradigm [Wat09] for achieving adaptive security.

Reviewing Mixed-FE. A mixed functional encryption scheme [GKW18] is a bounded-collusion secure[3] secret-key FE scheme, which we know how to construct from one-way functions [SS10, GVW12, AV19]. It has one interesting feature that makes it non-trivial. Ciphertexts for a policy f can be generated in two modes, a secret mode using $\mathsf{skEnc}(\mathsf{msk}, f)$ and a public mode using $\mathsf{pkEnc}(f)$. The pair $(\mathsf{skEnc}, \mathsf{KGen})$ works like FE and reveals $f(x)$, whereas $(\mathsf{pkEnc}, \mathsf{KGen})$ always outputs 0. Security is an indistinguishability game where an attacker makes unbounded queries for ciphertexts of policies $\{f_i\}_i$ and a single secret key for an input x adaptively[4], and it cannot distinguish the mode in which the ciphertexts were generated as long as $f_i(x) = 0$ for every i.

Adaptive ABE via Mixed-FE. At a high level, [WW24] uses $(\mathsf{KGen}, \mathsf{pkEnc})$ as the functional system and $(\mathsf{KGen}, \mathsf{skEnc})$ as the semi-functional one, and the switch between them is unnoticeable by Mixed-FE security. In a bit more detail, along with π_f the key gives out a Mixed-FE ciphertext $\mathsf{mfe.ct}_f \leftarrow \mathsf{pkEnc}(f)$. Along with the WE ciphertext, encryption samples a fresh $\mathsf{mfe.msk}$ and gives out $\mathsf{mfe.sk}_x \leftarrow \mathsf{KGen}(\mathsf{mfe.msk}, x)$. The normal mode of the NIZK language is unchanged, but in the trapdoor mode we check that com' is a commitment of $\mathsf{mhfe.msk}$ and that $\mathsf{mfe.ct}_f$ was generated in the secret mode, that is $\exists\ r$ such that $\mathsf{mfe.ct}_f = \mathsf{skEnc}(\mathsf{mfe.msk}, f; r)$. The WE language checks that the NIZK proof verifies, that $\mathsf{Dec}(\mathsf{mfe.sk}_x, \mathsf{mfe.ct}_f) = 0$, and that $f(x) = 1$. Since $\mathsf{mfe.ct}_f$ is generated in the public mode, the WE language is satisfied whenever $f(x) = 1$, and correctness holds.

To argue adaptive security, one first switches $\mathsf{mfe.ct}_f$ to $\mathsf{skEnc}(\mathsf{mfe.msk}, f)$ for every key query, which is unnoticeable by Mixed-FE security because $\mathsf{mfe.msk}$ appears nowhere in the NIZK language yet. From here one proceeds as in GGSW, setting com' to be a commitment of $\mathsf{mfe.msk}$, generating proofs in the trapdoor mode, and setting com_0 to be a commitment of 1. Now any satisfying WE witness must have $f(x) = 1$ and $\mathsf{Dec}(\mathsf{mfe.sk}_x, \mathsf{mfe.ct}_f) = 0$. But since com_0 perfectly binds to 1 and com' perfectly binds to $\mathsf{mfe.msk}$, the ciphertext $\mathsf{mfe.ct}_f$ must have been generated in the secret

[2] We point that [WW24] introduced a new notion that they refer to as functional tags, constructed them from one-way functions, and used functional tags instead of Mixed-FE. However, we observe that functional tags are merely an alternate approach to define a 1-query bounded version of Mixed-FE [GKW18]. We find it a bit easier to explain our approach using the Mixed-FE framework. Moreover, we believe that viewing this abstract dual-systems technique from the lens of Mixed-FE could be more meaningful for future work in adaptive security

[3] By bounded-collusion, we mean that security is only going to hold against attackers that corrupt an a-priori bounded number of secret keys.

[4] Note that [GKW18, CVW+18] proposed and constructed the "dual" notion of our Mixed-FE where indistinguishability holds against an attacker that possesses unbounded number of secret keys and a-priori bounded (not necessarily one) ciphertexts. In addition, their ciphertexts generated by pkEnc always output 1.

mode, and therefore Dec reveals $f(x)$. Here, we reach a contradiction as we require $f(x) = 0$ and 1 simultaneously, and thus the WE instance is unsatisfiable.

Adaptive HABE via Mixed-FE? It stands to reason that using Mixed-FE in the same way we could realize adaptive security for our chained construction readily. However, this argument is subtle and not so straightforward. Consider the following erroneous way of doing it. In KGen and in $\mathsf{Delegate}$ we generate $\mathsf{mfe.ct}_f \leftarrow \mathsf{pkEnc}(f)$ and give it out along with the secret key. In Enc we sample a fresh $\mathsf{mfe.msk}$, generate $\mathsf{mfe.sk}_x$, and give it out along with the WE ciphertext. The WE language checks that the proofs verify, that $\mathsf{Dec}(\mathsf{mfe.sk}_x, \mathsf{mfe.ct}) = 0$, and that $f_1 \wedge \ldots \wedge f_\ell(x) = 1$. The trapdoor mode of each level checks that com' is a commitment of $\mathsf{mfe.msk}$ and that $\exists\ r$ such that $\mathsf{mfe.ct} = \mathsf{skEnc}(\mathsf{mfe.msk}, f_1 \wedge \ldots \wedge f_i; r)$.

However, we cannot argue adaptive security readily. The issue is that when we delegate, generating $\mathsf{mfe.ct}_{f\wedge g}$ in the public mode completely erases any dependence on the underlying $\mathsf{mfe.ct}_f$. To see this more clearly, consider the case when com_0 is a commitment of 1, com' is a commitment of $\mathsf{mfe.msk}$, and every proof is generated in the trapdoor mode. At this point we hope to argue that the WE instance is unsatisfiable, but that is not the case. A satisfying witness can be produced as follows. Sample π_f in the trapdoor mode using $\mathsf{mfe.ct}_f = \mathsf{skEnc}(\mathsf{mfe.msk}, f; r)$, and then simply delegate (f, π_f). While delegating, we generate $\mathsf{mfe.ct}_{f\wedge g} \leftarrow \mathsf{pkEnc}(f \wedge g)$ and the delegation proof requires no secret information at all. By generating $\mathsf{mfe.ct}_{f\wedge g}$ in the public mode we have made sure that Dec outputs 0 rather than $f \wedge g(x)$. Thus, if $f \wedge g(x) = 1$, this is a valid witness for the WE instance and we cannot use WE security any more.

Mixed Hierarchical Functional Encryption. In order to achieve adaptive security, $\mathsf{mfe.ct}_{f\wedge g}$ should be generated such that it still carries information about f, and yet this is unnoticeable by the attacker as long as $f_1 \wedge \ldots \wedge f_\ell(x) = 0$. Hence we require a hierarchical variant of mixed functional encryption in which there is a separate delegation algorithm on ciphertexts that preserves information about the chain. We call this a mixed hierarchical functional encryption scheme (Mized-HFE) and describe it now.

As in Mixed-FE, ciphertexts can be generated in a public mode using pkEnc and in a secret mode using skEnc. In addition there is a delegation algorithm $\mathsf{Delegate}$ on ciphertexts, which takes $\mathsf{mhfe.ct}$ for a chain $f_1, \ldots, f_\ell$ and a policy g and outputs a ciphertext for $f_1, \ldots, f_\ell, g$. This algorithm doesn't need any secret information. Crucially, skEnc takes a *single* policy rather than a chain, so the secret mode is injected only at the root of a chain and every deeper secret mode ciphertext arises as an image of $\mathsf{Delegate}$. We require that any ciphertext generated using pkEnc, and any ciphertext obtained by delegating on top of it, decrypts to 0 under any $\mathsf{mhfe.sk}_x$. We require that if $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{skEnc}(\mathsf{mhfe.msk}, f_1)$ and $\mathsf{mhfe.ct}_i = \mathsf{Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$ for each i, then $\mathsf{Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell) = f_1 \wedge \ldots \wedge f_\ell(x)$.

The security we require is that an attacker cannot tell which mode the roots of the chains were generated in. The attacker plays the same Store, $\mathsf{Delegate}$, $\mathsf{Corrupt}$ game as in the HABE definition and asks for a single secret key for an input x adaptively. On a $\mathsf{Corrupt}$ query it receives the ciphertext of the corrupted node only, and not those of its ancestors. An admissible attacker only corrupts nodes whose chain satisfies $f_1(x) \wedge \ldots \wedge f_\ell(x) = 0$. We give the formal definition in Section 6.

Adaptive HABE via Mixed-HFE. The main idea is to replace the bit commitments com_i from the selectively secure HABE construction with commitments of $\mathsf{mhfe.ct}_i$. More concretely, in KGen we sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{pkEnc}(f_1)$ and commit to it as $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$. In $\mathsf{Delegate}$ we

compute $\mathsf{mhfe.ct}_{\ell+1} = \mathsf{Delegate}(\mathsf{mhfe.ct}_\ell, f_{\ell+1})$, commit to it, and discard the parent's ciphertext and opening. A secret key therefore looks like

$$\mathsf{SK}_{f_1 \wedge \ldots \wedge f_\ell} = \left(f_1, \pi_1, \mathsf{com}_1^{\mathsf{ct}}, \ \ldots \ , f_\ell, \pi_\ell, \mathsf{com}_\ell^{\mathsf{ct}}, \underline{\mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}}}\right)$$

where again the underlined part is what does not survive delegation.

We cannot reuse the same trapdoor modes of the selective construction, because x^* is not available. Instead the committed Mixed-HFE ciphertexts are what testify to the value of the chain on x^*. At level 1, where we generate secret keys from master secret key, the language has a normal branch asserting that com_0 is a commitment of 0 and that $\mathsf{com}_1^{\mathsf{ct}}$ commits a ciphertext produced by $\mathsf{pkEnc}(f_1)$, and a trapdoor branch asserting that com' is a commitment of $\mathsf{mhfe.msk}$ and that $\mathsf{com}_1^{\mathsf{ct}}$ commits a ciphertext produced by $\mathsf{skEnc}(\mathsf{mhfe.msk}, f_1)$. At every deeper level the language has no trapdoor mode at all. It simply asserts that $\mathsf{com}_{\ell-1}^{\mathsf{ct}}$ commits some $\mathsf{mhfe.ct}_{\ell-1}$ and that $\mathsf{com}_\ell^{\mathsf{ct}}$ commits $\mathsf{Delegate}(\mathsf{mhfe.ct}_{\ell-1}, f_\ell)$. The WE language asks that every proof in the chain verifies, that $f_1(x) \wedge \ldots \wedge f_\ell(x) = 1$, that $\mathsf{com}_\ell^{\mathsf{ct}}$ opens to the $\mathsf{mhfe.ct}_\ell$ in the witness, and that $\mathsf{Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell) = 0$. Correctness holds because in the real scheme every chain is grown from a pkEnc root, so its ciphertexts always decrypt to 0.

The security argument would then run as follows. We delay generating any key material until a $\mathsf{Corrupt}$ query is made, at which point we build the entire chain at once. Since the ciphertexts of the ancestors are not needed, we let their commitments $\mathsf{com}_i^{\mathsf{ct}}$ commit to an all zero string. Then we switch the roots from the public mode to the secret mode, program com', move the level 1 proofs to the trapdoor branch, and set com_0 to be a commitment of 1.

The problem, and equivocal commitments. This however does not work, and the reason is exactly the distinction between selective and adaptive security for HABE. For instance, consider three policies with $f_1(x^*) = 1$, $f_2(x^*) = 0$, and $f_3(x^*) = 0$ where x^* is the (eventual) challenge attribute. The attacker stores a key for f_1, delegates it to $f_1 \wedge f_2$, delegates that to $f_1 \wedge f_2 \wedge f_3$, and corrupts the last one, which is admissible. Answering that query publishes $\mathsf{com}_2^{\mathsf{ct}}$, and by the recipe above we filled it with a commitment of zeros because we did not need to expose the ciphertext of the second node. The attacker now corrupts the key for $f_1 \wedge f_2$, which is also admissible since $f_1 \wedge f_2(x^*) = 0$. We are now required to hand over the Mixed-HFE ciphertext of that node together with an opening of $\mathsf{com}_2^{\mathsf{ct}}$ to it, and we cannot, because we already committed to an all zero string.

The fix is to use equivocal commitments. An equivocal commitment scheme has a second setup mode, computationally indistinguishable from the normal one, in which one can publish a commitment string first and only later decide which message it opens to. Binding holds only in the normal mode, which is all we need, since every step that appeals to binding happens with the normal mode in force. We use equivocal commitments for the $\mathsf{com}_i^{\mathsf{ct}}$ along a chain, publish them when a descendant is corrupted, and equivocate them later to the right ciphertext if and when their own node is corrupted.

With this in place the argument runs in the following steps: (1) delay all key material to the $\mathsf{Corrupt}$ query and simulate every proof, (2) move the commitment setup to the equivocal mode, (3) switch the roots of every chain from the public mode to the secret mode, which is where equivocation earns its keep, since the commitment of an ancestor is fixed when a descendant is corrupted and its committed value is only learned much later, (4) move the commitment setup back to the normal mode, committing honestly, (5) program com' to be a commitment of $\mathsf{mhfe.msk}$, set com_0 to be a commitment of 1, and generate the proofs honestly again, using the trapdoor branch at level 1 and the delegation branch below it.

At this point, consider any candidate WE witness. Since com_0 perfectly binds to 1, the normal branch at level 1 cannot be satisfied, so perfect soundness tells us that $\mathsf{com}_1^{\mathsf{ct}}$ commits a ciphertext produced by skEnc under the master secret key that com' binds to. The delegation proofs make sure that $\mathsf{com}_\ell^{\mathsf{ct}}$ for $\ell > 1$ commits exactly the ciphertext obtained by delegating that root along $f_2, \ldots, f_\ell$. Thus, correctness of Mixed-HFE gives $\mathsf{Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell) = f_1 \wedge \ldots \wedge f_\ell(x^*)$. But the WE language also demands that this decryption be 0 while $f_1(x^*) \wedge \ldots \wedge f_\ell(x^*) = 1$. Here, we reach a contradiction as we require $f_1 \wedge \ldots \wedge f_\ell(x^*) = 0$ and 1 simultaneously, and thus the WE instance is unsatisfiable.

3 Preliminaries

Notation. By PPT we denote probabilistic polynomial-time. We denote the security parameter by λ and the set of positive integers by $\mathbb{N}$. For any $a, b \in \{0\} \cup \mathbb{N},\ a \leq b$, we denote by $[a, b]$, the set of all integers from a to b including a and b. In other words, $[a, b] = \{a, \ldots, b\}$. We denote by $[n] := [1, n]$. If $a > b$, then $[a, b]$ should be interpreted as an empty set. We denote by $x \xleftarrow{\$} \mathcal{X}$, the process of sampling an element x from the set $\mathcal{X}$, with uniform probability. Similarly, for any PPT algorithm $\mathcal{A}$, $x \leftarrow \mathcal{A}(y)$ denotes the process of sampling x from the output distribution of $\mathcal{A}$ when run on y. By $\mathsf{negl}(\cdot)$, we denote negligible functions. By $\mathsf{poly}(\cdot)$, we denote positive polynomials.

We say that two efficiently samplable probability distributions $\mathcal{X} = \{\mathcal{X}_\lambda\}_{\lambda \in \mathbb{N}}$ and $\mathcal{Y} = \{\mathcal{Y}_\lambda\}_{\lambda \in \mathbb{N}}$ are computationally indistinguishable if for any non-uniform PPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$, and for large enough $\lambda \in \mathbb{N}$,

$$\left| \Pr_{\alpha \leftarrow \mathcal{X}_\lambda} \left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha)\right] - \Pr_{\alpha \leftarrow \mathcal{Y}_\lambda} \left[1 \leftarrow \mathcal{D}(1^\lambda, \alpha)\right] \right| \leq \mathsf{negl}(\lambda)$$

For all preliminaries below, $\mu \in \{0,1\}^*$. Note that for encryption schemes we can use hybrid encryption to encrypt messages of arbitrary length. We utilize the notation of $\overline{0}$ to denote a bit string of all zeros of sufficient length. We assume that randomized algorithms take precisely λ-bits of randomness. This is without loss of generality as this holds by relying on the existence of length doubling pseudorandom generators.

3.1 Statistically Binding Commitments

Syntax. A statistically binding commitment scheme (COM) consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$**.** The randomized setup algorithm takes as input security parameter λ and outputs a common reference string crs. The following algorithms take crs implicitly.

$\mathsf{Com}(\mu; r) \to \mathsf{com}$**.** The randomized commitment algorithm takes as input message μ, randomness r, and outputs commitment value com.

Definition 3.1 (COM)**.** A COM scheme $(\mathsf{Setup}, \mathsf{Com})$ is said to be a COM scheme if it satisfies the following properties.

Computational Hiding. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\mathcal{O}_0(\cdot,\cdot)}(\mathsf{crs})\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\mathcal{O}_1(\cdot,\cdot)}(\mathsf{crs})\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)$ and $\mathcal{O}_b(\mu_0, \mu_1)$ responds with $\mathsf{com} \leftarrow \mathsf{Com}(\mathsf{crs}, \mu_b)$ for $b \in \{0,1\}$ if $|\mu_0| = |\mu_1|$ and $\perp$ otherwise.

Statistical Binding. There exists $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\Pr\left[\begin{array}{ll} \mu_0 \neq \mu_1 \wedge |\mu_0| = |\mu_1| \wedge & \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{com} = \mathsf{Com}(\mu_0; r_0) \wedge \mathsf{com} = \mathsf{Com}(\mu_1; r_1) & : \ \exists\ (\mathsf{com}, \mu_0, \mu_1, r_0, r_1) \end{array}\right] \leq \mathsf{negl}(\lambda)$$

where probability is taken on the randomness of crs.

3.2 Non-Interactive Zero-Knowledge Proofs

Syntax. A non-interactive zero-knowledge proof system (NIZK) for $\mathcal{L} = \{x : \exists\ w, \mathsf{R}(x,w) = 1\}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$**.** The randomized setup algorithm takes as input security parameter λ and outputs a common reference string crs. The following algorithms take crs implicitly.

$\mathsf{Prove}(x, w) \to \pi$**.** The randomized proving algorithm takes as input instance $x \in \mathcal{L}$, witness w, and outputs proof π.

$\mathsf{Verify}(x, \pi) \to 0/1$**.** The deterministic verification algorithm takes as input instance x, proof π, and outputs 0 (reject) or 1 (accept).

Definition 3.2 (NIZK)**.** A NIZK scheme $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ is said to be a NIZK scheme for language $\mathcal{L}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}, x \in \mathcal{L}$, $\Pr[1 = \mathsf{Verify}(x, \pi) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda), \pi \leftarrow \mathsf{Prove}(x, w)] = 1$.

Statistical Soundness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$, $\Pr[\exists\ x \notin \mathcal{L}, \exists\ \pi$ such that $1 = \mathsf{Verify}(x, \pi) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)] \leq \mathsf{negl}(\lambda)$

Computational Zero Knowledge. For any admissible adversary $\mathcal{A}$, there exists a stateful simulator Sim such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{O}_0(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathcal{A}^{\mathcal{O}_1^{\mathsf{Sim}(\cdot)}(\cdot,\cdot)}(1^\lambda, \mathsf{crs}) : \mathsf{crs} \leftarrow \mathsf{Sim}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}_0$ uses $\pi \leftarrow \mathsf{Prove}(x, w)$ and $\mathcal{O}_1$ uses $\pi \leftarrow \mathsf{Sim}(x)$ to respond to $\mathcal{A}$'s queries of the form (x, w). A stateful PPT machine is said to be admissible if for each of its queries, $\mathsf{R}(x, w) = 1$.

Theorem 3.3 ([GGI+15])**.** A NIZK scheme for any NP language exists assuming the existence of a fully homomorphic encryption scheme [RAD78, Gen09, BV11] which in turn exists assuming the hardness of circularly-secure LWE. Moreover, this NIZK scheme can read instances x and witnesses w of unbounded size and proof size $|\pi| = |w| + \mathsf{poly}(\lambda)$.

3.3 Witness Encryption

Syntax. A witness encryption (WE) scheme for language $\mathcal{L} = \{x : \exists\ w, \mathsf{R}(x, w) = 1\}$ consists of the following polynomial time algorithms.

$\mathsf{Enc}(1^\lambda, x, \mu) \to \mathsf{CT}$**.** The randomized encryption algorithm takes as input the security parameter λ, instance x, a message μ, and outputs a ciphertext CT.

$\mathsf{Dec}(\mathsf{CT}, w) \to \mu'$**.** The deterministic decryption algorithm takes as input a ciphertext CT, a witness w, and outputs a message μ'.

Definition 3.4 (WE)**.** A WE scheme $(\mathsf{Enc}, \mathsf{Dec})$ is said to be a WE scheme for language $\mathcal{L}$ if it satisfies the following properties.

Correctness. $\forall\ \lambda \in \mathbb{N},\ \mu, x \in \mathcal{L}$, if $\mathsf{R}(x, w) = 1, \Pr[\mu = \mathsf{Dec}(\mathsf{CT}, w) : \mathsf{CT} \leftarrow \mathsf{Enc}(1^\lambda, x, \mu)] = 1$.

Semantic Security. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}, x \notin \mathcal{L}, |\mu_0| = |\mu_1|$,

$$\Pr\left[\ b \leftarrow \mathcal{A}(\mathsf{CT}) \quad : \quad \begin{array}{l}(\mu_0, \mu_1) \leftarrow \mathcal{A}(1^\lambda), b \xleftarrow{\$} \{0,1\},\\ \mathsf{CT} \leftarrow \mathsf{Enc}(1^\lambda, x, \mu_b)\end{array}\ \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

4 Hierarchical Attribute-Based Encryption

In this section, we provide the definition of a hierarchical ABE (HABE) scheme. As is convention in the literature, we use the key-policy variant of ABE where KGen is associated with predicates.

Syntax. A HABE scheme for the policy class $\mathcal{F} = \{\mathcal{F}_{n(\lambda),s(\lambda)}\}_{\lambda\in\mathbb{N}}$ consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda, 1^n, 1^s) \to (\mathsf{PP}, \mathsf{MSK})$**.** The randomized setup algorithm takes as input security parameter λ, maximum attribute length $n = n(\lambda)$, maximum policy size $s = s(\lambda)$, outputs public parameters PP and master secret key MSK. The following algorithms take PP implicitly.

$\mathsf{KGen}(\mathsf{MSK}, f) \to \mathsf{SK}_f$**.** The possibly randomized key generation algorithm takes as input master secret key MSK, policy $f \in \mathcal{F}_{n,s}$, and outputs secret key SK_f.

$\mathsf{Enc}(x, \mu) \to \mathsf{CT}$**.** The randomized encryption algorithm takes attribute $x \in \{0,1\}^n$, message $\mu \in \{0,1\}^*$, and outputs ciphertext for attribute x, CT.

$\mathsf{Delegate}(\mathsf{SK}_f, g) \to \mathsf{SK}_{f\wedge g}$**.** The possibly randomized delegation algorithm takes as input a secret key for policy f, SK_f, policy g and outputs a secret key for the policy $f \wedge g, \mathsf{SK}_{f\wedge g}$.

$\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT}) \to \mu'/\perp$**.** The deterministic decryption algorithm takes as input secret key for policy f, ciphertext for attribute x, CT, and outputs message μ' or aborts and outputs $\perp$.

Definition 4.1. A HABE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Delegate})$ is said to be a HABE scheme for $\mathcal{F}_{n,s}$ if it satisfies the following properties.

Correctness. For any $\lambda \in \mathbb{N}$, positive polynomial $n = n(\lambda), s = s(\lambda), f \in \mathcal{F}_{n,s}, x \in \{0,1\}^n, \mu \in \{0,1\}^*$,

$$\Pr\left[\begin{array}{l|ll} f(x) = 1 \wedge & : & (\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^s), \\ \mu = \mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT}) & & \mathsf{SK}_f \leftarrow \mathsf{KGen}(\mathsf{MSK}, f), \\ & & \mathsf{CT} \leftarrow \mathsf{Enc}(x, \mu) \end{array}\right] = 1$$

Delegation Correctness. This is defined similarly to correctness. Except that the policy f can now be parsed as $(f_1 \wedge \ldots \wedge f_\ell)$ for some $\ell \in [d]$ where delegation depth $d = d(\lambda)$ is a positive polynomial. Here, SK_f is defined as follows:

$$\mathsf{SK}_{f_1} \leftarrow \mathsf{KGen}(\mathsf{MSK}, f_1) \text{ and for } i \in [2, \ell], \mathsf{SK}_{f_1 \wedge \ldots \wedge f_i} \leftarrow \mathsf{Delegate}(\mathsf{SK}_{f_1 \wedge \ldots \wedge f_{i-1}}, f_i)$$

Dec with high probability now outputs $\perp$ if for any $i \in [\ell], f_i(x) = 0$ and μ otherwise.

Adaptive Security. For any admissible PPT adversary, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[\begin{array}{l|ll} |\mu_0| = |\mu_1| \wedge & : & (1^n, 1^s) \leftarrow \mathcal{A}(1^\lambda), \\ b \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{CT}) & & (\mathsf{PP}, \mathsf{MSK}) \leftarrow \mathsf{Setup}(1^\lambda, 1^n, 1^s), \\ & & (x^*, \mu_0, \mu_1) \leftarrow \mathcal{A}^{\mathcal{O}(\cdot,\cdot)}(\mathsf{PP}), \\ & & b \xleftarrow{\$} \{0,1\}, \mathsf{CT} \leftarrow \mathsf{Enc}(x^*, \mu_b) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathcal{O}$ is a stateful oracle that initiates $\mathsf{h} := 1$ and answers these queries:

- $(\mathsf{Store}, f) \to \mathsf{h}$. $\mathcal{A}$ sends policy f. $\mathcal{O}$ samples $\mathsf{SK}_f \leftarrow \mathsf{KGen}(\mathsf{MSK}, f)$, stores $(\mathsf{SK}_f, \mathsf{h}, \perp)$, and sends with h to $\mathcal{A}$. Also, update $\mathsf{h} := \mathsf{h} + 1$.
- $(\mathsf{Delegate}, (\mathsf{h}', g)) \to \mathsf{h}$. $\mathcal{A}$ sends handle h' and policy g. If there is no tuple of the form $(\mathsf{SK}_f, \mathsf{h}', *)$, output $\perp$. Otherwise, $\mathcal{O}$ samples $\mathsf{SK}_{f \wedge g} \leftarrow \mathsf{Delegate}(\mathsf{SK}_f, g)$, stores $(\mathsf{SK}_{f \wedge g}, \mathsf{h}, \mathsf{h}')$ and sends h to $\mathcal{A}$. Also, update $\mathsf{h} := \mathsf{h} + 1$.
- $(\mathsf{Corrupt}, \mathsf{h}') \to \mathsf{SK}_f$. $\mathcal{A}$ sends handle h'. If there is no tuple of the form $(\mathsf{SK}_f, \mathsf{h}', *)$, output $\perp$. Otherwise, send SK_f to $\mathcal{A}$.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if for any $\mathsf{Corrupt}$ query made by $\mathcal{A}$, SK_f is for a policy such that $f(x^*) = 0$.

Definition 4.2 (Selectively secure HABE)**.** A HABE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Delegate}, \mathsf{Dec})$ is said to be a selectively secure HABE scheme for policy class $\mathcal{F}_{n,s}$ if in the above security game, the challenge attribute x^* is declared a-priori by $\mathcal{A}$ even before receiving PP.

Definition 4.3 (Adaptive/Selective HABE with Bounded Delegations)**.** A HABE scheme $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Enc}, \mathsf{Delegate}, \mathsf{Dec})$ is said to be a HABE scheme with bounded delegations for policy class $\mathcal{F}_{n,s}$ if Enc can only generate CT that can be decrypted by a secret key which is delegated at most $d = d(\lambda)$ times. Enc takes this bound 1^d as additional input. Adaptive and selective security definitions are also defined accordingly where $\mathcal{A}$ declares 1^d along with messages μ_0, μ_1.

5 Selectively Secure HABE from Witness Encryption

In this section, we provide the construction of a selectively secure HABE scheme with bounded delegations.

Construction 5.1 (HABE)**.** We provide the construction of a selectively secure HABE scheme (Definition 4.3) for any family of polynomial-size policies using the following components:

- A statistically-sound NIZK scheme (Definition 3.2) for languages $\mathcal{L}$ (Figure 1).
- A WE scheme (Definition 3.4) for language $\mathcal{L}_{\mathsf{WE}}$ (Figure 2).
- A statistically-binding COM scheme (Definition 3.1).

Language $\mathcal{L}$

Instance: $\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell-1]}, f_\ell, \mathsf{com}_\ell$ for some $\ell \in \mathbb{N}$
Witness: $r^{(0)}_{\ell-1}, r^{(0)}_{\ell}, r^{(1)}, x^*$
Relation: For $\ell = 1$, read $\mathsf{com}_{\ell-1}$ as com_0 and $r^{(0)}_{\ell-1}$ as $r^{(0)}$. Output 1 if and only if $(\phi_1 \wedge \phi_2) \vee (\phi_3 \wedge \phi_4 \wedge \phi_5) \vee (\phi_6 \wedge \phi_7) = 1$ where,

- $\phi_1 : \mathsf{com}_{\ell-1} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 0; r^{(0)}_{\ell-1})$
- $\phi_2 : \mathsf{com}_{\ell} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 0; r^{(0)}_{\ell})$
- $\phi_3 : \mathsf{com}' \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, x^*; r^{(1)})$
- $\phi_4 : \mathsf{com}_{\ell-1} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 1; r^{(0)}_{\ell-1}) \wedge \mathsf{com}_{\ell} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 0; r^{(0)}_{\ell})$
- $\phi_5 : f_\ell(x^*) \stackrel{?}{=} 0.$
- $\phi_6 : \mathsf{com}_{\ell-1} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 1; r^{(0)}_{\ell-1})$
- $\phi_7 : \mathsf{com}_{\ell} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 1; r^{(0)}_{\ell})$

Figure 1: Description of $\mathcal{L}$

Language $\mathcal{L}_{\mathsf{WE}}$

Instance: $\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s, d, x$
Witness: $(f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell-1]}, f_\ell, \pi_\ell, \mathsf{com}_\ell, r^{(0)}_{\ell}$ for some $\ell \in [d]$.
Relation: Output 1 if and only if,

1. For each $j \in [\ell]$, $1 \stackrel{?}{=} \mathsf{NIZK.Verify}(\mathsf{nizk.crs}, (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_k, \pi_k, \mathsf{com}_k)_{k\in[j-1]}, f_j, \mathsf{com}_j), \pi_j)$.
2. $1 \stackrel{?}{=} f_1(x) \wedge \ldots \wedge f_\ell(x)$
3. $\mathsf{com}_\ell \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 0; r^{(0)}_{\ell})$

Figure 2: Description of $\mathcal{L}_{\mathsf{WE}}$

$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^s)}$. Sample $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. In addition, sample $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda)$.

Compute $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$ and $\mathsf{com}' = \mathsf{Com}(0^n; r^{(1)})$ where $r^{(0)}, r^{(1)} \xleftarrow{\$} \{0,1\}^\lambda$.

Set $\mathsf{PP} := (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$ and $\mathsf{MSK} := r^{(0)}$. Output $(\mathsf{PP}, \mathsf{MSK})$.

$\underline{\mathsf{KGen}(\mathsf{MSK}, f)}$. Parse MSK as $r^{(0)}$. Sample $\mathsf{com}_f = \mathsf{Com}(0; r_f^{(0)})$ where $r_f^{(0)} \leftarrow \{0,1\}^\lambda$. Compute $\pi_f \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}, \mathtt{wit})$ where $\mathtt{inst} := (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f, \mathsf{com}_f), \mathtt{wit} := (r^{(0)}, r_f^{(0)}, \overline{0})$. Output $\mathsf{SK}_f := (f, \pi_f, \mathsf{com}_f, r_f^{(0)})$.

$\underline{\mathsf{Enc}(1^d, x, \mu)}$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}, \mu)$ where $\mathtt{inst} := (\mathsf{PP}, d, x)$.

Output $\mathsf{CT} := (x, d, \mathsf{we.ct})$.

$\underline{\mathsf{Delegate}(\mathsf{SK}_f, g)}$. Parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ for some $\ell \in \mathbb{N}$.

Sample $\mathsf{com}_g = \mathsf{Com}(0; r_g^{(0)})$ where $r_g^{(0)} \leftarrow \{0,1\}^\lambda$.

Sample $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}, \mathtt{wit})$ where $\mathtt{inst} := (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, g, \mathsf{com}_g), \mathtt{wit} := (r_\ell^{(0)}, r_g^{(0)}, \overline{0})$.

Output $\mathsf{SK}_{f\wedge g} := ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, g, \pi_{\ell+1}, \mathsf{com}_g, r_g^{(0)})$.

$\underline{\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT})}$. Parse CT as $(x, d, \mathsf{we.ct})$ and SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ for some ℓ. If $\ell > d$ output $\perp$. Otherwise, if $f_i(x) = 0$ for any $i \in [\ell]$, abort and output $\perp$. Otherwise, output $\mu' := \mathsf{WE.Dec}(\mathsf{we.ct}, \mathsf{SK}_f)$.

Correctness. The correctness of the construction follows from the correctness of WE, NIZK, and COM. In this case, $\mathsf{SK}_f = (f, \pi_f, \mathsf{com}_f, r_f^{(0)})$ and $\mathsf{CT} = (x, d, \mathsf{we.ct})$. Since $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$ for some $r^{(0)}$ and $\mathsf{com}_f = \mathsf{Com}(0; r_f^{(0)})$, we have by correctness of NIZK and COM, $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f, \mathsf{com}_f)), \pi_f) = 1$. Hence, if $f(x) = 1$, we have by correctness of WE, $\mu' = \mu$.

Delegation Correctness. We argue this by induction on $\ell \in [d]$. When $\ell = 1$, its the same as correctness property. Consider $\ell = 2$. We have that $\mathsf{SK}_{f\wedge g} = (f, \pi_f, \mathsf{com}_f, g, \pi_g, \mathsf{com}_g, r_g^{(0)})$. If SK_f was generated using KGen, we have that $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, (\mathsf{com}, \mathsf{com}_0, \mathsf{com}', f, \mathsf{com}_f), \pi_f) = 1$ (from correctness property) and $\mathsf{com}_f = \mathsf{Com}(0; r_f^{(0)})$ for some $r_f^{(0)}$. By construction, we have $\mathsf{com}_g = \mathsf{Com}(0; r_g^{(0)})$. Hence, $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, \mathtt{inst}_2, \pi_g) = 1$ for $\mathtt{inst}_2 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f, \pi_f, \mathsf{com}_f, g, \mathsf{com}_g)$. Hence, by correctness of WE if $(f \wedge g)(x) = 1$, i.e, $f(x) = 1 \wedge g(x) = 1$, we have that $\mu' = \mu$.

For the induction step, assume that delegation correctness holds for some $k \in [d-1]$ levels of delegation. And assume that $f_1 \wedge \ldots \wedge f_k(x) = 1$ and $g(x) = 1$. Arguing similarly as above, $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, \mathtt{inst}_{k+1}, \pi_{k+1}) = 1$. This is because by induction hypothesis, $\forall\, i \in [2, k]$, $\mathsf{NIZK.Verify}(\mathsf{nizk.crs}, (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_j, \pi_j, \mathsf{com}_j)_{j\in[i-1]}, f_i, \mathsf{com}_i), \pi_i) = 1$, $\mathsf{com}_{k+1} = \mathsf{Com}(0; r_{k+1}^{(0)})$, and $\mathsf{com}_k = \mathsf{Com}(0; r_k^{(0)})$ for some $r_k^{(0)}, r_{k+1}^{(0)}$. Thus, by WE correctness, $\mu' = \mu$.

Delegation Depth. The $\mathcal{L}$ for NIZK proof only checks openings of commitments and evaluates f_ℓ on x^*. Hence $|\mathtt{wit}| = \mathsf{poly}(\lambda, n)$, and by Theorem 3.3, $|\pi_\ell| = \mathsf{poly}(\lambda, n)$, independent of ℓ. Thus Construction 5.1 supports any polynomial $d = d(\lambda)$, and $|\mathsf{SK}_{f_1 \wedge \ldots \wedge f_\ell}| = \ell \cdot \mathsf{poly}(\lambda, n, s)$ grows only linearly in the number of delegations.

5.1 Security Analysis

In this section, we prove that Construction 5.1 satisfies selective security as defined in Definition 4.3. Specifically, we prove the following theorem.

Theorem 5.2. If WE is a WE scheme for $\mathcal{L}_{\mathsf{WE}}$ (Definition 3.4), COM is a COM scheme (Definition 3.1), NIZK is a statistically-sound NIZK scheme for $\mathcal{L}$ (Definition 3.2), then Construction 5.1 is a selectively secure HABE scheme with bounded delegations (Definition 4.3) for polynomial-size policies and any positive polynomial delegation bound $d = d(\lambda)$.

Proof. We prove this theorem using the following experiments and lemmas.

Notation for delegation chains. The chain of a handle h is the sequence of policies $f_1, \ldots, f_\ell$ read off the path from the root down to h, and ℓ is the depth of h. We index the commitments carried by a secret key by depth, so a handle of depth ℓ stores $(f_i, \pi_i, \mathsf{com}_i)_{i \in [\ell]}$ together with the opening $r_\ell^{(0)}$ of com_ℓ. Following the convention of Figure 1, we write com_0 for the commitment sampled by Setup and $r_0^{(0)}$ for $r^{(0)}$, so that every proof in a chain is about a consecutive pair $(\mathsf{com}_{i-1}, \mathsf{com}_i)$ with $i \in [\ell]$.

$\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$. This is the honest experiment with $\mathcal{O}$ where either $b = 0/1$.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda), \mathsf{com}_0 = \mathsf{Com}(0; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^n; r^{(1)})$, $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r_0^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$ where $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$ and $\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1), \mathtt{wit}_1 = (r_0^{(0)}, r_1^{(0)}, \overline{0})$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i \in [\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. Sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$ where $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$ and $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i \in [\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1}), \mathtt{wit}_{\ell+1} = (r_\ell^{(0)}, r_{\ell+1}^{(0)}, \overline{0})$.

  Set $\mathsf{SK}_{f \wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i \in [\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f \wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.
- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$. In this hybrid, we set com' to be a commitment of x^*. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda), \mathsf{com}_0 = \mathsf{Com}(0; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$, $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r_0^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store, Delegate, Corrupt.** Same as $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$.
- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$. In this experiment, we simulate the NIZK common reference string and every NIZK proof. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda), \mathsf{com}_0 = \mathsf{Com}(0; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$, $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Sim}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r_0^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$ where $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$ and $\pi_1 \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. Sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$ where $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$ and $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.
- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$. In this experiment, we change the committed values along every delegation chain. Concretely, we set com_0 to be a commitment of 1, and we set com_i to be a commitment of 1 for as long as the prefix $f_1 \wedge \ldots \wedge f_i$ satisfies x^*. The first depth at which the prefix stops satisfying x^* is the shallowest depth at which $\mathcal{A}$ is allowed to corrupt, and from that depth onwards the commitments stay commitments of 0. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(1; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$, $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Sim}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. If $f_1(x^*) = 1$, sample $\mathsf{com}_1 = \mathsf{Com}(1; r_1^{(0)})$. Otherwise sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$. Here $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_1 \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. If $f_1(x^*) \wedge \ldots \wedge f_{\ell+1}(x^*) = 1$, sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(1; r_{\ell+1}^{(0)})$. Otherwise sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$. Here $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$. Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.

- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

We record two simple observations about $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$ that we use repeatedly.

Claim 5.3. In $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$, for every handle of depth ℓ and every $i \in [2, \ell]$, if com_{i-1} is a commitment of 0 then com_i is a commitment of 0.

Proof. For $i \geq 2$, com_{i-1} is a commitment of 0 exactly when $f_1(x^*) \wedge \ldots \wedge f_{i-1}(x^*) = 0$, and in that case $f_1(x^*) \wedge \ldots \wedge f_i(x^*) = 0$ as well, so com_i is a commitment of 0. □

Claim 5.4. In $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$, for every $\mathsf{Corrupt}$ query made by an admissible $\mathcal{A}$ on a handle of depth ℓ, the commitment com_ℓ is a commitment of 0.

Proof. Admissibility of $\mathcal{A}$ requires $f_1 \wedge \ldots \wedge f_\ell(x^*) = 0$ for every corrupted handle, hence com_ℓ is sampled as $\mathsf{Com}(0; r_\ell^{(0)})$. □

$\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$. In this experiment, we go back to an honestly generated common reference string and we generate every NIZK proof honestly, using the branch of $\mathcal{L}$ that matches the values committed by the pair $(\mathsf{com}_{i-1}, \mathsf{com}_i)$. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(1; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$, $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. If $f_1(x^*) = 1$, sample $\mathsf{com}_1 = \mathsf{Com}(1; r_1^{(0)})$. Otherwise sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$. Here $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_1 \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$ and $\mathtt{wit}_1 = (r_0^{(0)}, r_1^{(0)}, r^{(1)}, x^*)$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. If $f_1(x^*) \wedge \ldots \wedge f_{\ell+1}(x^*) = 1$, sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(1; r_{\ell+1}^{(0)})$. Otherwise sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$. Here $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Prove}(\mathsf{nizk.crs}, \mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$ and $\mathtt{wit}_{\ell+1} = (r_\ell^{(0)}, r_{\ell+1}^{(0)}, r^{(1)}, x^*)$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.
- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

The following claim shows that the witnesses used in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ are valid, which is what makes the experiment well defined and what makes the reduction of Lemma 5.9 admissible.

Claim 5.5. For every instance $\mathtt{inst}_i$ arising in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$, the witness $\mathtt{wit}_i = (r_{i-1}^{(0)}, r_i^{(0)}, r^{(1)}, x^*)$ satisfies $\mathsf{R}(\mathtt{inst}_i, \mathtt{wit}_i) = 1$ for the relation of Figure 1.

Proof. By Claim 5.3 there are only three possibilities for the values committed by $(\mathsf{com}_{i-1}, \mathsf{com}_i)$.

- Both com_{i-1} and com_i are commitments of 1. Then $\phi_6 \wedge \phi_7$ holds using $r_{i-1}^{(0)}$ and $r_i^{(0)}$.
- com_{i-1} is a commitment of 1 and com_i is a commitment of 0. Then ϕ_4 holds using $r_{i-1}^{(0)}$ and $r_i^{(0)}$, and ϕ_3 holds using $r^{(1)}$ since $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$. Moreover $f_1(x^*) \wedge \ldots \wedge f_{i-1}(x^*) = 1$ while $f_1(x^*) \wedge \ldots \wedge f_i(x^*) = 0$, which forces $f_i(x^*) = 0$ and hence ϕ_5 holds. For $i = 1$ this reads $f_1(x^*) = 0$, which is exactly the condition under which com_1 was sampled as a commitment of 0.
- Both com_{i-1} and com_i are commitments of 0. Then $\phi_1 \wedge \phi_2$ holds using $r_{i-1}^{(0)}$ and $r_i^{(0)}$.

The remaining case, where com_{i-1} is a commitment of 0 and com_i is a commitment of 1, does not occur by Claim 5.3. Thus, the claim follows. □

$\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$. In this experiment, we encrypt an all zero string using WE. This experiment is independent of b. Changes are highlighted in red.

- **Setup, Store, Delegate, Corrupt.** Same as $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$.
- **Encryption.** $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, 0^{|\mu_b|})$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

Indistinguishability of Hybrids. Let $\mathcal{A}$ be an admissible adversary in the HABE security definition. By $p_i^{\mathcal{A}}$ we denote the probability of $b' = b$ in the experiment $\mathbf{Expt}_i^{\mathcal{A},b}(1^\lambda)$. That is $p_i^{\mathcal{A}} := \Pr\left[b \leftarrow \mathbf{Expt}_i^{\mathcal{A},b}(1^\lambda)\right]$.

Lemma 5.6. Assuming that COM satisfies computational hiding property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_0^{\mathcal{A}} - p_1^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational hiding property of COM. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Receive $\mathsf{com.crs}$ from $\mathcal{O}$. Compute $\mathsf{com}_0 = \mathsf{Com}(0; r_0^{(0)})$. Send $(0^n, x^*)$ to $\mathcal{O}$ to receive com'. Sample $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Setup}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r_0^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store, Delegate, Corrupt.** Same as $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$.
- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ commits to 0^n, $\mathcal{B}$ simulates $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ commits to x^*, $\mathcal{B}$ behaves like $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational hiding property of COM that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 5.7. Assuming that NIZK satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational zero-knowledge property of NIZK. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda), \mathsf{com}_0 = \mathsf{Com}(0; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$. Receive $\mathsf{nizk.crs}$ from $\mathcal{O}$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r_0^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. Sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$ where $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Send $(\mathtt{inst}_1, \mathtt{wit}_1)$ to $\mathcal{O}$ to receive π_1, where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$ and $\mathtt{wit}_1 = (r_0^{(0)}, r_1^{(0)}, \overline{0})$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \bot)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\bot$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. Sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$ where $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Send $(\mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ to $\mathcal{O}$ to receive $\pi_{\ell+1}$, where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$ and $\mathtt{wit}_{\ell+1} = (r_\ell^{(0)}, r_{\ell+1}^{(0)}, \overline{0})$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\bot$. Otherwise, send SK_f to $\mathcal{A}$.

- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ uses $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove})$, $\mathcal{B}$ simulates $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ uses $\mathsf{NIZK.Sim}$, $\mathcal{B}$ behaves like $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational zero-knowledge property of NIZK that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 5.8. Assuming that COM satisfies computational hiding property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational hiding property of COM. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Receive $\mathsf{com.crs}$ from $\mathcal{O}$. Send $(0,1)$ to $\mathcal{O}$ to receive com_0. Compute $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$ and sample $\mathsf{nizk.crs} \leftarrow \mathsf{NIZK.Sim}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. If $f_1(x^*) = 1$, send $(0,1)$ to $\mathcal{O}$ to receive com_1 and set $r_1^{(0)} := \bot$. Otherwise sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$ where $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_1 \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \bot)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. If $f_1(x^*) \wedge \ldots \wedge f_{\ell+1}(x^*) = 1$, send $(0,1)$ to $\mathcal{O}$ to receive $\mathsf{com}_{\ell+1}$ and set $r_{\ell+1}^{(0)} := \perp$. Otherwise sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$ where $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Sample $\pi_{\ell+1} \leftarrow \mathsf{NIZK.Sim}(\mathtt{inst}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.

- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ commits to 0, $\mathcal{B}$ simulates $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ commits to 1, $\mathcal{B}$ behaves like $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational hiding property of COM that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 5.9. Assuming that NIZK satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_3^{\mathcal{A}} - p_4^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_3^{\mathcal{A}} - p_4^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational zero-knowledge property of NIZK. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s, x^*$. Compute $\mathsf{com.crs} \leftarrow \mathsf{COM.Setup}(1^\lambda), \mathsf{com}_0 = \mathsf{Com}(1; r_0^{(0)})$, $\mathsf{com}' = \mathsf{Com}(x^*; r^{(1)})$. Receive $\mathsf{nizk.crs}$ from $\mathcal{O}$. Set $\mathsf{PP} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. If $f_1(x^*) = 1$, sample $\mathsf{com}_1 = \mathsf{Com}(1; r_1^{(0)})$. Otherwise sample $\mathsf{com}_1 = \mathsf{Com}(0; r_1^{(0)})$. Here $r_1^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Send $(\mathtt{inst}_1, \mathtt{wit}_1)$ to $\mathcal{O}$ to receive π_1, where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', f_1, \mathsf{com}_1)$ and $\mathtt{wit}_1 = (r_0^{(0)}, r_1^{(0)}, r^{(1)}, x^*)$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1, r_1^{(0)})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, r_\ell^{(0)})$ and set $f_{\ell+1} := g$. If $f_1(x^*) \wedge \ldots \wedge f_{\ell+1}(x^*) = 1$, sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(1; r_{\ell+1}^{(0)})$. Otherwise sample $\mathsf{com}_{\ell+1} = \mathsf{Com}(0; r_{\ell+1}^{(0)})$. Here $r_{\ell+1}^{(0)} \xleftarrow{\$} \{0,1\}^\lambda$. Send $(\mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ to $\mathcal{O}$ to receive $\pi_{\ell+1}$, where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1})$ and $\mathtt{wit}_{\ell+1} = (r_\ell^{(0)}, r_{\ell+1}^{(0)}, r^{(1)}, x^*)$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell+1]}, r_{\ell+1}^{(0)})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.

- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, \mu_0, \mu_1$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*)$. Send $\mathsf{CT} = (x^*, d, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ uses $\mathsf{NIZK.Sim}$, $\mathcal{B}$ simulates $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ uses $(\mathsf{NIZK.Setup}, \mathsf{NIZK.Prove})$, $\mathcal{B}$ behaves like $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational zero-knowledge property of NIZK that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 5.10. Assuming the semantic security property of WE, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_4^{\mathcal{A}} - p_5^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. We first argue that in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$, the instance $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{nizk.crs}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', n, s, d, x^*)$ is unsatisfiable with overwhelming probability.

By statistical binding of COM (Definition 3.1), with overwhelming probability over $\mathsf{com.crs}$ no commitment string admits two distinct openings. We condition on this event. In particular, com_0 opens only to 1 and com' opens only to x^*, since these are the values committed in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$. Similarly, by statistical soundness of NIZK (Definition 3.2), with overwhelming probability over $\mathsf{nizk.crs}$ there is no instance outside $\mathcal{L}$ that admits an accepting proof. We condition on this event as well.

Assume towards a contradiction that there is a witness $\mathtt{wit}_{\mathsf{WE}} = ((f_i, \pi_i, \mathsf{com}_i)_{i\in[\ell-1]}, f_\ell, \pi_\ell, \mathsf{com}_\ell, r_\ell^{(0)})$ for some $\ell \in [d]$ such that $\mathsf{WE.R}(\mathtt{inst}_{\mathsf{WE}}, \mathtt{wit}_{\mathsf{WE}}) = 1$. By the first condition of Figure 2, for each $j \in [\ell]$ the proof π_j verifies with respect to the instance $\mathtt{inst}_j = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', (f_k, \pi_k, \mathsf{com}_k)_{k\in[j-1]}, f_j, \mathsf{com}_j)$, hence $\mathtt{inst}_j \in \mathcal{L}$.

For each $j \in \{0\} \cup [\ell]$, let β_j denote the unique value to which com_j opens, which is well defined whenever an opening exists, and recall that $\beta_0 = 1$. Fix any $j \in [\ell]$ and consider which disjunct of the relation of Figure 1 is satisfied by $\mathtt{inst}_j$.

- If $\phi_1 \wedge \phi_2$ is satisfied, then $\beta_{j-1} = 0$ and $\beta_j = 0$.
- If $\phi_3 \wedge \phi_4 \wedge \phi_5$ is satisfied, then $\beta_{j-1} = 1$ and $\beta_j = 0$. Moreover ϕ_3 forces the value x^* inside the witness to be an opening of com', which by binding is the challenge attribute itself, and therefore ϕ_5 gives $f_j(x^*) = 0$.
- If $\phi_6 \wedge \phi_7$ is satisfied, then $\beta_{j-1} = 1$ and $\beta_j = 1$.

In every case $\beta_j \leq \beta_{j-1}$, and whenever $\beta_{j-1} = 1$ and $\beta_j = 0$ we additionally obtain $f_j(x^*) = 0$.

By the third condition of Figure 2, $\mathsf{com}_\ell = \mathsf{Com}(\mathsf{com.crs}, 0; r_\ell^{(0)})$, hence $\beta_\ell = 0$. Since $\beta_0 = 1$ and the sequence $\beta_0, \ldots, \beta_\ell$ is non-increasing, there is a smallest index $\ell^* \in [\ell]$ with $\beta_{\ell^*} = 0$, and for that index $\beta_{\ell^*-1} = 1$. By the case analysis above, $f_{\ell^*}(x^*) = 0$. On the other hand, the second condition of Figure 2 gives $f_1(x^*) \wedge \ldots \wedge f_\ell(x^*) = 1$, which gives $f_{\ell^*}(x^*) = 1$. This is a contradiction, hence no such $\mathtt{wit}_{\mathsf{WE}}$ exists and $\mathcal{L}_{\mathsf{WE}}$ contains no instance of the above form.

Thus, we can use the semantic security of WE to switch μ_b to $0^{|\mu_b|}$. Any PPT adversary that can notice this switch can be readily used to break the semantic security of WE, since the reduction generates PP and answers every query of $\mathcal{A}$ exactly as in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$, forwards $(\mu_b, 0^{|\mu_b|})$ to its own challenger, and outputs 0 if and only if $\mathcal{A}$ guesses b correctly. Thus, $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Note that the ciphertext in $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$ is independent of b. Thus $p_5^{\mathcal{A}} = \frac{1}{2}$. Hence, security of Construction 5.1 follows. □

6 Mixed Hierarchical Functional Encryption

In this section, we formally define a mixed hierarchical functional encryption scheme ($\mathsf{Mixed\text{-}HFE}$) that we use to realize an adaptively secure hierarchical attribute-based encryption scheme. A $\mathsf{Mixed\text{-}HFE}$ scheme extends a mixed functional encryption scheme [GKW18, CVW+18] with the ability to publicly delegate ciphertexts.

Syntax. A mixed hierarchical functional encryption scheme ($\mathsf{Mixed\text{-}HFE}$) for any polynomial-size policies consists of the following polynomial-time algorithms.

$\mathsf{Setup}(1^\lambda, 1^s) \to \mathsf{pp}$**.** The randomized setup algorithm takes as input security parameter λ, policy index s, and outputs public parameters pp. The following algorithms take pp implicitly.

$\mathsf{MSKGen}(\mathsf{pp}; r^*) \to \mathsf{msk}$**.** The master secret key generation algorithm takes as input public parameters pp, randomness r^*, and outputs master secret key msk. This algorithm is deterministic given r^*.

$\mathsf{pkEnc}(f; \rho) \to \mathsf{ct}$**.** The randomized *public key* encryption algorithm takes as input a policy $f \in \mathcal{F}_s$, randomness ρ, and outputs a ciphertext ct.

$\mathsf{skEnc}(\mathsf{msk}, f; \rho) \to \mathsf{ct}$**.** The randomized *secret key* encryption algorithm takes as input master secret key msk, a policy $f \in \mathcal{F}_s$, randomness ρ, and outputs a ciphertext ct.

$\mathsf{Delegate}(\mathsf{ct}, g) \to \mathsf{ct}'$**.** The delegation algorithm takes as input a ciphertext ct, a policy $g \in \mathcal{F}_s$, and outputs a ciphertext ct'. This algorithm is public and deterministic.

$\mathsf{KGen}(\mathsf{msk}, x; \rho_x) \to \mathsf{sk}_x$**.** The randomized key generation algorithm takes as input master secret key msk, input x, randomness ρ_x, and outputs a secret key sk_x.

$\mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}) \to 0/1$**.** The deterministic decryption algorithm takes as input secret key sk_x, ciphertext ct, and outputs either 0 or 1.

We additionally require that ciphertext length be publicly determined by the depth. That is, there is an efficiently computable function len such that every ciphertext for a chain of length ℓ has size $\mathsf{len}(\lambda, s, \ell)$, whether it was produced by pkEnc, by skEnc, or by $\ell - 1$ applications of $\mathsf{Delegate}$.

Definition 6.1 ($\mathsf{Mixed\text{-}HFE}$)**.** A $\mathsf{Mixed\text{-}HFE}$ scheme ($\mathsf{Setup}, \mathsf{MSKGen}, \mathsf{pkEnc}, \mathsf{skEnc}, \mathsf{Delegate}, \mathsf{KGen}, \mathsf{Dec}$) is said to be a mixed hierarchical functional encryption scheme for policy class $\mathcal{F} = \{\mathcal{F}_s\}_{s \in \mathbb{N}}$ if it satisfies the following properties.

Public Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$, positive polynomial $s = s(\lambda)$, any $\ell \in \mathbb{N}$, any input x, and any policies $f_1, \ldots, f_\ell \in \mathcal{F}_s$,

$$\Pr\left[\begin{array}{lcl} & & \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^s), \\ & & \mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp}), \\ 0 = \mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}_\ell) & : & \mathsf{sk}_x \leftarrow \mathsf{KGen}(\mathsf{msk}, x), \\ & & \mathsf{ct}_1 \leftarrow \mathsf{pkEnc}(f_1), \\ & & \forall\ i \in [2, \ell], \mathsf{ct}_i = \mathsf{Delegate}(\mathsf{ct}_{i-1}, f_i) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Secret Correctness. There exists a negligible function $\mathsf{negl}(\cdot)$ such that for any $\lambda \in \mathbb{N}$ and positive polynomial $s = s(\lambda)$, and any input x,

$$\Pr\left[\begin{array}{l} \forall\ \ell \in \mathbb{N}, \forall\ f_1, \ldots, f_\ell \in \mathcal{F}_s, \forall\ \rho, \\ f_1 \wedge \ldots \wedge f_\ell(x) = \mathsf{Dec}(\mathsf{sk}_x, \mathsf{ct}_\ell) \\ \text{where } \mathsf{ct}_1 = \mathsf{skEnc}(\mathsf{msk}, f_1; \rho) \text{ and} \\ \forall\ i \in [2, \ell], \mathsf{ct}_i = \mathsf{Delegate}(\mathsf{ct}_{i-1}, f_i) \end{array} \quad : \quad \begin{array}{l} \mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^s), \\ \mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp}), \\ \mathsf{sk}_x \leftarrow \mathsf{KGen}(\mathsf{msk}, x) \end{array}\right] \geq 1 - \mathsf{negl}(\lambda)$$

Mode Indistinguishability. For any admissible PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[b \leftarrow \mathbf{Expt}_b^{\mathcal{A}}(1^\lambda)\ :\ b \xleftarrow{\$} \{0,1\}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda)$$

where $\mathbf{Expt}_b^{\mathcal{A}}(1^\lambda)$ is defined as follows.

- **Setup.** $\mathcal{A}$ sends 1^s. Sample $\mathsf{pp} \leftarrow \mathsf{Setup}(1^\lambda, 1^s)$ and $\mathsf{msk} \leftarrow \mathsf{MSKGen}(\mathsf{pp})$. Send pp to $\mathcal{A}$. Initialize $\mathsf{h} := 1$.
- **Store.** $\mathcal{A}$ sends policy f. Store $(f, \mathsf{h}, \perp)$ and send h to $\mathcal{A}$. Increment $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends handle h' and policy g. If there is no entry of the form $(*, \mathsf{h}', *)$, respond with $\perp$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$ and send h to $\mathcal{A}$. Increment $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends handle h'. If there is no entry of the form $(*, \mathsf{h}', *)$, respond with $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the sequence of policies on the path from the root down to h'. Let $j \in \{0\} \cup [\ell]$ be the largest index such that a ciphertext ct_j was already recorded for the depth-j ancestor of h', where $j = 0$ means that no ancestor carries a ciphertext.
  If $j = 0$, sample $\mathsf{ct}_1 \leftarrow \mathsf{pkEnc}(f_1)$ when $b = 0$ and $\mathsf{ct}_1 \leftarrow \mathsf{skEnc}(\mathsf{msk}, f_1)$ when $b = 1$, record ct_1 for the depth-1 ancestor, and set $j := 1$.
  For each $i \in [j+1, \ell]$, compute $\mathsf{ct}_i = \mathsf{Delegate}(\mathsf{ct}_{i-1}, f_i)$ and record ct_i for the depth-i ancestor. Previously recorded ciphertexts are never resampled.
  Send ct_ℓ to $\mathcal{A}$.
- **Key Query.** $\mathcal{A}$ sends an input x. This query is allowed only once. Sample $\mathsf{sk}_x \leftarrow \mathsf{KGen}(\mathsf{msk}, x)$ and send sk_x to $\mathcal{A}$.
- **Store, Delegate, Corrupt.** These are handled similarly to the queries made before the key query.
- **Guess.** $\mathcal{A}$ outputs a guess b'. Output b'.

A stateful PPT machine $\mathcal{A}$ is said to be admissible if it makes exactly one key query, for an input x, and for every Corrupt query made by $\mathcal{A}$ on a handle whose path carries the policies $f_1, \ldots, f_\ell$, we have $f_1 \wedge \ldots \wedge f_\ell(x) = 0$.

7 Adaptively Secure HABE from Witness Encryption and Mixed Hierarchical Functional Encryption

In this section, we provide the construction of an adaptively secure HABE scheme with bounded delegations from witness encryption and a mixed hierarchical functional encryption scheme (Definition 6.1). The construction follows the chaining idea of Construction 5.1, where the difference is that the chain now commits to Mixed-HFE ciphertexts rather than to bits, and the selective trapdoor condition $f_\ell(x^*) = 0$ is replaced by a condition on the mode in which the root ciphertext of the chain was generated.

7.1 Preliminary: Equivocal Commitments

Syntax. An equivocal commitment scheme (eCOM) consists of the following polynomial time algorithms.

$\mathsf{Setup}(1^\lambda) \to \mathsf{crs}$**.** The randomized binding mode setup algorithm takes as input security parameter λ and outputs a common reference string crs. The following algorithms take crs implicitly.

$\mathsf{Com}(\mu; r) \to \mathsf{com}$**.** The randomized commitment algorithm takes as input message μ, randomness r, and outputs commitment value com.

$\mathsf{EqSetup}(1^\lambda) \to (\mathsf{crs}, \mathsf{td})$**.** The randomized equivocal mode setup algorithm takes as input security parameter λ and outputs a common reference string crs together with a trapdoor td.

$\mathsf{EqCommit}(\mathsf{td}; \rho) \to \mathsf{com}$**.** The randomized equivocal commitment algorithm takes as input trapdoor td, randomness ρ, and outputs a commitment value com. Note that this algorithm does not take a message.

$\mathsf{Equiv}(\mathsf{td}, \rho, \mu) \to r$**.** The deterministic equivocation algorithm takes as input trapdoor td, randomness ρ, message μ, and outputs randomness r.

Definition 7.1 (eCOM)**.** An eCOM scheme $(\mathsf{Setup}, \mathsf{Com}, \mathsf{EqSetup}, \mathsf{EqCommit}, \mathsf{Equiv})$ is said to be an equivocal commitment scheme if it satisfies the following properties.

Binding Mode. The pair $(\mathsf{Setup}, \mathsf{Com})$ is a COM scheme as per Definition 3.1. In particular it satisfies computational hiding and statistical binding.

Mode Indistinguishability. For any PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}) : \mathsf{crs} \leftarrow \mathsf{Setup}(1^\lambda)\right] - \Pr\left[1 \leftarrow \mathcal{A}(\mathsf{crs}) : (\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{EqSetup}(1^\lambda)\right]\right| \leq \mathsf{negl}(\lambda)$$

Computational Equivocality. For any stateful PPT adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\forall\ \lambda \in \mathbb{N}$,

$$\left|\Pr\left[0 \leftarrow \mathcal{A}^{\mathcal{O}_0(\cdot)}(\mathsf{crs})\right] - \Pr\left[0 \leftarrow \mathcal{A}^{\mathcal{O}_1(\cdot)}(\mathsf{crs})\right]\right| \leq \mathsf{negl}(\lambda)$$

where $(\mathsf{crs}, \mathsf{td}) \leftarrow \mathsf{EqSetup}(1^\lambda)$ in both experiments, and on a query μ the oracles respond as follows. $\mathcal{O}_0$ samples $r \xleftarrow{\$} \{0,1\}^\lambda$ and responds with $(\mathsf{Com}(\mathsf{crs}, \mu; r), r)$. $\mathcal{O}_1$ samples $\rho \xleftarrow{\$} \{0,1\}^\lambda$, sets $\mathsf{com} = \mathsf{EqCommit}(\mathsf{td}; \rho)$ and $r = \mathsf{Equiv}(\mathsf{td}, \rho, \mu)$, and responds with (com, r).

7.2 Construction

Construction 7.2 (Adaptively Secure HABE)**.** We provide the construction of an adaptively secure HABE scheme with bounded d-many delegations (Definition 4.3) for any family of polynomial-size policies using the following components.

- A statistically-sound NIZK scheme (Definition 3.2) for languages $\mathcal{L}_1$ and $\mathcal{L}_{\mathsf{Del}}$ (Figure 3 and Figure 4 respectively). We denote the two instantiations by NIZK_1 and $\mathsf{NIZK}_{\mathsf{Del}}$.
- A WE scheme (Definition 3.4) for language $\mathcal{L}_{\mathsf{WE}}$ (Figure 5).
- An equivocal commitment scheme (Definition 7.1).
- A Mixed-HFE scheme (Definition 6.1) for polynomial-size policies.

Language $\mathcal{L}_1$

Instance: $\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}}$
Witness: $r^{(0)}, \rho, r_1^{\mathsf{ct}}, r^{(1)}, r^*$
Relation: Output 1 if and only if $(\phi_1 \wedge \phi_2) \vee (\phi_3 \wedge \phi_4) = 1$ where,

- $\phi_1 : \mathsf{com}_0 \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, 0; r^{(0)})$
- $\phi_2 : \mathsf{com}_1^{\mathsf{ct}} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho); r_1^{\mathsf{ct}})$
- $\phi_3 : \mathsf{com}' \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, r^*; r^{(1)})$
- $\phi_4 : \mathsf{com}_1^{\mathsf{ct}} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*), f_1; \rho); r_1^{\mathsf{ct}})$

Figure 3: Description of $\mathcal{L}_1$

Language $\mathcal{L}_{\mathsf{Del}}$

Instance: $\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell-1]}, f_\ell, \mathsf{com}_\ell^{\mathsf{ct}}$ for some $\ell \geq 2$
Witness: $\mathsf{mhfe.ct}_{\ell-1}, r_{\ell-1}^{\mathsf{ct}}, r_\ell^{\mathsf{ct}}$
Relation: Output 1 if and only if $\psi_1 \wedge \psi_2 = 1$ where,

- $\psi_1 : \mathsf{com}_{\ell-1}^{\mathsf{ct}} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, \mathsf{mhfe.ct}_{\ell-1}; r_{\ell-1}^{\mathsf{ct}})$
- $\psi_2 : \mathsf{com}_\ell^{\mathsf{ct}} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{\ell-1}, f_\ell); r_\ell^{\mathsf{ct}})$

Figure 4: Description of $\mathcal{L}_{\mathsf{Del}}$

$\underline{\mathsf{Setup}(1^\lambda, 1^n, 1^s)}$**.** Sample $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$ and $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Setup}(1^\lambda)$. Sample $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$ and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Compute $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$ and $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$ where $r^{(0)}, r^{(1)} \stackrel{\$}{\leftarrow} \{0,1\}^\lambda$.

Set $\mathsf{PP} := (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$ and $\mathsf{MSK} := r^{(0)}$. Output $(\mathsf{PP}, \mathsf{MSK})$.

Language $\mathcal{L}_{\mathsf{WE}}$

Instance: $\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, n, s, d, x, \mathsf{mhfe.sk}_x$
Witness: $(f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}}$ for some $\ell \in [d]$.
Relation: Output 1 if and only if,

1. $1 \stackrel{?}{=} \mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \pi_1)$ and for each $j \in [2,\ell]$, $1 \stackrel{?}{=} \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Verify}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_j, \pi_j)$.
2. $1 \stackrel{?}{=} f_1(x) \wedge \ldots \wedge f_\ell(x)$
3. $\mathsf{com}_\ell^{\mathsf{ct}} \stackrel{?}{=} \mathsf{Com}(\mathsf{com.crs}, \mathsf{mhfe.ct}_\ell; r_\ell^{\mathsf{ct}})$
4. $0 \stackrel{?}{=} \mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell)$

Here, $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and, for $j \in [2,\ell]$, $\mathtt{inst}_j = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[j-1]}, f_j, \mathsf{com}_j^{\mathsf{ct}})$.

Figure 5: Description of $\mathcal{L}_{\mathsf{WE}}$

$\underline{\mathsf{KGen}(\mathsf{MSK}, f_1)}$**.** Parse MSK as $r^{(0)}$. Sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho)$ and $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$ where $\rho, r_1^{\mathsf{ct}} \stackrel{\$}{\leftarrow} \{0,1\}^\lambda$.

Compute $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{inst}_1 := (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and $\mathtt{wit}_1 := (r^{(0)}, \rho, r_1^{\mathsf{ct}}, \overline{0})$.

Output $\mathsf{SK}_{f_1} := (f_1, \pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$.

$\underline{\mathsf{Enc}(1^d, x, \mu)}$**.** Sample $\mathsf{mhfe.msk}' \leftarrow \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$ and $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}', x)$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}, \mu)$ where $\mathtt{inst} := (\mathsf{PP}, d, x, \mathsf{mhfe.sk}_x)$.

Output $\mathsf{CT} := (x, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$.

$\underline{\mathsf{Delegate}(\mathsf{SK}_f, g)}$**.** Parse SK_f as $((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ for some $\ell \in \mathbb{N}$ and set $f_{\ell+1} := g$.

Compute $\mathsf{mhfe.ct}_{\ell+1} = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_\ell, f_{\ell+1})$, sample $\mathsf{com}_{\ell+1}^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_{\ell+1}; r_{\ell+1}^{\mathsf{ct}})$ where $r_{\ell+1}^{\mathsf{ct}} \stackrel{\$}{\leftarrow} \{0,1\}^\lambda$.

Compute $\pi_{\ell+1} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} := (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1}^{\mathsf{ct}})$ and $\mathtt{wit}_{\ell+1} := (\mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}}, r_{\ell+1}^{\mathsf{ct}})$.

Output $\mathsf{SK}_{f\wedge g} := ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell+1]}, \mathsf{mhfe.ct}_{\ell+1}, r_{\ell+1}^{\mathsf{ct}})$.

$\underline{\mathsf{Dec}(\mathsf{SK}_f, \mathsf{CT})}$**.** Parse CT as $(x, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ and SK_f as $((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ for some ℓ. If $\ell > d$ output $\perp$. Otherwise, if $f_i(x) = 0$ for any $i \in [\ell]$, abort and output $\perp$. Otherwise, output $\mu' := \mathsf{WE.Dec}(\mathsf{we.ct}, \mathsf{SK}_f)$.

Correctness. The correctness of the construction follows from the correctness of WE, NIZK, Mixed-HFE, and eCOM. We have $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$ and $\mathsf{CT} = (x, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$. Since $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$ and $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$ for $\mathsf{mhfe.ct}_1 = \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho)$, the witness $\mathtt{wit}_1$ satisfies $\phi_1 \wedge \phi_2$, hence by correctness of NIZK we have $\mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \pi_1) = 1$. The third condition of Figure 5 holds by construction using r_1^{ct}. Moreover $\mathsf{mhfe.sk}_x$ is generated by Mixed-HFE.KGen under a master secret key sampled independently of everything

else, so by public correctness of Mixed-HFE we have $\mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_1) = 0$ with overwhelming probability. Hence, if $f_1(x) = 1$, we have by correctness of WE, $\mu' = \mu$.

Delegation Correctness. We argue this by induction on $\ell \in [d]$. When $\ell = 1$, it is the same as the correctness property. Consider $\ell = 2$. We have that $\mathsf{SK}_{f_1 \wedge f_2} = (f_1, \pi_1, \mathsf{com}_1^{\mathsf{ct}}, f_2, \pi_2, \mathsf{com}_2^{\mathsf{ct}}, \mathsf{mhfe.ct}_2, r_2^{\mathsf{ct}})$ where $\mathsf{mhfe.ct}_2 = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_1, f_2)$. If SK_{f_1} was generated using KGen, then $\mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \pi_1) = 1$ from the correctness property and $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$ for some r_1^{ct}. By construction $\mathsf{com}_2^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_2; r_2^{\mathsf{ct}})$, hence $\mathtt{wit}_2 = (\mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}}, r_2^{\mathsf{ct}})$ satisfies $\psi_1 \wedge \psi_2$ and therefore $\mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Verify}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_2, \pi_2) = 1$. Moreover, by public correctness of Mixed-HFE, with overwhelming probability $\mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_2) = 0$. Hence, by correctness of WE if $(f_1 \wedge f_2)(x) = 1$, we have that $\mu' = \mu$.

For the induction step, assume that delegation correctness holds for some $k \in [d-1]$ levels of delegation, and assume that $f_1 \wedge \ldots \wedge f_k(x) = 1$ and $f_{k+1}(x) = 1$. By induction hypothesis, $\mathsf{NIZK}_1.\mathsf{Verify}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \pi_1) = 1$ and $\forall\ i \in [2, k], \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Verify}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_i, \pi_i) = 1$, and moreover $\mathsf{com}_k^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_k; r_k^{\mathsf{ct}})$ for some r_k^{ct}. Arguing similarly as above, $\mathtt{wit}_{k+1} = (\mathsf{mhfe.ct}_k, r_k^{\mathsf{ct}}, r_{k+1}^{\mathsf{ct}})$ satisfies $\psi_1 \wedge \psi_2$, hence $\mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Verify}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_{k+1}, \pi_{k+1}) = 1$. Since $\mathsf{mhfe.ct}_{k+1}$ is obtained from $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1)$ by k applications Mixed-HFE.Delegate, public correctness of Mixed-HFE gives $\mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_{k+1}) = 0$ with overwhelming probability. Thus, by WE correctness, $\mu' = \mu$.

7.3 Security Analysis

In this section, we prove that Construction 7.2 satisfies adaptive security as defined in Definition 4.1. Specifically, we prove the following theorem.

Theorem 7.3. If WE is a WE scheme for $\mathcal{L}_{\mathsf{WE}}$ (Definition 3.4), eCOM is an equivocal commitment scheme (Definition 7.1), NIZK_1 and $\mathsf{NIZK}_{\mathsf{Del}}$ are statistically-sound NIZK schemes for $\mathcal{L}_1$ and $\mathcal{L}_{\mathsf{Del}}$ (Definition 3.2), Mixed-HFE is a mixed hierarchical functional encryption scheme (Definition 6.1) for polynomial-size policies, then Construction 7.2 is an adaptively secure HABE scheme with bounded delegations (Definition 4.3) for polynomial-size policies and any positive polynomial delegation bound $d = d(\lambda)$.

Proof. We prove this theorem using the following experiments and lemmas.

Notation for delegation chains. The chain of a handle h is the sequence of policies $f_1, \ldots, f_\ell$ read off the path from the root down to h, and ℓ is the depth of h. We index the material carried by a secret key by depth, so a handle of depth ℓ corresponds to $(f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell]}$ together with the pair $(\mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$. From $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ onwards the challenger materializes this data on-the-fly. We say that the depth-i node of a chain is materialized if the challenger has already recorded a tuple $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i, r_i^{\mathsf{ct}})$ for it, and recorded tuples are never resampled.

$\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$. This is the honest experiment with $\mathcal{O}$ where either $b = 0/1$.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Setup}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. Sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho_1)$, $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and $\mathtt{wit}_1 = (r^{(0)}, \rho_1, r_1^{\mathsf{ct}}, \overline{0})$. Set $\mathsf{SK}_{f_1} = (f_1, \pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$. Store $(\mathsf{SK}_{f_1}, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, parse SK_f as $((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ and set $f_{\ell+1} := g$. Compute $\mathsf{mhfe.ct}_{\ell+1} = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_\ell, f_{\ell+1})$, sample $\mathsf{com}_{\ell+1}^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_{\ell+1}; r_{\ell+1}^{\mathsf{ct}})$ and $\pi_{\ell+1} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_{\ell+1}, \mathtt{wit}_{\ell+1})$ where $\mathtt{inst}_{\ell+1} = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, f_{\ell+1}, \mathsf{com}_{\ell+1}^{\mathsf{ct}})$ and $\mathtt{wit}_{\ell+1} = (\mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}}, r_{\ell+1}^{\mathsf{ct}})$.

  Set $\mathsf{SK}_{f\wedge g} = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell+1]}, \mathsf{mhfe.ct}_{\ell+1}, r_{\ell+1}^{\mathsf{ct}})$. Store $(\mathsf{SK}_{f\wedge g}, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(\mathsf{SK}_f, \mathsf{h}', *)$, send $\perp$. Otherwise, send SK_f to $\mathcal{A}$.

- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.msk}' \leftarrow \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$ and $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}', x^*)$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$. In this experiment we delay all key material generation to the Corrupt query. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$.

- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let $j \in \{0\} \cup [\ell]$ be the largest index whose node is materialized, where $j = 0$ means that no node on the chain is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho_1)$, $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \mathtt{wit}_1)$. Materialize the depth-1 node with $(\pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$ and set $j := 1$.

  For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_i, \mathtt{wit}_i)$, and materialize the depth-i node with $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i, r_i^{\mathsf{ct}})$.

  Here $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$, $\mathtt{wit}_1 = (r^{(0)}, \rho_1, r_1^{\mathsf{ct}}, \overline{0})$, and for $i \geq 2$, $\mathtt{inst}_i = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_k, \pi_k, \mathsf{com}_k^{\mathsf{ct}})_{k\in[i-1]}, f_i, \mathsf{com}_i^{\mathsf{ct}})$ and $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r_{i-1}^{\mathsf{ct}}, r_i^{\mathsf{ct}})$.

  Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.

- **Encryption.** Same as $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\underline{\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)}$**.** In this experiment we simulate the common reference string and the proofs of NIZK_1. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(1^\lambda)$.
- **Store, Delegate.** Same as $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$.
- **Corrupt.** Same as $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$, except $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(\mathtt{inst}_1)$.
- **Encryption, Guess.** Same as $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$.

$\underline{\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)}$**.** In this experiment we simulate the common reference string and the proofs of $\mathsf{NIZK}_{\mathsf{Del}}$. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(1^\lambda)$.
- **Store, Delegate.** Same as $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$.
- **Corrupt.** Same as $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$, except for each $i \in [j+1, \ell]$, $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(\mathtt{inst}_i)$.
- **Encryption, Guess.** Same as $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$.

At this point no experiment uses any NIZK witness, hence no experiment needs an opening of any $\mathsf{com}_i^{\mathsf{ct}}$ in order to answer a Store or a $\mathsf{Delegate}$ query.

$\underline{\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)}$**.** In this experiment we sample the commitment common reference string in the equivocal mode. Commitments are still generated by Com. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $(\mathsf{com.crs}, \mathsf{td}) \leftarrow \mathsf{eCOM.EqSetup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(1^\lambda)$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \bot)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\bot$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\bot$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let $j \in \{0\} \cup [\ell]$ be the largest index whose node is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho_1)$, $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(\mathtt{inst}_1)$. Materialize the depth-1 node with $(\pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$ and set $j := 1$.

For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(\mathtt{inst}_i)$, and materialize the depth-i node with $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i, r_i^{\mathsf{ct}})$.

Here $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and, for $i \geq 2$, $\mathtt{inst}_i = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_k, \pi_k, \mathsf{com}_k^{\mathsf{ct}})_{k\in[i-1]}, f_i, \mathsf{com}_i^{\mathsf{ct}})$.

Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.

- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.msk}' \leftarrow \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$ and $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}', x^*)$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$. In this experiment we generate every $\mathsf{com}_i^{\mathsf{ct}}$ using $\mathsf{EqCommit}$ and produce its opening using Equiv. Note that com_0 and com' are still generated by Com. Changes are highlighted in red.

- **Setup, Store, Delegate.** Same as $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$.
- **Corrupt.** Same as $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$, except that for every node materialized during this query, sample $\rho_i^{\mathsf{ct}} \xleftarrow{\$} \{0,1\}^\lambda$, set $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{EqCommit}(\mathsf{td}; \rho_i^{\mathsf{ct}})$ and $r_i^{\mathsf{ct}} = \mathsf{Equiv}(\mathsf{td}, \rho_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i)$.
- **Encryption, Guess.** Same as $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$.

$\mathbf{Expt}_6^{\mathcal{A},b}(1^\lambda)$. In this experiment we generate the Mixed-HFE ciphertexts in the secret key mode and we generate $\mathsf{mhfe.sk}_x$ under the same master secret key. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$, except sample $\mathsf{mhfe.msk} \leftarrow \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$.
- **Store, Delegate.** Same as $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$.
- **Corrupt.** Same as $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho_1)$ when the depth-1 node is materialized.
- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}, x^*)$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$.

  Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.
- **Store, Delegate, Corrupt, Guess.** Same as $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$.

$\mathbf{Expt}_7^{\mathcal{A},b}(1^\lambda)$. In this experiment we sample the master secret key of Mixed-HFE from an explicit random string r^*. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_6^{\mathcal{A},b}(1^\lambda)$, except sample $r^* \xleftarrow{\$} \{0,1\}^\lambda$ and set $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$.
- **Store, Delegate, Corrupt, Encryption, Guess.** Same as $\mathbf{Expt}_6^{\mathcal{A},b}(1^\lambda)$.

$\underline{\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)}$. In this experiment we go back to generating every $\mathsf{com}_i^{\mathsf{ct}}$ using Com. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $(\mathsf{com.crs}, \mathsf{td}) \leftarrow \mathsf{eCOM.EqSetup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(1^\lambda)$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Sample $r^* \xleftarrow{\$} \{0,1\}^\lambda$ and set $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s, d)$, $\mathsf{MSK} := r^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let $j \in \{0\} \cup [\ell]$ be the largest index whose node is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho_1)$, $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$ where $r_1^{\mathsf{ct}} \xleftarrow{\$} \{0,1\}^\lambda$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(\mathtt{inst}_1)$. Materialize the depth-1 node with $(\pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$ and set $j := 1$.

  For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$ where $r_i^{\mathsf{ct}} \xleftarrow{\$} \{0,1\}^\lambda$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(\mathtt{inst}_i)$, and materialize the depth-i node with $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i, r_i^{\mathsf{ct}})$.

  Here $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and, for $i \geq 2$, $\mathtt{inst}_i = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_k, \pi_k, \mathsf{com}_k^{\mathsf{ct}})_{k \in [i-1]}, f_i, \mathsf{com}_i^{\mathsf{ct}})$.

  Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.
- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}, x^*)$ and $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\underline{\mathbf{Expt}_9^{\mathcal{A},b}(1^\lambda)}$. In this experiment we set com' to be a commitment of r^*. The commitment common reference string is still equivocal at this point. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{com}' = \mathsf{Com}(r^*; r^{(1)})$.
- **Store, Delegate, Corrupt, Encryption, Guess.** Same as $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$.

$\underline{\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)}$. In this experiment we sample the commitment common reference string in the binding mode. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_9^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$ and no trapdoor is generated.
- **Store, Delegate, Corrupt, Encryption, Guess.** Same as $\mathbf{Expt}_9^{\mathcal{A},b}(1^\lambda)$.

$\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$. In this experiment we set com_0 to be a commitment of 1. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)$, except $\mathsf{com}_0 = \mathsf{Com}(1; r^{(0)})$ and MSK is no longer set.
- **Store, Delegate, Corrupt, Encryption, Guess.** Same as $\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)$.

The following claim records that the trapdoor witnesses are available from $\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$ onwards.

Claim 7.4. In $\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$, for every materialized node the following hold. The witness $\mathtt{wit}_1 = (\overline{0}, \rho_1, r_1^{\mathsf{ct}}, r^{(1)}, r^*)$ satisfies $\phi_3 \wedge \phi_4$ of Figure 3, and for every $i \geq 2$ the witness $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r_{i-1}^{\mathsf{ct}}, r_i^{\mathsf{ct}})$ satisfies $\psi_1 \wedge \psi_2$ of Figure 4.

Proof. In $\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$ we have $\mathsf{com}' = \mathsf{Com}(r^*; r^{(1)})$, hence ϕ_3 holds. The depth-1 node of every materialized chain was created with $\mathsf{mhfe.ct}_1 = \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho_1)$ for $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$ and with $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, hence ϕ_4 holds. For $i \geq 2$, the depth-i node was created with $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$ and $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$, and the depth-$(i-1)$ node carries $\mathsf{com}_{i-1}^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_{i-1}; r_{i-1}^{\mathsf{ct}})$, hence $\psi_1 \wedge \psi_2$ holds. Note that the challenger stores $\mathsf{mhfe.ct}_{i-1}$ and r_{i-1}^{ct} for every materialized node, including nodes that $\mathcal{A}$ never corrupts. Thus, the claim follows. □

$\mathbf{Expt}_{12}^{\mathcal{A},b}(1^\lambda)$. In this experiment we go back to an honestly generated common reference string for NIZK_1 and we generate the proofs of NIZK_1 honestly using the trapdoor witness. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(1; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(r^*; r^{(1)})$ where $r^* \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Set $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$ and $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let $j \in \{0\} \cup [\ell]$ be the largest index whose node is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho_1)$, $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{wit}_1 = (\overline{0}, \rho_1, r_1^{\mathsf{ct}}, r^{(1)}, r^*)$. Materialize the depth-1 node with $(\pi_1, \mathsf{com}_1^{\mathsf{ct}}, \mathsf{mhfe.ct}_1, r_1^{\mathsf{ct}})$ and set $j := 1$.

  For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(\mathtt{inst}_i)$, and materialize the depth-i node with $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \mathsf{mhfe.ct}_i, r_i^{\mathsf{ct}})$.

  Here $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}_1^{\mathsf{ct}})$ and, for $i \geq 2$, $\mathtt{inst}_i = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_k, \pi_k, \mathsf{com}_k^{\mathsf{ct}})_{k \in [i-1]}, f_i, \mathsf{com}_i^{\mathsf{ct}})$.

  Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.

- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}, x^*)$ and $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

$\underline{\mathbf{Expt}^{\mathcal{A},b}_{13}(1^\lambda)}$. In this experiment we go back to an honestly generated common reference string for $\mathsf{NIZK}_{\mathsf{Del}}$ and we generate the proofs of $\mathsf{NIZK}_{\mathsf{Del}}$ honestly. Changes are highlighted in red.

- **Setup.** Same as $\mathbf{Expt}^{\mathcal{A},b}_{12}(1^\lambda)$, except $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Setup}(1^\lambda)$.

- **Store, Delegate.** Same as $\mathbf{Expt}^{\mathcal{A},b}_{12}(1^\lambda)$.

- **Corrupt.** Same as $\mathbf{Expt}^{\mathcal{A},b}_{12}(1^\lambda)$, except for each $i \in [j+1, \ell]$, $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_i, \mathtt{wit}_i)$ where $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r^{\mathsf{ct}}_{i-1}, r^{\mathsf{ct}}_i)$.

- **Encryption, Guess.** Same as $\mathbf{Expt}^{\mathcal{A},b}_{12}(1^\lambda)$.

$\underline{\mathbf{Expt}^{\mathcal{A},b}_{14}(1^\lambda)}$. In this experiment, we encrypt an all zero string using WE. This experiment is independent of b. Changes are highlighted in red.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(1; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(r^*; r^{(1)})$ where $r^* \xleftarrow{\$} \{0,1\}^\lambda$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Setup}(1^\lambda)$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Setup}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Set $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$ and $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \bot)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\bot$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\bot$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let $j \in \{0\} \cup [\ell]$ be the largest index whose node is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho_1)$, $\mathsf{com}^{\mathsf{ct}}_1 = \mathsf{Com}(\mathsf{mhfe.ct}_1; r^{\mathsf{ct}}_1)$, and $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Prove}(\mathsf{nizk.crs}_1, \mathtt{inst}_1, \mathtt{wit}_1)$ where $\mathtt{wit}_1 = (\overline{0}, \rho_1, r^{\mathsf{ct}}_1, r^{(1)}, r^*)$. Materialize the depth-1 node with $(\pi_1, \mathsf{com}^{\mathsf{ct}}_1, \mathsf{mhfe.ct}_1, r^{\mathsf{ct}}_1)$ and set $j := 1$.

  For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}^{\mathsf{ct}}_i = \mathsf{Com}(\mathsf{mhfe.ct}_i; r^{\mathsf{ct}}_i)$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_i, \mathtt{wit}_i)$ where $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r^{\mathsf{ct}}_{i-1}, r^{\mathsf{ct}}_i)$, and materialize the depth-i node with $(\pi_i, \mathsf{com}^{\mathsf{ct}}_i, \mathsf{mhfe.ct}_i, r^{\mathsf{ct}}_i)$.

  Here $\mathtt{inst}_1 = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, f_1, \mathsf{com}^{\mathsf{ct}}_1)$ and, for $i \geq 2$, $\mathtt{inst}_i = (\mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, (f_k, \pi_k, \mathsf{com}^{\mathsf{ct}}_k)_{k \in [i-1]}, f_i, \mathsf{com}^{\mathsf{ct}}_i)$.

  Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}^{\mathsf{ct}}_i)_{i \in [\ell]}, \mathsf{mhfe.ct}_\ell, r^{\mathsf{ct}}_\ell)$ to $\mathcal{A}$.

- **Encryption.** $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}, x^*)$ and $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, 0^{|\mu_b|})$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** Output whatever $\mathcal{A}$ outputs.

Indistinguishability of Hybrids. Let $\mathcal{A}$ be an admissible adversary in the HABE security definition. By $p_i^{\mathcal{A}}$ we denote the probability of $b' = b$ in the experiment $\mathbf{Expt}_i^{\mathcal{A},b}(1^\lambda)$. That is $p_i^{\mathcal{A}} := \Pr\left[b \leftarrow \mathbf{Expt}_i^{\mathcal{A},b}(1^\lambda)\right]$.

Lemma 7.5. For any admissible adversary $\mathcal{A}$, $p_0^{\mathcal{A}} = p_1^{\mathcal{A}}$.

Proof. The only change is that KGen and Delegate are delayed to the Corrupt query, and recorded material is reused rather than resampled. Since $\mathcal{A}$ receives only a handle in response to a Store or a Delegate query, its view is unchanged until an Corrupt query is made. There are 3 cases to consider.

- *No node on the chain of the corrupted handle is materialized.* In this case we run the KGen procedure at the root and then run the Delegate procedure down to the leaf, which is exactly what $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$ would have done at Store and Delegate time.
- *Some node at depth $j < \ell$ is materialized.* In this case we reuse the recorded material up to depth j and keep delegating down to depth ℓ, which is again what $\mathbf{Expt}_0^{\mathcal{A},b}(1^\lambda)$ would have produced, since Mixed-HFE.Delegate is deterministic and the remaining randomness is fresh.
- *$\mathcal{A}$ corrupts a handle of depth $\ell' < \ell$ after a handle of depth ℓ on the same chain was corrupted.* In this case the depth-ℓ' node is already materialized and its recorded tuple forms the required secret key.

Hence, these two experiments are identical. Thus, the lemma follows. □

Lemma 7.6. Assuming that NIZK_1 satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_1^{\mathcal{A}} - p_2^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational zero-knowledge property of NIZK_1. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Compute $\mathsf{com.crs} \leftarrow \mathsf{eCOM.Setup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$, $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Setup}(1^\lambda)$, and $\mathsf{mhfe.pp} \leftarrow \mathsf{Mixed\text{-}HFE.Setup}(1^\lambda, 1^s)$. Receive $\mathsf{nizk.crs}_1$ from $\mathcal{O}$. Set $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.
- **Store.** $\mathcal{A}$ sends f_1. Store $(f_1, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.
- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h' and let j be the largest index whose node is materialized.

  If $j = 0$, sample $\mathsf{mhfe.ct}_1 \leftarrow \mathsf{Mixed\text{-}HFE.pkEnc}(f_1; \rho_1)$ and $\mathsf{com}_1^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_1; r_1^{\mathsf{ct}})$, send $(\mathtt{inst}_1, \mathtt{wit}_1)$ to $\mathcal{O}$ to receive π_1, materialize the depth-1 node, and set $j := 1$.

  For each $i \in [j+1, \ell]$, compute $\mathsf{mhfe.ct}_i = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{i-1}, f_i)$, sample $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{Com}(\mathsf{mhfe.ct}_i; r_i^{\mathsf{ct}})$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Prove}(\mathsf{nizk.crs}_{\mathsf{Del}}, \mathtt{inst}_i, \mathtt{wit}_i)$, and materialize the depth-i node.

  Send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.

- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Sample $\mathsf{mhfe.msk}' \leftarrow \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$ and $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}', x^*)$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_x)$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_x, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ uses $(\mathsf{NIZK}_1.\mathsf{Setup}, \mathsf{NIZK}_1.\mathsf{Prove})$, $\mathcal{B}$ simulates $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ uses $\mathsf{NIZK}_1.\mathsf{Sim}$, $\mathcal{B}$ behaves like $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational zero-knowledge property of NIZK_1 that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_1^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_2^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 7.7. Assuming that $\mathsf{NIZK}_{\mathsf{Del}}$ satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_2^{\mathcal{A}} - p_3^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma 7.6, where the reduction now receives $\mathsf{nizk.crs}_{\mathsf{Del}}$ from its oracle and routes every proof of $\mathsf{NIZK}_{\mathsf{Del}}$ through it, while sampling $\mathsf{nizk.crs}_1$ and simulating the proofs of NIZK_1 on its own. The witnesses $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r_{i-1}^{\mathsf{ct}}, r_i^{\mathsf{ct}})$ used for $i \geq 2$ satisfy $\psi_1 \wedge \psi_2$ by construction, hence the reduction is admissible. □

Lemma 7.8. Assuming that eCOM satisfies mode indistinguishability property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_3^{\mathcal{A}} - p_4^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The reduction $\mathcal{B}$ receives $\mathsf{com.crs}$ from its challenger and then runs $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$ verbatim, sampling everything else on its own and using Com for all commitments. Note that $\mathcal{B}$ never uses a trapdoor and never invokes statistical binding, since every NIZK proof is simulated in both experiments. If $\mathsf{com.crs}$ is sampled by $\mathsf{eCOM.Setup}$, then $\mathcal{B}$ simulates $\mathbf{Expt}_3^{\mathcal{A},b}(1^\lambda)$, and if $\mathsf{com.crs}$ is sampled by $\mathsf{eCOM.EqSetup}$, then $\mathcal{B}$ behaves like $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$. The rest of the argument is as in Lemma 7.6. □

Lemma 7.9. Assuming that eCOM satisfies computational equivocality property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_4^{\mathcal{A}} - p_5^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_4^{\mathcal{A}} - p_5^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the computational equivocality property of eCOM. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows. $\mathcal{B}$ receives $\mathsf{com.crs}$ from its challenger and runs $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ with the following change. Whenever a node of depth i is materialized during a $\mathsf{Corrupt}$ query, $\mathcal{B}$ computes $\mathsf{mhfe.ct}_i$ exactly as in $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$, sends $\mathsf{mhfe.ct}_i$ to $\mathcal{O}$, and receives a pair $(\mathsf{com}_i^{\mathsf{ct}}, r_i^{\mathsf{ct}})$ which it records for that node. Everything else, including com_0 and com', is generated by $\mathcal{B}$ on its own using Com.

Note that $\mathcal{B}$ knows $\mathsf{mhfe.ct}_i$ at the moment it materializes the depth-i node, because the chain $\mathsf{mhfe.ct}_1, \ldots, \mathsf{mhfe.ct}_\ell$ is computed from $f_1, \ldots, f_\ell$ before any commitment is needed. Hence $\mathcal{B}$ is a valid adversary in the computational equivocality game. As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ is $\mathcal{O}_0$, then $\mathcal{B}$ simulates $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ and if $\mathcal{O}$ is $\mathcal{O}_1$, then $\mathcal{B}$ behaves like $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$. Hence, $\mathcal{B}$ is a valid adversary against the computational equivocality property of eCOM that can break its security with probability $\epsilon(\lambda)$. Thus, $\mathbf{Expt}_4^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 7.10. Assuming that $\mathsf{Mixed\text{-}HFE}$ satisfies mode indistinguishability property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_5^{\mathcal{A}} - p_6^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. Assume that there exists an admissible adversary $\mathcal{A}$ such that $\left|p_5^{\mathcal{A}} - p_6^{\mathcal{A}}\right| = \epsilon(\lambda)$ for some non-negligible $\epsilon(\cdot)$. We will construct a reduction adversary $\mathcal{B}$ that can break the mode indistinguishability property of $\mathsf{Mixed\text{-}HFE}$. The description of $\mathcal{B}^{\mathcal{O}}$ is as follows.

- **Setup.** $\mathcal{A}$ sends $1^n, 1^s$. Send 1^s to $\mathcal{O}$ and receive $\mathsf{mhfe.pp}$. Compute $(\mathsf{com.crs}, \mathsf{td}) \leftarrow \mathsf{eCOM.EqSetup}(1^\lambda)$, $\mathsf{com}_0 = \mathsf{Com}(0; r^{(0)})$, $\mathsf{com}' = \mathsf{Com}(0^\lambda; r^{(1)})$, $\mathsf{nizk.crs}_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(1^\lambda)$, and $\mathsf{nizk.crs}_{\mathsf{Del}} \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(1^\lambda)$. Set $\mathsf{PP} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{mhfe.pp}, \mathsf{com}_0, \mathsf{com}', n, s)$, $\mathsf{MSK} := r^{(0)}$. Initiate h to 1. Send PP to $\mathcal{A}$.

- **Store.** $\mathcal{A}$ sends f_1. Send (Store, f_1) to $\mathcal{O}$ and record the handle it returns. Store $(f_1, \mathsf{h}, \perp)$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Delegate.** $\mathcal{A}$ sends h', g. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$ to $\mathcal{A}$. Otherwise, send $(\mathsf{Delegate}, (\mathsf{h}', g))$ to $\mathcal{O}$ and record the handle it returns. Store $(g, \mathsf{h}, \mathsf{h}')$, send h to $\mathcal{A}$, and set $\mathsf{h} := \mathsf{h} + 1$.

- **Corrupt.** $\mathcal{A}$ sends h'. If there is no entry $(*, \mathsf{h}', *)$, send $\perp$. Otherwise, let $f_1, \ldots, f_\ell$ be the chain of h'. Send $(\mathsf{Corrupt}, \mathsf{h}')$ to $\mathcal{O}$ to receive $\mathsf{mhfe.ct}_\ell$.

  For every node of depth $i \in [\ell]$ on this chain that is not yet materialized, sample $\rho_i^{\mathsf{ct}} \xleftarrow{\$} \{0,1\}^\lambda$, set $\mathsf{com}_i^{\mathsf{ct}} = \mathsf{EqCommit}(\mathsf{td}; \rho_i^{\mathsf{ct}})$, sample $\pi_1 \leftarrow \mathsf{NIZK}_1.\mathsf{Sim}(\mathtt{inst}_1)$ if $i = 1$ and $\pi_i \leftarrow \mathsf{NIZK}_{\mathsf{Del}}.\mathsf{Sim}(\mathtt{inst}_i)$ otherwise, and materialize the depth-i node with $(\pi_i, \mathsf{com}_i^{\mathsf{ct}}, \rho_i^{\mathsf{ct}})$.

  Set $r_\ell^{\mathsf{ct}} = \mathsf{Equiv}(\mathsf{td}, \rho_\ell^{\mathsf{ct}}, \mathsf{mhfe.ct}_\ell)$ and send $\mathsf{SK}_f = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i \in [\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ to $\mathcal{A}$.

- **Encryption.** Sample $b \xleftarrow{\$} \{0,1\}$. $\mathcal{A}$ sends $1^d, x^*, \mu_0, \mu_1$. Send x^* to $\mathcal{O}$ as its key query and receive $\mathsf{mhfe.sk}_{x^*}$. Sample $\mathsf{we.ct} \leftarrow \mathsf{WE.Enc}(1^\lambda, \mathtt{inst}_{\mathsf{WE}}, \mu_b)$ where $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{PP}, d, x^*, \mathsf{mhfe.sk}_{x^*})$. Send $\mathsf{CT} = (x^*, d, \mathsf{mhfe.sk}_{x^*}, \mathsf{we.ct})$ to $\mathcal{A}$.

Store, Delegate, Corrupt. These are handled similarly to pre-challenge phase.

- **Guess.** $\mathcal{A}$ outputs b'. If $b = b'$, output 0. Otherwise output 1.

As we can see, the running time of $\mathcal{B}$ is polynomial in the running time of $\mathcal{A}$ and λ. If $\mathcal{O}$ uses $\beta = 0$, then every chain starts from $\mathsf{Mixed\text{-}HFE.pkEnc}(f_1)$ and $\mathsf{mhfe.sk}_x$ is generated under a master secret key that is used nowhere else, hence $\mathcal{B}$ simulates $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$. If $\mathcal{O}$ uses $\beta = 1$, then every chain starts from $\mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1)$ and $\mathsf{mhfe.sk}_x$ is generated under the same $\mathsf{mhfe.msk}$, hence $\mathcal{B}$ behaves like $\mathbf{Expt}_6^{\mathcal{A},b}(1^\lambda)$. Thus, $\mathbf{Expt}_5^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_6^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Lemma 7.11. For any admissible adversary $\mathcal{A}$, $p_6^{\mathcal{A}} = p_7^{\mathcal{A}}$.

Proof. Since $\mathsf{Mixed\text{-}HFE.MSKGen}$ is deterministic given its randomness, sampling $r^* \xleftarrow{\$} \{0,1\}^\lambda$ and setting $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$ produces exactly the distribution of $\mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp})$. The value r^* is not used anywhere else in $\mathbf{Expt}_7^{\mathcal{A},b}(1^\lambda)$. Hence, these two experiments are identical. Thus, the lemma follows. □

Lemma 7.12. Assuming that eCOM satisfies computational equivocality property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_7^{\mathcal{A}} - p_8^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma 7.9, with the roles of the two oracles exchanged. As noted after the description of $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$, in $\mathbf{Expt}_7^{\mathcal{A},b}(1^\lambda)$ the challenger computes the entire chain $\mathsf{mhfe.ct}_1, \ldots, \mathsf{mhfe.ct}_\ell$ before it publishes any commitment on that chain, so the reduction can submit each committed value to its oracle at the moment the corresponding node is materialized. □

Lemma 7.13. Assuming that eCOM satisfies computational equivocality property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_8^{\mathcal{A}} - p_9^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The only difference between $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_9^{\mathcal{A},b}(1^\lambda)$ is the message committed by com', and the commitment common reference string is sampled by $\mathsf{eCOM.EqSetup}$ in both experiments. Note that $r^{(1)}$ is not needed anywhere in either experiment, since every NIZK proof is still simulated, and that the trapdoor td is not needed either, since every $\mathsf{com}_i^{\mathsf{ct}}$ is produced by Com from $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$ onwards. Hence a reduction $\mathcal{B}$ that samples $r^* \xleftarrow{\$} \{0,1\}^\lambda$ on its own, receives a pair $(\mathsf{com.crs}, \mathsf{com}')$ from its challenger, and runs $\mathbf{Expt}_8^{\mathcal{A},b}(1^\lambda)$ otherwise, distinguishes $(\mathsf{crs}, \mathsf{Com}(\mathsf{crs}, 0^\lambda; r^{(1)}))$ from $(\mathsf{crs}, \mathsf{Com}(\mathsf{crs}, r^*; r^{(1)}))$ under an equivocal crs whenever $\mathcal{A}$ distinguishes the two experiments. These two distributions are computationally indistinguishable for every fixed pair of messages, and averaging over r^* gives the claim. The rest of the argument is as in Lemma 7.6. □

Lemma 7.14. Assuming that eCOM satisfies mode indistinguishability property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_9^{\mathcal{A}} - p_{10}^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma 7.8, with the roles of $\mathsf{eCOM.Setup}$ and $\mathsf{eCOM.EqSetup}$ exchanged. Note that in $\mathbf{Expt}_9^{\mathcal{A},b}(1^\lambda)$ the trapdoor td is no longer used anywhere, so the reduction only needs $\mathsf{com.crs}$ from its challenger. □

Lemma 7.15. Assuming that eCOM satisfies computational hiding property in the binding mode, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_{10}^{\mathcal{A}} - p_{11}^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The reduction $\mathcal{B}$ receives $\mathsf{com.crs}$ from its challenger, samples $r^* \xleftarrow{\$} \{0,1\}^\lambda$, computes $\mathsf{com}' = \mathsf{Com}(r^*; r^{(1)})$ on its own, sends $(0,1)$ to its oracle to receive com_0, and runs $\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)$ otherwise. Note that $r^{(0)}$ is not needed anywhere in $\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)$ or $\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$, since MSK is never sent to $\mathcal{A}$ and every NIZK proof is still simulated in both experiments. If the oracle commits to 0, then $\mathcal{B}$ simulates $\mathbf{Expt}_{10}^{\mathcal{A},b}(1^\lambda)$ and if it commits to 1, then $\mathcal{B}$ behaves like $\mathbf{Expt}_{11}^{\mathcal{A},b}(1^\lambda)$. The rest of the argument is as in Lemma 7.6. □

Lemma 7.16. Assuming that NIZK_1 satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_{11}^{\mathcal{A}} - p_{12}^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma 7.6, with the roles of the two oracles exchanged and with the witness $\mathtt{wit}_1 = (\overline{0}, \rho_1, r_1^{\mathsf{ct}}, r^{(1)}, r^*)$ in place of $(r^{(0)}, \rho_1, r_1^{\mathsf{ct}}, \overline{0})$. By Claim 7.4, this witness satisfies $\phi_3 \wedge \phi_4$, hence the reduction is admissible. □

Lemma 7.17. Assuming that $\mathsf{NIZK}_{\mathsf{Del}}$ satisfies computational zero-knowledge property, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_{12}^{\mathcal{A}} - p_{13}^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. The proof of this lemma is similar to the proof of Lemma 7.6, with the roles of the two oracles exchanged and with the witnesses $\mathtt{wit}_i = (\mathsf{mhfe.ct}_{i-1}, r_{i-1}^{\mathsf{ct}}, r_i^{\mathsf{ct}})$ for $i \geq 2$. By Claim 7.4, these witnesses satisfy $\psi_1 \wedge \psi_2$, hence the reduction is admissible. □

Lemma 7.18. Assuming the semantic security property of WE, for any admissible adversary $\mathcal{A}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that $\left|p_{13}^{\mathcal{A}} - p_{14}^{\mathcal{A}}\right| \leq \mathsf{negl}(\lambda)$.

Proof. We first argue that in $\mathbf{Expt}_{13}^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_{14}^{\mathcal{A},b}(1^\lambda)$, the instance $\mathtt{inst}_{\mathsf{WE}} = (\mathsf{nizk.crs}_1, \mathsf{nizk.crs}_{\mathsf{Del}}, \mathsf{com.crs}, \mathsf{com}_0, \mathsf{com}', \mathsf{mhfe.pp}, n, s, d, x^*, \mathsf{mhfe.sk}_x)$ is unsatisfiable with high probability.

Recall that $\mathsf{com.crs}$ is sampled by $\mathsf{eCOM.Setup}$ in both experiments, hence the binding mode is in force and $(\mathsf{Setup}, \mathsf{Com})$ is a COM scheme by Definition 7.1. By statistical binding, with overwhelming probability over $\mathsf{com.crs}$ no commitment string admits two distinct openings, and we condition on this event. In particular com_0 opens only to 1 and com' opens only to r^*. Similarly, by statistical soundness of NIZK_1 and $\mathsf{NIZK}_{\mathsf{Del}}$ (Definition 3.2), with overwhelming probability over $\mathsf{nizk.crs}_1$ and $\mathsf{nizk.crs}_{\mathsf{Del}}$ there is no instance outside $\mathcal{L}_1$ or outside $\mathcal{L}_{\mathsf{Del}}$ that admits an accepting proof, and we condition on this event as well. Both NIZK common reference strings are honestly generated from $\mathbf{Expt}_{13}^{\mathcal{A},b}(1^\lambda)$ onwards, which is what makes this step available.

Assume towards a contradiction that there is a witness $\mathtt{wit}_{\mathsf{WE}} = ((f_i, \pi_i, \mathsf{com}_i^{\mathsf{ct}})_{i\in[\ell]}, \mathsf{mhfe.ct}_\ell, r_\ell^{\mathsf{ct}})$ for some $\ell \in [d]$ such that $\mathsf{WE.R}(\mathtt{inst}_{\mathsf{WE}}, \mathtt{wit}_{\mathsf{WE}}) = 1$. By the first condition of Figure 5, $\mathtt{inst}_1 \in \mathcal{L}_1$ and $\mathtt{inst}_j \in \mathcal{L}_{\mathsf{Del}}$ for each $j \in [2, \ell]$.

Since com_0 opens only to 1, the conjunct ϕ_1 of Figure 3 cannot be satisfied, hence $\mathtt{inst}_1 \in \mathcal{L}_1$ forces $\phi_3 \wedge \phi_4$. By ϕ_3 and binding of com', the value r^* inside the witness is the one fixed at Setup, and therefore ϕ_4 says that $\mathsf{com}_1^{\mathsf{ct}}$ is a commitment of $\mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho)$ for some ρ, where $\mathsf{mhfe.msk} = \mathsf{Mixed\text{-}HFE.MSKGen}(\mathsf{mhfe.pp}; r^*)$.

We now show by induction on $j \in [\ell]$ that $\mathsf{com}_j^{\mathsf{ct}}$ is a commitment of $\mathsf{mhfe.ct}_j^*$, where $\mathsf{mhfe.ct}_1^* = \mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho)$ and $\mathsf{mhfe.ct}_j^* = \mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{j-1}^*, f_j)$ for $j \geq 2$. The base case is the previous paragraph. For the induction step, $\mathtt{inst}_j \in \mathcal{L}_{\mathsf{Del}}$ gives $\psi_1 \wedge \psi_2$, so $\mathsf{com}_{j-1}^{\mathsf{ct}}$ is a commitment of the value $\mathsf{mhfe.ct}_{j-1}$ appearing in the witness and $\mathsf{com}_j^{\mathsf{ct}}$ is a commitment of $\mathsf{Mixed\text{-}HFE.Delegate}(\mathsf{mhfe.ct}_{j-1}, f_j)$. By the induction hypothesis and binding, $\mathsf{mhfe.ct}_{j-1} = \mathsf{mhfe.ct}_{j-1}^*$, hence $\mathsf{com}_j^{\mathsf{ct}}$ is a commitment of $\mathsf{mhfe.ct}_j^*$.

By the third condition of Figure 5, $\mathsf{com}_\ell^{\mathsf{ct}} = \mathsf{Com}(\mathsf{com.crs}, \mathsf{mhfe.ct}_\ell; r_\ell^{\mathsf{ct}})$, so binding gives $\mathsf{mhfe.ct}_\ell = \mathsf{mhfe.ct}_\ell^*$. Now $\mathsf{mhfe.sk}_x \leftarrow \mathsf{Mixed\text{-}HFE.KGen}(\mathsf{mhfe.msk}, x^*)$ and $\mathsf{mhfe.ct}_\ell^*$ is obtained from $\mathsf{Mixed\text{-}HFE.skEnc}(\mathsf{mhfe.msk}, f_1; \rho)$ by applying $\mathsf{Mixed\text{-}HFE.Delegate}$ with $f_2, \ldots, f_\ell$, hence secret correctness of $\mathsf{Mixed\text{-}HFE}$ (Definition 6.1) gives $\mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell^*) = f_1 \wedge \ldots \wedge f_\ell(x^*)$. We stress that secret correctness is a statistical guarantee that holds simultaneously for every chain and every choice of ρ, which is what lets us apply it to a chain that $\mathcal{A}$ chose after seeing PP.

The fourth condition of Figure 5 gives $\mathsf{Mixed\text{-}HFE.Dec}(\mathsf{mhfe.sk}_x, \mathsf{mhfe.ct}_\ell) = 0$, while the second condition gives $f_1 \wedge \ldots \wedge f_\ell(x^*) = 1$. This is a contradiction, hence no such $\mathtt{wit}_{\mathsf{WE}}$ exists.

Thus, we can use the semantic security of WE to switch μ_b to $0^{|\mu_b|}$. Any PPT adversary that can notice this switch can be readily used to break the semantic security of WE, since the reduction generates PP and answers every query of $\mathcal{A}$ exactly as in $\mathbf{Expt}_{13}^{\mathcal{A},b}(1^\lambda)$, forwards $(\mu_b, 0^{|\mu_b|})$ to its own challenger, and outputs 0 if and only if $\mathcal{A}$ guesses b correctly. Thus, $\mathbf{Expt}_{13}^{\mathcal{A},b}(1^\lambda)$ and $\mathbf{Expt}_{14}^{\mathcal{A},b}(1^\lambda)$ are computationally indistinguishable. □

Note that the ciphertext in $\mathbf{Expt}_{14}^{\mathcal{A},b}(1^\lambda)$ is independent of b. Thus $p_{14}^{\mathcal{A}} = \frac{1}{2}$. Hence, security of Construction 7.2 follows. □

References

[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 Berlin 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 Berlin Heidelberg, Germany.

[ABSV15] Prabhanjan Ananth, Zvika Brakerski, Gil Segev, and Vinod Vaikuntanathan. From selective to adaptive security in functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 657–677, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[ACG+24] Abtin Afshar, Jiaqi Cheng, Rishab Goyal, Aayush Yadav, and Saikumar Yadugiri. Encrypted RAM delegation: Applications to rate-1 extractable arguments, homomorphic NIZKs, MPC, and more. Cryptology ePrint Archive, Paper 2024/1806, 2024.

[AJ15] Prabhanjan Ananth and Abhishek Jain. Indistinguishability obfuscation from compact functional encryption. In *Annual Cryptology Conference*, pages 308–326. Springer, 2015.

[AJS15] Prabhanjan Ananth, Abhishek Jain, and Amit Sahai. Indistinguishability obfuscation from functional encryption for simple functions. *Cryptology ePrint Archive*, 2015.

[AKY24] Shweta Agrawal, Simran Kumari, and Shota Yamada. Attribute based encryption for turing machines from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances*

in Cryptology – CRYPTO 2024, Part III, volume 14922 of *Lecture Notes in Computer Science*, pages 352–386, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[AMR25] Damiano Abram, Giulio Malavolta, and Lawrence Roy. Key-homomorphic computations for RAM: Fully succinct randomised encodings and more. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 236–268, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[AV19] Prabhanjan Ananth and Vinod Vaikuntanathan. Optimal bounded-collusion secure functional encryption. In *Theory of Cryptography Conference*, pages 174–198. Springer, 2019.

[AWY20] Shweta Agrawal, Daniel Wichs, and Shota Yamada. Optimal broadcast encryption from LWE and pairings in the standard model. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part I*, volume 12550 of *Lecture Notes in Computer Science*, pages 149–178, Durham, NC, USA, November 16–19, 2020. Springer, Cham, Switzerland.

[AY20] Shweta Agrawal and Shota Yamada. Optimal broadcast encryption from pairings and LWE. In Anne Canteaut and Yuval Ishai, editors, *Advances in Cryptology – EUROCRYPT 2020, Part I*, volume 12105 of *Lecture Notes in Computer Science*, pages 13–43, Zagreb, Croatia, May 10–14, 2020. Springer, Cham, Switzerland.

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

[BDS25] Pedro Branco, Nico Döttling, and Akshayaram Srinivasan. Rate-1 statistical non-interactive zero-knowledge. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part VII*, volume 16006 of *Lecture Notes in Computer Science*, pages 514–547, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

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

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology – CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 776–806, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Cham, Switzerland.

[BLSV18] Zvika Brakerski, Alex Lombardi, Gil Segev, and Vinod Vaikuntanathan. Anonymous IBE, leakage resilience and circular security from new assumptions. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part I*, volume 10820 of *Lecture Notes in Computer Science*, pages 535–564, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Cham, Switzerland.

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

[BS15] Zvika Brakerski and Gil Segev. Hierarchical functional encryption. *Cryptology ePrint Archive*, 2015.

[BSW07] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE symposium on security and privacy (SP'07)*, pages 321–334. IEEE, 2007.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *Theory of Cryptography: 8th Theory of Cryptography Conference, TCC 2011, Providence, RI, USA, March 28-30, 2011. Proceedings 8*, pages 253–273. Springer, 2011.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. Efficient fully homomorphic encryption from (standard) LWE. In Rafail Ostrovsky, editor, *52nd Annual Symposium on Foundations of Computer Science*, pages 97–106, Palm Springs, CA, USA, October 22–25, 2011. IEEE Computer Society Press.

[BV15] Nir Bitansky and Vinod Vaikuntanathan. Indistinguishability obfuscation from functional encryption. In *FOCS*, 2015.

[CGW15] Jie Chen, Romain Gay, and Hoeteck Wee. Improved dual system ABE in prime-order groups via predicate encodings. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology – EUROCRYPT 2015, Part II*, volume 9057 of *Lecture Notes in Computer Science*, pages 595–624, Sofia, Bulgaria, April 26–30, 2015. Springer Berlin Heidelberg, Germany.

[CHKP12] David Cash, Dennis Hofheinz, Eike Kiltz, and Chris Peikert. Bonsai trees, or how to delegate a lattice basis. *Journal of Cryptology*, 25(4):601–639, October 2012.

[CVW+18] Yilei Chen, Vinod Vaikuntanathan, Brent Waters, Hoeteck Wee, and Daniel Wichs. Traitor-tracing from lwe made simple and attribute-based. In *Theory of Cryptography: 16th International Conference, TCC 2018, Panaji, India, November 11–14, 2018, Proceedings, Part II 16*, pages 341–369. Springer, 2018.

[CW23] Valerio Cini and Hoeteck Wee. ABE for circuits with poly (λ)-sized keys from LWE. In *64th Annual Symposium on Foundations of Computer Science*, pages 435–446, Santa Cruz, CA, USA, November 6–9, 2023. IEEE Computer Society Press.

[CW24] Valerio Cini and Hoeteck Wee. Unbounded ABE for circuits from LWE, revisited. In Kai-Min Chung and Yu Sasaki, editors, *Advances in Cryptology – ASIACRYPT 2024, Part IV*, volume 15487 of *Lecture Notes in Computer Science*, pages 238–267, Kolkata, India, December 9–13, 2024. Springer, Singapore, Singapore.

[DG17a] Nico Döttling and Sanjam Garg. From selective IBE to full IBE and selective HIBE. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 372–408, Baltimore, MD, USA, November 12–15, 2017. Springer, Cham, Switzerland.

[DG17b] Nico Döttling and Sanjam Garg. Identity-based encryption from the Diffie-Hellman assumption. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 537–569, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Cham, Switzerland.

[FWW23] Cody Freitag, Brent Waters, and David J. Wu. How to use (plain) witness encryption: Registered ABE, flexible broadcast, and more. In Helena Handschuh and Anna Lysyanskaya, editors, *Advances in Cryptology – CRYPTO 2023, Part IV*, volume 14084 of *Lecture Notes in Computer Science*, pages 498–531, Santa Barbara, CA, USA, August 20–24, 2023. Springer, Cham, Switzerland.

[Gen09] Craig Gentry. Fully homomorphic encryption using ideal lattices. In Michael Mitzenmacher, editor, *41st Annual ACM Symposium on Theory of Computing*, pages 169–178, Bethesda, MD, USA, May 31 – June 2, 2009. ACM Press.

[GGH+13] Sanjam Garg, Craig Gentry, Shai Halevi, Mariana Raykova, Amit Sahai, and Brent Waters. Candidate indistinguishability obfuscation and functional encryption for all circuits. In *54th Annual Symposium on Foundations of Computer Science*, pages 40–49, Berkeley, CA, USA, October 26–29, 2013. IEEE Computer Society Press.

[GGI+15] Craig Gentry, Jens Groth, Yuval Ishai, Chris Peikert, Amit Sahai, and Adam Smith. Using fully homomorphic hybrid encryption to minimize non-interative zero-knowledge proofs. *Journal of Cryptology*, 28(4):820–843, 2015.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[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 Berlin Heidelberg, Germany, March 15–17, 2009.

[GKP+13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run Turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 536–553, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[GKR25] Rishab Goyal, Venkata Koppula, and Mahesh Sreekumar Rajasree. A note on adaptive security in hierarchical identity-based encryption. In Yael Tauman Kalai and Seny F. Kamara, editors, *Advances in Cryptology – CRYPTO 2025, Part III*, volume 16002 of *Lecture Notes in Computer Science*, pages 101–136, Santa Barbara, CA, USA, August 17–21, 2025. Springer, Cham, Switzerland.

[GKW18] Rishab Goyal, Venkata Koppula, and Brent Waters. Collusion resistant traitor tracing from learning with errors. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 660–670, 2018.

[GKW19] Rishab Goyal, Venkata Koppula, and Brent Waters. New approaches to traitor tracing with embedded identities. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 149–179, Nuremberg, Germany, December 1–5, 2019. Springer, Cham, Switzerland.

[GLW14] Craig Gentry, Allison B. Lewko, and Brent Waters. Witness encryption from instance independent assumptions. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 426–443, Santa Barbara, CA, USA, August 17–21, 2014. Springer Berlin Heidelberg, Germany.

[GLW21] Rishab Goyal, Jiahui Liu, and Brent Waters. Adaptive security via deletion in attribute-based encryption: Solutions from search assumptions in bilinear groups. In Mehdi Tibouchi and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2021, Part IV*, volume 13093 of *Lecture Notes in Computer Science*, pages 311–341, Singapore, December 6–10, 2021. Springer, Cham, Switzerland.

[GM15] Matthew D Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320. IEEE, 2015.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In Ari Juels, Rebecca N. Wright, and Sabrina De Capitani di Vimercati, editors, *ACM CCS 2006: 13th Conference on Computer and Communications Security*, pages 89–98, Alexandria, Virginia, USA, October 30 – November 3, 2006. ACM Press. Available as Cryptology ePrint Archive Report 2006/309.

[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 Berlin Heidelberg, Germany.

[GSW13] Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part I*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92, Santa Barbara, CA, USA, August 18–22, 2013. Springer Berlin Heidelberg, Germany.

[GVW12] Sergey Gorbunov, Vinod Vaikuntanathan, and Hoeteck Wee. Functional encryption with bounded collusions via multi-party computation. In *Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 162–179. Springer, 2012.

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

[GVW19] Rishab Goyal, Satyanarayana Vusirikala, and Brent Waters. Collusion resistant broadcast and trace from positional witness encryption. In Dongdai Lin and Kazue Sako, editors, *PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 11443 of *Lecture Notes in Computer Science*, pages 3–33, Beijing, China, April 14–17, 2019. Springer, Cham, Switzerland.

[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 Berlin Heidelberg, Germany.

[HLL23] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. Attribute-based encryption for circuits of unbounded depth from lattices. In *64th Annual Symposium on Foundations of Computer Science*, pages 415–434, Santa Cruz, CA, USA, November 6–9, 2023. IEEE Computer Society Press.

[HLL24] Yao-Ching Hsieh, Huijia Lin, and Ji Luo. A general framework for lattice-based ABE using evasive inner-product functional encryption. In Marc Joye and Gregor Leander, editors, *Advances in Cryptology – EUROCRYPT 2024, Part II*, volume 14652 of *Lecture Notes in Computer Science*, pages 433–464, Zurich, Switzerland, May 26–30, 2024. Springer, Cham, Switzerland.

[KNYY20] Shuichi Katsumata, Ryo Nishimaki, Shota Yamada, and Takashi Yamakawa. Adaptively secure inner product encryption from LWE. In Shiho Moriai and Huaxiong Wang, editors, *Advances in Cryptology – ASIACRYPT 2020, Part III*, volume 12493 of *Lecture Notes in Computer Science*, pages 375–404, Daejeon, South Korea, December 7–11, 2020. Springer, Cham, Switzerland.

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

[KW20] Lucas Kowalczyk and Hoeteck Wee. Compact adaptively secure ABE for NC^1 from k-Lin. *Journal of Cryptology*, 33(3):954–1002, July 2020.

[LLL22] Hanjun Li, Huijia Lin, and Ji Luo. ABE for circuits with constant-size secret keys and adaptive security. In Eike Kiltz and Vinod Vaikuntanathan, editors, *TCC 2022: 20th Theory of Cryptography Conference, Part I*, volume 13747 of *Lecture Notes in*

Computer Science, pages 680–710, Chicago, IL, USA, November 7–10, 2022. Springer, Cham, Switzerland.

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

[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 Berlin Heidelberg, Germany.

[LW14] Allison B. Lewko and Brent Waters. Why proving HIBE systems secure is difficult. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 58–76, Copenhagen, Denmark, May 11–15, 2014. Springer Berlin Heidelberg, Germany.

[PRV12] Bryan Parno, Mariana Raykova, and Vinod Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In Ronald Cramer, editor, *TCC 2012: 9th Theory of Cryptography Conference*, volume 7194 of *Lecture Notes in Computer Science*, pages 422–439, Taormina, Sicily, Italy, March 19–21, 2012. Springer Berlin Heidelberg, Germany.

[RAD78] Ronald L Rivest, Len Adleman, and Michael L Dertouzos. On data banks and privacy homomorphisms. *Foundations of secure computation*, 4(11):169–180, 1978.

[SS10] Amit Sahai and Hakan Seyalioglu. Worry-free encryption: functional encryption with public keys. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 463–472, 2010.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Advances in Cryptology–EUROCRYPT 2005: 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22-26, 2005. Proceedings 24*, pages 457–473. Springer, 2005.

[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 Berlin Heidelberg, Germany.

[Tsa19] Rotem Tsabary. Fully secure attribute-based encryption for t-CNF from LWE. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 62–85, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Cham, Switzerland.

[Tsa22] Rotem Tsabary. Candidate witness encryption from lattice techniques. In *Annual International Cryptology Conference*, pages 535–559. Springer, 2022.

[VWW22] Vinod Vaikuntanathan, Hoeteck Wee, and Daniel Wichs. Witness encryption and null-io from evasive lwe. In *International Conference on the Theory and Application of Cryptology and Information Security*, pages 195–221. Springer, 2022.

[Wat09] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology – CRYPTO 2009*, volume 5677 of *Lecture Notes in Computer Science*, pages 619–636, Santa Barbara, CA, USA, August 16–20, 2009. Springer Berlin Heidelberg, Germany.

[Wat11] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011: 14th International Conference on Theory and Practice of Public Key Cryptography*, volume 6571 of *Lecture Notes in Computer Science*, pages 53–70, Taormina, Italy, March 6–9, 2011. Springer Berlin Heidelberg, Germany.

[Wat15] Brent Waters. A punctured programming approach to adaptively secure functional encryption. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 678–697, Santa Barbara, CA, USA, August 16–20, 2015. Springer Berlin Heidelberg, Germany.

[Wee22] Hoeteck Wee. Optimal broadcast encryption and cp-abe from evasive lattice assumptions. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 217–241. Springer, 2022.

[Wee24] Hoeteck Wee. Circuit ABE with $\mathsf{poly}(\mathsf{depth}, \lambda)$-sized ciphertexts and keys from lattices. In Leonid Reyzin and Douglas Stebila, editors, *Advances in Cryptology – CRYPTO 2024, Part III*, volume 14922 of *Lecture Notes in Computer Science*, pages 178–209, Santa Barbara, CA, USA, August 18–22, 2024. Springer, Cham, Switzerland.

[Wee25] Hoeteck Wee. Almost optimal KP and CP-ABE for circuits from succinct LWE. In Serge Fehr and Pierre-Alain Fouque, editors, *Advances in Cryptology – EUROCRYPT 2025, Part III*, volume 15603 of *Lecture Notes in Computer Science*, pages 34–62, Madrid, Spain, May 4–8, 2025. Springer, Cham, Switzerland.

[WW24] Brent Waters and Daniel Wichs. Adaptively secure attribute-based encryption from witness encryption. In Elette Boyle and Mohammad Mahmoody, editors, *TCC 2024: 22nd Theory of Cryptography Conference, Part III*, volume 15366 of *Lecture Notes in Computer Science*, pages 65–90, Milan, Italy, December 2–6, 2024. Springer, Cham, Switzerland.