Dual System Encryption: Realizing Fully Secure IBE and HIBE under Simple Assumptions

Brent Waters *
University of Texas at Austin

Abstract

We present a new methodology for proving security of encryption systems using what we call *Dual System Encryption*. Our techniques result in *fully* secure Identity-Based Encryption (IBE) and Hierarchical Identity-Based Encryption (HIBE) systems under the simple and established decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our IBE system has ciphertexts, private keys, and public parameters each consisting of a constant number of group elements. These results are the first HIBE system and the first IBE system with short parameters under simple assumptions.

In a Dual System Encryption system both ciphertexts and private keys can take on one of two indistinguishable forms. A private key or ciphertext will be *normal* if they are generated respectively from the system's key generation or encryption algorithm. These keys and ciphertexts will behave as one expects in an IBE system. In addition, we define *semi-functional* keys and ciphertexts. A semi-functional private key will be able to decrypt all normally generated ciphertexts; however, decryption will fail if one attempts to decrypt a semi-functional ciphertext with a semi-functional private key. Analogously, semi-functional ciphertexts will be decryptable only by normal private keys.

Dual System Encryption opens up a new way to prove security of IBE and related encryption systems. We define a sequence of games where we change first the challenge ciphertext and then the private keys one by one to be semi-functional. We finally end up in a game where the challenge ciphertext and all private keys are semi-functional at which point proving security is straightforward.

*Supported by NSF CNS-0716199, Air Force Office of Scientific Research (AFOSR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO) and the U.S. Department of Homeland Security under Grant Award Number 2006-CS-001-000001.

1 Introduction

The concept of Identity-Based Encryption (IBE) was first proposed by Shamir in 1984. In an IBE system a user can encrypt to another party simply by knowing their identity as well as a set of global parameters — eliminating the need to distribute a separate public key for each user in the system.

Although the concept received much interest, it wasn't until several years later that Boneh and Franklin [7] introduced the first Identity-Based Encryption scheme using groups with efficiently computable bilinear maps. The original Boneh and Franklin result used the random oracle heuristic to prove security under the Bilinear Diffie-Hellman assumption and a significant open question was whether the random oracle model could be removed.

Following the breakthrough result of Boneh and Franklin, there has been significant progress in realizing IBE in the standard model. First, Canetti, Halevi, and Katz [14] proved security without the random oracle heuristic, but under a weaker "Selective-ID" model where the attacker must declare the identity $\mathcal{I}^*$ that he will attack *before* even seeing the system's public parameters. Boneh and Boyen [3] then provided an efficient selectively secure scheme. Subsequently, Boneh and Boyen [5] and Waters [29] gave fully secure solutions in the standard model. The Waters scheme provided an efficient and provably fully secure system in the standard model under the decisional Bilinear Diffie-Hellman assumption; however, one drawback was that the public parameters consisted of $\mathcal{O}(\lambda)$ group elements for security parameter λ.

Partitioning Reductions One very important common thread in all of the above systems is that they use what we call a *partitioning* strategy to prove security. In these systems, one proves security to an underlying complexity assumption by creating a reduction algorithm $\mathcal{B}$ that partitions the identity space into two parts — 1) identities for which it can create private keys; and 2) identities that it can use in the challenge ciphertext phase. This partitioning is embedded either in the public parameters at setup time in the standard model systems [14, 3, 5, 29] or programed into the random oracle [7]. In the selective model, systems the identity space can be "tightly" partitioned so that all the keys except $\mathcal{I}^*$ fall into the key creating partition, while reductions in fully secure systems will partition the space according to the number of private key queries $q(\lambda)$ that an attacker makes and the reduction "hopes" that the queries and challenge ciphertext identity fall favorably in the partition.

While the partitioning techniques have proved useful, they have two fundamental limitations. First, the most efficient fully secure and standard model IBE system due to Waters has large public parameters that might be impractical for some applications. The second and more compelling concern is that partitioning techniques appear to be inadequate for proving security of encryption systems that offer more functionality such as Hierarchical IBE [23, 21] and Attribute-Based Encryption [25] *even if we apply the random oracle model.* For instance, all known Hierarchical Identity-Based Encryption (HIBE) systems (in this vein) have an exponential degradation of security with the depth, n, of the hierarchy — rendering the security reductions meaningless for large n. The fundamental problem is that more advanced systems such as HIBE have more structure on the identity space that make (any known) partitioning strategies unusable. For example, in an HIBE system a partitioning reduction algorithm is constrained such that if it can create a private key for a particular identity vector then it must be able to for all of its descendants.

Moving beyond the partitioning paradigm To overcome these obstacles, Gentry [18] proposed an IBE system with short public parameters that has a security reduction which moves beyond the partitioning paradigm. In his reduction the simulator is able to create a key for all identities and also use any identity as the challenge identity $\mathcal{I}^*$. At first glance, there is an apparent paradox in this strategy since it seems that the reduction algorithm could simply answer the challenge ciphertext itself by creating a private key for $\mathcal{I}^*$. To deal with this obstacle, Gentry's reduction algorithm can only generate *one* private key for each identity. For an attacker that makes at most q queries, the algorithm embeds a degree q polynomial $F(\cdot)$ and can create a private key with a tag component $F(\mathcal{I})$ for identity $\mathcal{I}$. The challenge ciphertext for $\mathcal{I}^*$ is structured such that it decrypts to the challenge message for the single key for $\mathcal{I}^*$ that the reduction could generate even though the message might be information theoretically hidden to an attacker with no knowledge of $F(\mathcal{I}^*)$.

Although the Gentry IBE achieved security in the standard model, it did so at the cost of using a significantly more complicated assumption called the decisional q-ABHDE assumption. In this assumption a generator g raised to several powers of an exponent a are given out (e.g., $g, g^a, g^{a^2}, \ldots, g^{a^q}$). In addition to the added complexity, the actual assumption used in the proof is *dependent* on the number of private key queries the adversary makes. This seems to be inherently tied to the need to embed the degree q polynomial f into a constant number group elements.

Interestingly, Gentry and Halevi [19] recently showed how to extend these concepts to get a fully secure HIBE system, although this system actually used an even more involved assumption. In addition, the "jump" from Gentry's IBE to the HIBE system added a significant amount of complexity to the system and proof of security.

Our Contribution We present a new methodology for proving security of encryption systems using what we call *Dual System Encryption*. Our techniques result in *fully* secure IBE and HIBE systems under the simple and established decisional Bilinear Diffie-Hellman and decisional Linear assumptions. Our IBE system has ciphertexts, private keys, and public parameters each consisting of a constant number of group elements. Our results give the first HIBE system and the first IBE system with short parameters under simple assumptions.

Our conceptual approach departs significantly from both the partitioning paradigm and Gentry's approach. In a Dual System Encryption system, both ciphertexts and private keys can take on one of two indistinguishable forms. A private key or ciphertext will be *normal* if they are generated respectively from the system's key generation or encryption algorithm. These keys and ciphertexts will behave as one expects in an IBE system. In addition, we define *semi-functional* keys and ciphertexts. A semi-functional private key will be able to decrypt all normally generated ciphertexts; however, decryption will fail if one attempts to decrypt a semi-functional ciphertext with a semi-functional private key. Analogously, semi-functional ciphertexts will be decryptable only by normal private keys.

Dual System Encryption opens up a new way to prove security of IBE and related encryption systems. Intuitively, to prove security we define a sequence of games arguing that an attacker cannot distinguish one game from the next. The first game will be the real security game in which the challenge ciphertext and all private keys are distributed normally. Next, we switch our normal challenge ciphertext with a semi- functional one. We argue that no adversary can detect this (under our complexity assumption) since all private keys given can decrypt the challenge ciphertext regardless of whether it is normal or semi-functional. In the next series of games, we change the

private keys one game at a time from normal to semi-functional, again arguing indistinguishability. In both the above proof arguments, our reduction algorithm $\mathcal{B}$ will be able to provide private keys for any identity and use any identity as a challenge identity — eliminating the need to worry about an abort condition. Finally, we end up in a game where the challenge ciphertext and all private keys are semi-functional. At this point proving security is straightforward since the reduction algorithm does not need to present any normal keys to the attacker and all semi-functional keys are useless for decrypting a semi-functional ciphertext.

The reader may have noticed one issue in our indistinguishability argument over private keys. If the reduction algorithm $\mathcal{B}$ wants to know whether a secret key $\mathrm{SK}_{\mathcal{I}}$ for $\mathcal{I}$ was semi-functional, couldn't it simply create a semi-functional ciphertext for $\mathcal{I}$ and test this itself (without using the attacker)? To deal with this issue our reduction algorithm embeds a degree one polynomial $F(\mathcal{I}) = A \cdot \mathcal{I} + B$ (over $\mathbb{Z}_p$). In each hybrid game the attacker can only create a semi-functional ciphertext for ciphertext identity $\mathcal{I}_c$ with a "tag" value of $\mathrm{tag}_c = F(\mathcal{I}_c)$ and can only create a private key of unknown type for identity $\mathcal{I}_k$ with tag value of $\mathrm{tag}_k = F(\mathcal{I}_k)$. Our system use the "two equation revocation" technique of Sahai and Waters [26] to enforce that the decryption algorithm will only work if the key tag and ciphertext tag are not equal. If the reduction algorithm attempted to test the key in question, decryption would fail unconditionally; and thus independently of whether it was a semi-functional key. [1].

In reflection, one reason our dual system achieves security from a simple assumption is that by changing the keys in small hybrid steps one by one we only need to worry about the relationship between the challenge ciphertext and one private key at a time. Our function F only needs to be able to embed a degree one polynomial; in contrast the Gentry reduction "takes on" all private keys at the same time and needs a complex assumption to embed a degree q polynomial.

HIBE and Other Encryption Systems Building on our IBE system, we also provide a fully secure HIBE system. One remarkable feature is that the added complexity of the solution is rather small. Furthermore, our system combines the structure of the Boneh-Boyen [3] selective-ID HIBE. This hints that we can leverage our methodology to adapt ideas from other selectively secure encryption systems (or those with complex assumptions) into fully secure ones under simple assumptions and also that prior selectively secure systems may have "lead us down the right path".

We believe that our Dual System methodology in the future will become a catalyst for proving adaptive security under simple assumptions for several other encryption systems including: Anonymous IBE and searchable encryption [6, 1, 13, 12, 27], Broadcast Encryption [17, 10], and Attribute-Based Encryption [25]. To add credence to this belief we give an adaptively secure broadcast system in Appendix D proven under the same simple assumptions. Our broadcast system has ciphertext overhead of a constant number of group elements and is the first such system with a proof under a simple assumption.

Other Related Work We note that there are remarkable IBE systems of Cocks [16] and Boneh, Gentry, and Hamburg [9] based on the quadratic residuosity assumption and Gentry, Peikert, and Vaikuntanathan [20] based on lattice assumptions. These systems are all proven secure under the random oracle heuristic.

[1] Our core system has a negligible correctness error; however, we outline how to build a perfectly correct system in Section 4.

Katz and Wang [24] gave an IBE system with a tight security reduction in the random oracle model using a two-key approach. One might view this as falling outside the partition approach, although their techniques do not appear to give a path to full security for HIBE and related problems.

Bellare and Ristenpart [2] introduced an interesting method recently to eliminate the artificial abort from the Waters IBE system; there method also falls into the partitioning category.

2 Background

We present a few facts related to groups with efficiently computable bilinear maps and then define the decisional Billinear-Diffie-Hellman and decisional Linear Assumptions. For space considerations, the definitions of security for Identity-Based Encryption and Hierarchical Identity-Based Encryption are included in Appendix A.

2.1 Bilinear Maps

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.2 Decisional Bilinear Diffie-Hellman Assumption

We define the decisional Bilinear Diffie-Hellman problem as follows. Choose a group $\mathbb{G}$ of prime order p, where the size of p is a function of the security parameters. Next, choose a random generator g and random exponents $c_1, c_2, c_3 \in \mathbb{Z}_p$. If an adversary is given

$$\vec{y} = g, g^{c_1}, g^{c_2}, g^{c_3},$$

it must remain hard to distinguish $e(g, g)^{c_1 c_2 c_3} \in \mathbb{G}_T$ from a random element in $\mathbb{G}_T$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0, 1\}$ has advantage ϵ in solving decisional BDH problem in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}\big(\vec{y}, T = e(g, g)^{c_1 c_2 c_3}\big) = 0\right] - \Pr\left[\mathcal{B}\big(\vec{y}, T = R\big) = 0\right] \right| \geq \epsilon \ .$$

Definition 1. *We say that the decisional BDH assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional BDH problem.*

2.3 Decisional Linear Assumption

We define the decisional Linear problem as follows. Choose a group $\mathbb{G}$ of prime order p, where the size of p is a function of the security paramters. Next, choose random generators g, f, ν and random exponents $c_1, c_2 \in \mathbb{Z}_p$. If an adversary is given

$$\vec{y} = g, f, \nu, g^{c_1}, f^{c_2},$$

it must remain hard to distinguish $\nu^{c_1+c_2} \in \mathbb{G}$ from a random element in $\mathbb{G}$.

An algorithm $\mathcal{B}$ that outputs $z \in \{0, 1\}$ has advantage ϵ in solving decisional Linear problem in $\mathbb{G}$ if

$$\left| \Pr\left[\mathcal{B}(\vec{y}, T = \nu^{c_1+c_2}) = 0\right] - \Pr\left[\mathcal{B}(\vec{y}, T = R) = 0\right] \right| \geq \epsilon \ .$$

Definition 2. *We say that the decisional Linear assumption holds if no polytime algorithm has a non-negligible advantage in solving the decisional Linear problem.*

3 Identity-Based Encryption

We now present our core Identity-Based Encryption construction along with our proof of its security under the the decisional Linear and decisional BDH assumptions.

We first give the four algorithms of our IBE system. Next, we describe two additional algorithms for the creation of semi-functional ciphertexts and private keys respectively. The purpose of these algorithms is to define the structure of semi-functional ciphertexts and keys for our proof of security. We emphasize that these algorithms are not used in the actual system; indeed it is crucial for our security argument that no attacker could create ciphertexts or keys of this form.

Finally, we give the proof of our system against an attacker that makes at most q private key queries[2]. We organize our proof as a sequence of games. In the sequence, we will gradually change the actual security game; first by introducing a semi-functional challenge ciphertext and then introduce semi-functional private keys one by one. We show that under the decisional Linear Assumption no adversary can distinguish between each successive game. Finally, we end up in a game where the challenge ciphertext and the *all* the private keys given out are semi-functional. At this point we can prove security under decisional-BDH.

Note to the reader: The techniques applied for this system were originally devised using composite order groups. Using composite order groups allows us to obtain a simpler exposition at the expense of using a somewhat more complex assumption. In Appendix E we provide a sketch of how to achieve security using composite order subgroups.

3.1 Construction

Setup(λ) The authority first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators $g, v, v_1, v_2, w, u, h \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the public parameters PK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b \cdot a_1},\ g^{b \cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b,\ w,\ u,\ h, e(g, g)^{\alpha \cdot a_1 \cdot b}.$$

[2]The maximum number of queries an attacker makes is, of course, a polynomial function $q(\cdot)$ of the security parameter; however, for notational simplicity we simply will speak of it making q private key queries.

The master secret key MSK consists of $g, g^{\alpha}, g^{\alpha \cdot a_1}, v, v_1, v_2$ as well as the public parameters. The identity space for the described scheme will be $\mathbb{Z}_p$, although we note in practice one can apply a collision resistant function to identities of arbitrary lengths.

Encrypt$(\mathrm{PK}, \mathcal{I}, M)$ The encryption algorithm chooses random s_1, s_2, t, and $\mathrm{tag}_c \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It then blinds $M \in \mathbb{G}_T$ as $C_0 = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}$ and creates:

$$C_1 = (g^b)^{s_1+s_2},\ C_2 = (g^{b \cdot a_1})^{s_1}, \quad C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b \cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},\ C_6 = \tau_1^{s_1}\tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1}(\tau_2^b)^{s_2} w^{-t},$$

$$E_1 = (u^{\mathcal{I}} w^{\mathrm{tag}_c} h)^t,\ E_2 = g^t.$$

The ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, E_2, \mathrm{tag}_c$.

KeyGen$(\mathrm{MSK}, \mathcal{I})$ The authority chooses random $r_1, r_2, z_1, z_2, \mathrm{tag}_k \in \mathbb{Z}_p$. Let $r = r_1 + r_2$.

Then it creates:

$$D_1 = g^{\alpha \cdot a_1} v^r. \quad D_2 = g^{-\alpha} v_1^r g^{z_1}. \quad D_3 = (g^b)^{-z_1}. \quad D_4 = v_2^r g^{z_2}, \quad D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2 \cdot b}, \quad D_7 = g^{r_1},$$

$$K = (u^{\mathcal{I}} w^{\mathrm{tag}_k} h)^{r_1}.$$

The secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \mathrm{tag}_k$.

Decrypt$(\mathrm{CT}, K_{\mathcal{I}})$ The decryption algorithm will be able to decrypt a ciphertext encrypted for $\mathcal{I}$ with private key $\mathrm{SK}_{\mathcal{I}}$ if the ciphertext tag_c is not equal to the private key tag_k. Since both tags are chosen randomly, decryption will succeed with all but a negligible $1/p$ probability.

We break the decryption algorithm into a set of calculations. First, it computes:

$$\begin{aligned} A_1 &= e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5) \\ &= e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(v, g)^{b(s_1+s_2)r} e(v_1, g)^{a_1 b s_1 r} e(v_2, g)^{a_2 b s_2 r}. \end{aligned}$$

Recall that $r = r_1 + r_2$. Next, it computes

$$\begin{aligned} A_2 &= e(C_6, D_6) \cdot e(C_7, D_7) \\ &= e(v, g)^{b(s_1+s_2)r} e(v_1, g)^{a_1 b s_1 r} e(v_2, g)^{a_2 b s_2 r} \cdot e(g, w)^{-r_1 t}. \end{aligned}$$

Taking, $A_3 = A_1/A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g, w)^{r_1 \cdot t}$ leaves us with one more cancellation to get the message blinding factor. If $\mathrm{tag}_c \neq \mathrm{tag}_k$ then the decryption algorithm can compute

$$A_4 = \big(e(E_1, D_7)/e(E_2, K)\big)^{1/(\mathrm{tag}_c - \mathrm{tag}_k)} = e(g, w)^{r_1 \cdot t}.$$

Finally, we can recover the message by computing

$$C_0/(A_3/A_4) = M.$$

Altogether, decryption requires nine applications of the pairing algorithm.

3.2 Semi-Functional Algorithms

We now describe the semi-functional ciphertext and key generation algorithms. We will define them as algorithms that are executed with knowledge of the secret exponents; however, in a real system they will not be used. Their main purpose is to define the structures that will be used in our proof. We define both semi-functional ciphertexts and keys in terms of a transformation on a normal ciphertext or key.

Semi-Functional Ciphertexts The algorithm first runs the encryption algorithm to generate a normal ciphertext CT for identity $\mathcal{I}$ and message M with $C'_1, \ldots, C'_7, E'_1, E'_2$. Then it chooses a random $x \in \mathbb{Z}_p$. It sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3, E_1 = E'_1, E_2 = E'_2$, leaving these elements and the tag_c unchanged. It then sets

$$C_4 = C'_4 \cdot g^{ba_2x}, \quad C_5 = C'_5 \cdot g^{a_2x}, \quad C_6 = C'_6 \cdot v_2^{a_2x}, \quad C_7 = C'_7 \cdot v_2^{a_2bx}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2, \text{tag}_c$.

Semi-Functional Secret Keys The algorithm first runs the encryption algorithm to generate a normal private key $\text{SK}_{\mathcal{I}}$ for identity $\mathcal{I}$ with $D'_1, \ldots, D'_7, K$. Then it chooses a random $\gamma \in \mathbb{Z}_p$. It sets $D_3 = D'_3, D_5 = D'_5, D_6 = D'_6, D_7 = D'_7, K = K'$, leaving these elements and the tag_k unchanged. It then sets

$$D_1 = D'_1 g^{-a_1a_2\gamma}, \quad D_2 = D'_2 \cdot g^{a_2\gamma}, \quad D_4 = D'_4 \cdot g^{a_1\gamma}$$

The semi-functional secret key is $\text{SK} = D_1, \ldots, D_7, K, \text{tag}_k$

Intuition We make a few remarks about the nature of the semi-functional keys and the structure of the system. First, we note that if one attempted to decrypt a semi-functional ciphertext with a normal key, then the decryption would succeed. This follows from the fact that

$$e(g^{ba_2x}, D_4)e(g^{a_2x}, D_5)/\big(e(v_2^{a_2x}, D_6)e(v_2^{a_2bx}, D_7)\big) = 1$$

when D_4, D_5, D_6, D_7 come from a normally generated ciphertext. One can view this as the extra "random" space occupied by the semi-functional part of the ciphertext as being orthogonal to the space defined by a normal key. For similar reasons, the semi-functional components of a private key will not impede decryption when applied on a normal ciphertext. However, when a semi-functional key is used to decrypt a semi-functional ciphertext decryption will fail (or end up giving a random message) because an extra $e(g, g)^{-a_1a_2x\gamma b}$ will be multiplied by the intended message.

We note that in order to generate semi-functional ciphertexts and private keys (according to the defined procedures) one respectively needs $v_2^{a_2b}$ and $g^{a_1a_2}$ — neither of which is available from the public parameters.

3.3 Proof of Security

We organize our proof as a sequence of games. The first game defined will be the real identity-based encryption game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next (under a complexity assumption). As stated before, the crux of our strategy is to move to a security game where both the challenge ciphertext and private keys are semi-functional. At this point any keys the challenger gives out are not useful in decrypting the ciphertext. We first define the games as:

$\mathbf{Game_{Real}}$: The actual IBE security game defined in Appendix A.

$\mathbf{Game}_i$: The real security game with the following two exceptions: 1) The challenge ciphertext will be a semi-functional ciphertext on the challenge identity $\mathcal{I}^*$. 2) The first i private key queries will return semi-functional private keys. The rest of the keys will be normal.

For an adversary that makes at most q queries we will be interested in $\mathbf{Game}_0, \ldots, \mathbf{Game}_q$. We note that in $\mathbf{Game}_0$ the challenge ciphertext is semi-functional, but all keys are normal and in $\mathbf{Game}_q$ all private keys are semi-functional.

$\mathbf{Game_{Final}}$: The real security game with the following exceptions: 1) The challenge ciphertext is a semi-functional encryption on a *random* group element of $\mathbb{G}_T$. 2) *All* of the private key queries result in semi-functional keys.

We now prove a set of Lemmas that argue about the distinguishablity of these games. For each proof we need to build a reduction simulator that both answers private key queries and creates a challenge ciphertext. We let $\mathbf{Game}_{\mathrm{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ denote an algorithm $\mathcal{A}$'s advantage in the real game.

Lemma 1. *Suppose that there exists an algorithm $\mathcal{A}$ where $\mathbf{Game}_{Real}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_0\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision Linear problem. We now describe how it executes the Setup, Key Phase, and Challenge phases of the IBE game with $\mathcal{A}$.

Setup The algorithm chooses random exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_p$ and random group elements $u, w, h \in \mathbb{G}$. It then sets $g = g, g^{a_1} = f, g^{a_2} = \nu$; intuitively a_1, a_2 are the exponents that the reduction cannot know itself.

Finally, it sets the variables as:

$$g^b,\ g^{b \cdot a_1} = f^b\ g^{b \cdot a_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

Using this it can calculate $\tau_1, \tau_2, \tau_1^b, \tau_2^b$ and $e(g,g)^{\alpha a_1 b} = e(g,f)^{\alpha \cdot b}$ in order to publish the public parameters PK. We also note that using α it can compute the master secret key for itself.

Key Generation Phases 1,2 Since $\mathcal{B}$ has the actual master secret key MSK it simply runs the key generation to generate the keys in both phases. Note that the MSK it has only allows for the creation of normal keys.

Challenge ciphertext $\mathcal{B}$ receives two messages M_0, M_1 and challenge identity $\mathcal{I}^*$. It then flips a coin β. We describe the creation of the challenge ciphertext in two steps. First, it creates a normal ciphertext using the real algorithm by calling $\mathrm{Encrypt}(\mathrm{PK}, \mathcal{I}^*, M_\beta)$, which outputs a ciphertext $\mathrm{CT} = C'_0, \ldots, C'_7, E'_1, E'_2, \mathrm{tag}_c$. Let s'_1, s'_2, t' be the random exponents used in creating the ciphertext.

Then we modify components of our ciphertext as follows. It sets

$$C_0 = C'_0 \cdot \big(e(g^{c_1}, f) \cdot e(g, f^{c_2})\big)^{b \cdot \alpha}, \quad C_1 = C'_1 \cdot (g^{c_1})^b, \quad C_2 = C'_2 \cdot (f^{c_2})^{-b}, \quad C_3 = C'_3 \cdot (f^{c_2}), \quad C_4 = C'_4 \cdot (T)^b,$$

$C_5 = C_5' \cdot T, \quad C_6 = C_6' \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}, \quad C_7 = C_7' \cdot \big((g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}\big)^b, \quad E_1 = E_1', \quad E_2 = E_2'.$

The returned ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, E_2, \mathrm{tag}_c$.

If T is a tuple, then this assignment implicitly sets $s_1 = -c_2 + s_1', s_2 = s_2' + c_1 + c_2$, and $s = s_1 + s_2 = c_1 + s_1' + s_2'$. If $T = \nu^{c_1+c_2}$ it will have the same distribution as a standard ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Lemma 2. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and $\mathbf{Game}_{\mathsf{k}-1}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{\mathsf{k}}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$ for some k where $1 \le k \le q$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision Linear problem. We now describe how it executes the Setup, Key Phase, and Challenge phases of the IBE game with $\mathcal{A}$.

Setup Algorithm $\mathcal{B}$ first chooses random exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_u, y_h$. It then defines

$$g = g, g^b = f, g^{b \cdot a_1} = f^{a_1},\ g^{b \cdot a_2} = f^{a_2},\ v = \nu^{-a_1 \cdot a_2},\ v_1 = \nu^{a_2} \cdot g^{y_{v_1}},\ v_2 = \nu^{a_1} \cdot g^{y_{v_2}},\ e(g,g)^{\alpha \cdot a_1 b} = e(f,g)^{\alpha \cdot a_1}.$$

Now it can create

$$\tau_1 = v v_1^{a_1} = g^{y_{v_1} a_1} \quad \tau_2 = v v_1^{a_2} = g^{y_{v_2} a_2} \quad \tau_1^b = v v_1^{a_1} = f^{y_{v_1} a_1} \quad \tau_2^b = v v_1^{a_2} = f^{y_{v_2} a_2}.$$

Finally, $\mathcal{B}$ chooses random $A, B \in \mathbb{Z}_p$. It then sets

$$w = f g^{y_w}, \quad u = f^{-A} g^{y_u}, \quad h = f^{-B} g^{y_h}.$$

This will define all the public parameters of the system. Note that by virtue of knowing α, the algorithm $\mathcal{B}$ will know the regular master secret key.

We highlight the importance of the function $F(\mathcal{I}) = A \cdot \mathcal{I} + B$. One important feature is that for $\mathrm{tag}_c = F(\mathcal{I})$ we have $(u^{\mathcal{I}} w^{\mathrm{tag}_c} h) = f^{\mathrm{tag}_c - A \cdot \mathcal{I} - B} g^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_c \cdot y_w} = g^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_c \cdot y_w}$. In this case $\mathcal{B}$ will know the discrete log base g of the function. We also note that A, B are initially information theoretically hidden from the adversary. Since it is a pairwise independent function, if the adversary is given $F(\mathcal{I})$ for some identity, the, all values in $\mathbb{Z}_p$ are equally likely for $F(\mathcal{I}')$ for some $\mathcal{I} \neq \mathcal{I}'$.

Key Gen Phases 1,2 We break the Key Generation into three cases. Key Generation is done the same regardless of whether we are in phase 1 or 2.

Consider the i-th query made by $\mathcal{A}$.

Case 1: $i > k$

When i is greater than k our algorithm $\mathcal{B}$ will generate a normal key for the requested identity $\mathcal{I}$. Since it has the master secret key MSK it can run that algorithm.

Case 2: $i < k$

When i is less than k our algorithm $\mathcal{B}$ will generate a semi-functional key for the requested identity $\mathcal{I}$. It first creates a normal key using MSK. Then it makes it semi-functional using the procedure from above in Subsection 3.2. It can run this procedure since it knows $g^{a_1 a_2}$.

Case 3: $i = k$

The algorithm first runs the key generation algorithm to generate a normal private key $\mathrm{SK}_{\mathcal{I}}$ for identity $\mathcal{I}$ with $D'_1, \ldots, D'_7, K$ using $\mathrm{tag}_k{}^* = F(\mathcal{I})$. Let r'_1, r'_2, z'_1, z'_2 be the random exponents used.

It then sets

$$D_1 = D'_1 \cdot T^{-a_1 \cdot a_2},\ D_2 = D'_2 \cdot T^{a_2}(g^{c_1})^{y_{v_1}},\ D_3 = D'_3 \cdot (f^{c_2})^{y_{v_1}},\ D_4 = D'_4 \cdot T^{a_1}(g^{c_1})^{y_{v_2}},$$

$$D_5 = D'_5 \cdot (f^{c_2})^{y_{v_2}},\ D_6 = D'_6 \cdot f^{c_2},\ D_7 = D'_7 \cdot (g^{c_1}),\ K = K' \cdot (g^{c_1})^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_k \cdot y_w}.$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \mathrm{tag}_k$. We emphasize that the fact that $\mathrm{tag}_k = F(\mathcal{I})$ allowed us to created the component K. In addition, we note that we implicitly set $z_1 = z'_1 - y_{v_1} c_2$ and $z_2 = z'_2 - y_{v_2} c_2$ in order to be able to create D_2 and D_4.

If T is a Linear tuple of the form $T = \nu^{c_1 + c_2}$, then the k-th query results in a normal key under randomness $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. Otherwise, if T is a random group element, then we can write $T = \nu^{c_1 + c_2} g^{\gamma}$ for random $\gamma \in \mathbb{Z}_p$. This forms a semi-functional key where γ is the added randomness to make it semi-functional.

Challenge Ciphertext Algorithm $\mathcal{B}$ is given a challenge identity $\mathcal{I}^*$ and messages M_0, M_1. Then it flips a coin β.

In this phase $\mathcal{B}$ needs to be able to generate a semi-functional challenge ciphertext. One problem is that $\mathcal{B}$ does not have the group element v_2^b so it cannot directly create such a ciphertext. However, in the case where $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$ it will have a different method of doing so.

$\mathcal{B}$ first runs the normal encryption algorithm to generate a normal ciphertext CT for identity $\mathcal{I}^*$ and message M^*; during this run it uses $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$. It then gets a standard ciphertext $C'_1, \ldots, C'_7, E'_1, E'_2$ under random exponents s'_1, s'_2, t'

To make it semi-functional it chooses a random $x \in \mathbb{Z}_p$. It first sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$ leaving these elements and the $\mathrm{tag}_c{}^*$ unchanged. It then sets

$$C_4 = C'_4 \cdot f^{a_2 \cdot x}, \quad C_5 = C'_5 \cdot g^{a_2 \cdot x}, \quad C_6 = C'_6 \cdot v_2^{a_2 x}, \quad C_7 = C'_7 \cdot f^{y_{v_2} \cdot x \cdot a_2} \nu^{-a_1 \cdot x \cdot y_w \cdot a_2},$$

$$E_1 = E'_1 \cdot (\nu^{\mathcal{I} \cdot y_u + y_h + \mathrm{tag}_c \cdot y_w})^{a_1 a_2 x} \quad E_2 = E'_2 \cdot \nu^{a_1 a_2 \cdot x}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2, \mathrm{tag}_c$.

Intuitively, the algorithm implicitly sets $g^t = g^{t'} + \nu^{a_1 a_2 x}$. This allows for the cancellation of the term $v_2^{a_1 a_2 b x}$ by w^{-t} in constructing C_7. Normally, this would be problematic for the generation of E_1; however since $\mathrm{tag}_c{}^* = F(\mathcal{I}^*)$ $\mathcal{B}$ is able to create this term.

If T is a tuple, then we are in $\mathbf{Game}_{k-1}$, otherwise we are in $\mathbf{Game}_k$. We highlight that the adversary cannot detect any special relationship between $\mathrm{tag}_c{}^*$ and $\mathrm{tag}_k{}^*$ since $F(\mathcal{I}) = A \cdot \mathcal{I} + B$ is a pairwise independent function and A, B are hidden from its view.

$\mathcal{B}$ receives a bit β' and outputs 0 if $\beta = \beta'$.

Lemma 3. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and* $\mathbf{Game}_{\mathsf{q}}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{Final}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. *Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision BDH game.*

Proof. We give the proof of security in Appendix B.

Theorem 1. *If the decisional Linear and decisional BDH assumptions hold then no poly-time algorithm can break our IBE system.*

Proof. Any attacker's advantage in $\mathbf{Game}_{\text{Final}}\,\mathsf{Adv}_{\mathcal{A}}$ in the final game must be 0 since it completely hides the bit β. By the sequence of games we established and Lemmas 1,2,3 an attacker's advantage in the real game $\mathbf{Game}_{\text{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ must be negligibly close to 0.

4 Discussion

In this section we discuss a few potential future variations and implications of our IBE system.

Achieving Perfect Correctness Although having a negligible correctness error seems acceptable in practice, we would like to point out that we can close this gap by simply giving any user two private keys for an identity $\mathcal{I}$ each time they make a key request. The authority will simply run the original key generation algorithm twice with the restriction that the two key tags, $\text{tag}_{kA}, \text{tag}_{kB}$ are not equal. When attempting to decrypt a ciphertext at least one of the keys will work. The proof of security will work over each key piece — that is, each key request in the modified system will generate two distinct key requests (for the same identity) in the proof. We could also use a complementary two ciphertext approach and one private key approach.

Another potential solution is to run an efficient selectively secure IBE scheme [3] "in parallel". When a user encrypts a message M to $\mathcal{I}$ with tag_c in our original system, he will also encrypt M to the "identity" tag_c in the second selective system. A user with a key with tag_k will get a private key for "identity" tag_k in the second system. On decryption with $1-1/p$ probability the decryption algorithm will use the first ciphertext. However, if the tags align it can use the second ciphertext.

Signature Scheme Naor [3] observed that any (fully secure) IBE system gives rise to a signature scheme secure under the same assumptions. The signature system from our IBE scheme has the favorable properties that the public parameters and signatures are a constant number of group elements, it is provable in the standard model, and it is stateless. While some previous signature schemes derived from IBE systems (e.g. BLS [11] or Waters [29] signatures) depended on the computational variants of the assumptions, our proof technique seems to require the decisional Linear Assumption. One interesting approach would be to see if one could create shorter signatures than those generated in the generic conversion by using the IBE systems private keys.

Chosen Ciphertext Security We note that using the transformation of Canetti, Halevi, and Katz [15] we can achieve chosen ciphertext security from the HIBE scheme of Section 5.

Security under the SXDH Assumption One factor in the size and complexity of our IBE system is that it relies upon the Decisional Linear Assumption to hide the form of both keys and ciphertexts. One potential alternative is to use asymmetric bilinear groups, where we have $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. Using these group we might assume DDH is hard both within $\mathbb{G}_1$ and within $\mathbb{G}_2$; this has also been called the XDH assumption. Using this assumption we might hope to shave off three group elements from both ciphertexts and private keys.

[3] The observation was documented by Boneh and Franklin [7].

Alternative to Incompleteness An critical part to arguing security is that an attacker could not distinguish normal keys from semi-functional ones. Our approach was to use a hybrid argument where for the key in question its $\text{tag}_k = F(\mathcal{I})$. If the simulator attempted to create the key in question for $\mathcal{I}^*$ and test it on the challenge ciphertext this would not work since $\text{tag}_k = \text{tag}_c$. Intuitively, the simulator could not test whether the key was semi-functional since decryption would fail *regardless* of whether the key was semi-functional or not. One might consider taking the opposite approach where decryption would always succeed if $\text{tag}_c = \text{tag}_k$ even if both the key and ciphertext are semi-functional. We note this approach would also require a slightly different proof strategy for proving Lemma 3.

5 Hierarchical Identity-Based Encryption

In this section we present our Hierarchical Identity-Based Encryption system. Our construction will build on top of our IBE scheme of Section 3. The reader will notice that the added complexity of moving from our IBE to HIBE system is remarkably simple. The same core concepts of our construction and proof methodology apply. One might view the HIBE system as "combining" the structure of the Boneh-Boyen [3] HIBE system with our techniques to get full security.

One challenging aspect in the proof of security is that a private key of depth d will have associated tags: $\text{tag}_{k1}, \ldots, \text{tag}_{kd}$. If we run our delegation algorithm to create a new key of depth $d+1$, the new key will inherit the previous key's tag values and there is no method for "re-randomizing" them. Most prior security definitions of HIBE [23, 21] define a game where all keys come from an authority and don't model any distinctions on how a key was created (i.e. trace paths of delegation). The prior definitions are only valid if keys from the delegation algorithm are distributed identically to a fresh call to the key generation algorithm [4]; however, due to the "tag lineage" described this is clearly not the case. To argue security we use a "complete" model of HIBE security introduced by Shi and Waters [28] that we define in appendix A. Due to space considerations our proof of security is in Appendix C.

5.1 Construction

In our system we will consider a hierarchical identity as an identity vector $\vec{\mathcal{I}} = \mathcal{I}_1 : \cdots : \mathcal{I}_d$ for some depth d, where $d \leq n$ for some maximum depth n. We assume that the identities are encoded such that for two identities $\vec{\mathcal{I}}, \vec{\mathcal{I}}'$ if $\mathcal{I}_i = \mathcal{I}'_i$ then $\mathcal{I}_j = \mathcal{I}'_j$ for all $j \leq i$. We can enforce this by encoding all previous levels. For example, an identity of level one "com" and level two "yahoo" can be encoded as "com":"com.yahoo", where '.' is a special symbol. In practice, one will use a collision resistant hash function to hash identities of arbitrary length to $\mathbb{Z}_p$.

Setup(λ, n) The setup algorithm takes as input a security parameter and the maximum depth n. The authority first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators g, v, v_1, v_2, w, $u_1, \ldots, u_n, h_1, \ldots, h_n \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the public parameters PK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b \cdot a_1},\ g^{b \cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b, v,\ v_1,\ v_2,\ \ w,\ u_1, \ldots, u_n,\ h_1, \ldots, h_n, e(g,g)^{\alpha \cdot a_1 \cdot b}.$$

[4] This is actually the case for most prior systems, so the proofs of security do hold up.

The master secret key MSK consists of $g, g^{\alpha}, g^{\alpha \cdot a_1}$ as well as the public parameters. The identity space for the described scheme will be $\mathbb{Z}_p$.

Encrypt$(\mathrm{PK}, \vec{\mathcal{I}} = \mathcal{I}_1 : \cdots : \mathcal{I}_d, M)$ The encryption algorithm will encrypt to an identity vector of depth $d \leq n$. It chooses random $s_1, s_2, t \in \mathbb{Z}_p$ and $\mathrm{tag}_{c1}, \ldots, \mathrm{tag}_{cd} \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It then blinds $M \in \mathbb{G}_T$ as $C_0 = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}$ and creates:

$$C_1 = (g^b)^{s_1+s_2},\ C_2 = (g^{b \cdot a_1})^{s_1},\ C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b \cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},\ C_6 = \tau_1^{s_1}\tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1}(\tau_2^b)^{s_2} w^{-t},$$

$$E_1 = (u_1^{\mathcal{I}_1} w^{\mathrm{tag}_{c1}} h_1)^t, \ldots, E_d = (u_d^{\mathcal{I}_d} w^{\mathrm{tag}_{cd}} h_d)^t, \quad \tilde{E} = g^t.$$

The ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, , \ldots, E_d, \tilde{E}, \mathrm{tag}_{c1}, \ldots, \mathrm{tag}_{kd}$.

KeyGen$(\mathrm{MSK}, \vec{\mathcal{I}} = \mathcal{I}_1 : \cdots : \mathcal{I}_d)$ The authority chooses random $\mu_1, \ldots, \mu_d, r_2, z_1, z_2, \mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd} \in \mathbb{Z}_p$. First let $r_1 = \sum_{1 \leq i \leq d} \mu_i$ and then let $r = r_1 + r_2$. Then it creates:

$$D_1 = g^{\alpha \cdot a_1} v^r, \quad D_2 = g^{-\alpha} v_1^r g^{z_1}, \quad D_3 = (g^b)^{-z_1}, \quad D_4 = v_2^r g^{z_2}, \quad D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2 \cdot b}, \quad D_7 = g^{r_1},$$

$$(K_{1,1} = (u_1^{\mathcal{I}_1} w^{\mathrm{tag}_{k1}} h_1)^{\mu_1},\ K_{1,2} = g^{\mu_1}.), \ldots, (K_{d,1} = (u_d^{\mathcal{I}_d} w^{\mathrm{tag}_{kd}} h_d)^{\mu_d},\ K_{d,2} = g^{\mu_d})$$

The secret key is $\mathrm{SK} = D_1, \ldots, D_7, (K_{1,1}, K_{1,2}), \ldots, (K_{d,1}, K_{d,2}) \mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd}$.

Delegate$(\mathrm{PK}, \mathrm{SK}_{\vec{\mathcal{I}} = \mathcal{I}_1 : \cdots : \mathcal{I}_d}, \mathcal{I}_{d+1})$ The algorithm will take a secret key $\mathrm{SK} = D_1', \ldots, D_7'$, $(K_{1,1}', K_{1,2}')$, $\ldots$, $(K_{d,1}', K_{d,2}')$, $\mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd}$ for $\vec{\mathcal{I}}$ and extend it to depth $d+1$ by creating a key for $\vec{\mathcal{I}} : \mathcal{I}_{d+1}$.

The algorithm will "re-randomize" the existing key in the process of appending on a new key component; however, the existing tag_k values will remain. It chooses random $\mu_1, \ldots, \mu_{d+1}, r_2, z_1, z_2, \mathrm{tag}_{kd+1} \in \mathbb{Z}_p$. First let $r_1 = \sum_{1 \leq i \leq d+1} \mu_i$ and then let $r = r_1 + r_2$. Then it creates:

$$D_1 = D_1' \cdot v^r, \quad D_2 = D_2' \cdot v_1^r g^{z_1}, \quad D_3 = D_3' \cdot (g^b)^{-z_1}, \quad D_4 = D_4' \cdot v_2^r g^{z_2},$$

$$D_5 = D_5' \cdot (g^b)^{-z_2}, \quad D_6 = D_6' \cdot g^{r_2 \cdot b}, \quad D_7 = D_7' \cdot g^{r_1},$$

$$K_{1,1} = K_{1,1}' \cdot (u_1^{\mathcal{I}_1} w^{\mathrm{tag}_{k1}} h_1)^{\mu_1}, \ldots, K_{d,1} = K_{d,1}' \cdot (u_d^{\mathcal{I}_d} w^{\mathrm{tag}_{kd}} h_d)^{\mu_d}, \quad K_{d+1,1} = (u_{d+1}^{\mathcal{I}_{d+1}} w^{\mathrm{tag}_{kd+1}} h_{d+1})^{\mu_{d+1}},$$

$$K_{1,2} = K_{1,2}' \cdot g^{\mu_1}, \ldots, K_{d,2} = K_{d,2}' \cdot g^{\mu_d}, \quad K_{d+1,2} = g^{\mu_{d+1}}.$$

The secret key is $\mathrm{SK} = D_1, \ldots, D_7, (K_{1,1}, K_{1,2}), \ldots, (K_{d+1,1}, K_{d+1,2}), \mathrm{tag}_{k1}, \ldots, \mathrm{tag}_{kd+1}$.

Decrypt$(\mathrm{CT}, K_{\mathcal{I}})$ The decryption algorithm will be able to decrypt a ciphertext encrypted for $\vec{\mathcal{I}}'$ of depth d' with private key $\mathrm{SK}_{\vec{\mathcal{I}}}$ of depth d if 1) $\forall i \leq d : \ \vec{\mathcal{I}}_i' = \vec{\mathcal{I}}_i$ for all $i \leq d$ and 2) $\forall i \leq d : \ \mathrm{tag}_{ci} \neq \mathrm{tag}_{ki}$. We break the decryption algorithm into a set of calculations: First, it computes:

$$A_1 = e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5)$$

$$A_2 = e(C_6, D_6) \cdot e(C_7, D_7) \quad A_3 = A_1 / A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g,w)^{r_1 \cdot t}.$$

If $\forall i \leq d$ we have $\mathrm{tag}_{ci} \neq \mathrm{tag}_{ki}$ then the decryption algorithm can compute

$$A_4 = \left(e(E_1, K_{1,2}) / e(\tilde{E}, K_{1,1})\right)^{1/(\mathrm{tag}_{c1} - \mathrm{tag}_{k1})} \cdots \left(e(E_d, K_{d,2}) / e(\tilde{E}, K_{d,1})\right)^{1/(\mathrm{tag}_{cd} - \mathrm{tag}_{kd})} = e(g,w)^{t \sum_{1 \leq d} \mu_i}$$

Finally, we can recover the message by computing $C_0 / (A_3 / A_4) = M$.

6 A Signature Scheme

In this section we describe the signature scheme that falls out of our IBE system. [5] Our signature scheme has the desirable combination of properties that both signatures and public keys consist of a constant number of group elements and that it is provably secure under simple assumptions. We begin by describing the construction and then enter into a discussion.

6.1 Construction

Setup(λ) The setup algorithm first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators $g, v, v_1, v_2, w, u, h \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the verification key parameters VK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b \cdot a_1},\ g^{b \cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b,\ w,\ u,\ h, e(g,g)^{\alpha \cdot a_1 \cdot b}.$$

The signing key SK consists of $g, g^\alpha, g^{\alpha \cdot a_1}, v, v_1, v_2$ as well as the public parameters. The message space for the described scheme will be $\mathbb{Z}_p$. Again, we note that in practice one can apply a collision resistant function to messages of arbitrary lengths.

Sign(SK, M) The signature algorithm is virtually identical to the private key generation algorithm of Section 3. The signature algorithm chooses random $r_1, r_2, z_1, z_2, \text{tag}_k \in \mathbb{Z}_p$. Let $r = r_1 + r_2$.

Then it creates:

$$\sigma_1 = g^{\alpha \cdot a_1} v^r. \quad \sigma_2 = g^{-\alpha} v_1^r g^{z_1}. \quad \sigma_3 = (g^b)^{-z_1}. \quad \sigma_4 = v_2^r g^{z_2}, \quad \sigma_5 = (g^b)^{-z_2}, \quad \sigma_6 = g^{r_2 \cdot b}, \quad \sigma_7 = g^{r_1},$$

$$\sigma_K = (u^M w^{\text{tag}_k} h)^{r_1}.$$

The signature is $\sigma = (\sigma_1, \ldots, \sigma_7, \sigma_K, \text{tag}_k)$.

Verify(VK, σ, M) The verification algorithm has two main steps. First, it creates a ciphertext using the IBE encryption algorithm in key encapsulation mode. Then it attempts to decrypt (or decapsulate) the ciphertext it created with the signature given and test to see if it produces the same key that was encapsulated. We point out that this verification algorithm follows Naor's method much more closely than other signature schemes derived from IBE systems.

The verification algorithm first chooses random s_1, s_2, t, and $\text{tag}_c \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It creates

$$C_1 = (g^b)^{s_1 + s_2},\ C_2 = (g^{b \cdot a_1})^{s_1}, \quad C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b \cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},\ C_6 = \tau_1^{s_1} \tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1} (\tau_2^b)^{s_2} w^{-t},$$

$$E_1 = (u^M w^{\text{tag}_c} h)^t,\ E_2 = g^t.$$

The verification algorithm then stores these variables as a temporary ciphertext and moves on to the next phase. We break the next phase into a set of calculations. First, it computes:

$$A_1 \quad = \quad e(C_1, \sigma_1) \cdot e(C_2, \sigma_2) \cdot e(C_3, \sigma_3) \cdot e(C_4, \sigma_4) \cdot e(C_5, \sigma_5)$$

[5] Naor observed that any IBE system can be converted into a signature scheme with the IBE private keys playing the role of signatures. The observation of Naor appeared in the work of Boneh and Franklin [8].

Next, it computes

$$A_2 \quad = \quad e(C_6, \sigma_6) \cdot e(C_7, \sigma_7)$$

Then let $A_3 = A_1/A_2$. Then, if $\text{tag}_c \neq \text{tag}_k$ then the verification algorithm computes

$$A_4 = \big(e(E_1, \sigma_7)/e(E_2, K)\big)^{1/(\text{tag}_c - \text{tag}_k)}$$

Finally, we can test if

$$(e(g,g)^{\alpha a_1 \cdot b})^{s_2} \stackrel{?}{=} A_3/A_4.$$

If the last equation holds then the verification output is "accept"; otherwise, it outputs "reject".

6.2 Discussion

The proof security of the signature system will fall directly the security of the IBE system, since the verification algorithm follows Naor's method of encrypting a random message and then decrypting with the signature. Since the message space is exponentially large the verification algorithm only needs to execute one instance of the encrypt then decrypt routine.

We make a few observations about this signature scheme. First, it is interesting to consider how a proof for this signature system would be structured if we made one directly. We can think of the system as potentially producing two types of signatures. "Type A" signatures will be those given in the algorithm above, which correspond to normal private keys. There is also a second "Type B" signatures. These are formed by first taking taking a Type A signature $\sigma' = \sigma'_1, \ldots, \sigma'_K$ and a random γ and computing

$$\sigma_1 = \sigma'_1 g^{-a_1 a_2 \gamma}, \quad \sigma_2 = \sigma'_2 \cdot g^{a_2 \gamma}, \quad \sigma_4 = \sigma'_4 \cdot g^{a_1 \gamma}.$$

The other signature components are simply set as $\sigma_3 = \sigma'_3, \sigma_5 = \sigma'_5, \sigma_6 = \sigma'_6, \sigma_7 = \sigma'_7, \sigma_K = \sigma'_K$.

The proof roughly works as follows. In the first game the challenger will only output Type A signatures. We consider two cases at this stage. In the first case, if the attacker outputs a Type B forgery with non-neglgible probability, then reduction algorithm will use this to break an underlying assumption. For this particular system, the reduction would extract the triplet $g^{-a_1 a_2 \gamma}, g^{a_2 \gamma}, g^{a_1 \gamma}$ from the Type B signature, which can be used to solve an instance of the decisional linear problem. Consider the decision linear instance: given $g, g^{a_1}, g^{a_2}, g^x, g^{a_1 \cdot y}$ decide between $g^{a_2(x+y)}$ and a random group element in $\mathbb{G}$. The triplet above (for non-zero γ) can be used to solve this problem.

In other case, the attacker only produces Type A forgeries. In this case, the reduction algorithm will change each signature given out from Type A to Type B. The change will occur one step at a time. By the linear assumption we will argue that the attack will continue to only produce Type A signatures for its forgery. The key to this argument is the signature component σ_K and the tag tag_k. In this proof, the reduction algorithm will embed two values $A, B \in \mathbb{Z}_p$ such that it can tell whether a signature is Type A or B if and only if $\text{tag}_k \neq A \cdot M + B$. In this manner, the reduction can embed the "challenge signature". At step i in the hybrid argument it won't know if the signature is of type A or B. However, it can observe the type of forgery the attacker gives. The attacker cannot switch to making Type B forgeries; otherwise, this will indicate whether signature i is type A or B. Finally, we arrive at a step where the challenger outputs all Type B signatures and the attacker outputs a Type A forgery. We end using this to solve some other cryptographic

System	Public Key Size	Signature Size	Assumption	Standard Model
BLS [11]	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Computational DH	NO
Boneh-Boyen [4]	$\mathcal{O}(1)$	$\mathcal{O}(1)$	q-Strong DH	YES
Waters [29]	$\mathcal{O}(\lambda)$	$\mathcal{O}(1)$	Computational DH	YES
This work	$\mathcal{O}(1)$	$\mathcal{O}(1)$	Decision Linear	YES

Table 1: Comparison of signature systems in terms of public key size, signature size, and assumption used. Public key size is given in terms of group elements and λ denotes the security parameter.

assumption. In the case of this signature scheme, a reduction algorithm can extract g^{α} given only $e(g,g)^{\alpha}$ — breaking the computational Diffie-Hellman problem.

This methodology for producing signature schemes can apply outside the context of IBE. In general, one can encrypt either a secret A or secret B and give a non-interactive zero knowledge proof that one of the two secrets is encrypted. Note that the real scheme could start with either giving Type A or Type B signatures then moving the hybrid proof in the corresponding direction.

Comparisons to Other Signature Schemes It is interesting to compare our signature scheme with other ones that have sprung out of the IBE literature. For instance, BLS [11] signatures are private keys in the Boneh-Franklin scheme, Boneh and Boyen [4] signatures are private keys in Gentry's [18] IBE, and Waters [29] gave an IBE and signature scheme together.

A comparison of the main differentiating features is summarized in Table 6.2. Our work achieves security under the decision linear assumption, which is arguably a simpler assumption than the q-Strong Diffie-Hellman of Boneh and Boyen. However, it should be noted that these assumptions are not strictly comparable. In addition, we have a shorter public key than the Waters [29] signature scheme, where the number of group elements in the public key is linear in the security parameter.

Given these combination of these properties, it might be interesting to explore the application of our signature scheme in the context of other bilinear map protocols such as aggregate signatures, range proofs, e-cash, etc. Perhaps the biggest drawback is that while signature sizes are a constant number of group elements and verification takes a constant number of pairings, these constants are somewhat larger than in other systems. In addition, the proof of security is considerably more complex. When applying this signature in the context of a larger security protocol, it will likely be desirable to reduce security directly to it.

References

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *CRYPTO*, pages 205–222, 2005.

[2] Mihir Bellare and Thomas Ristenpart. Simulation without the artificial abort: Simplified proof and improved concrete security for waters' ibe scheme. In *EUROCRYPT*, pages 407–424, 2009.

[3] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[4] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In *EUROCRYPT*, pages 223–238, 2004.

[5] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[6] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

[7] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003.

[9] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[10] Dan Boneh, Craig Gentry, and Brent Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *CRYPTO*, pages 258–275, 2005.

[11] Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the weil pairing. In *ASIACRYPT*, pages 514–532, 2001.

[12] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[13] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In *CRYPTO*, pages 290–307, 2006.

[14] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[15] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In *EUROCRYPT*, pages 207–222, 2004.

[16] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[17] Amos Fiat and Moni Naor. Broadcast encryption. In *CRYPTO*, pages 480–491, 1993.

[18] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.

[19] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In *TCC*, 2009.

[20] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

[21] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[22] Craig Gentry and Brent Waters. Adaptive security in broadcast encryption systems. In *Eurocrypt*, 2009.

[23] Jeremy Horwitz and Ben Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.

[24] Jonathan Katz and Nan Wang. Efficiency improvements for signature schemes with tight security reductions. In *ACM Conference on Computer and Communications Security*, pages 155–164, 2003.

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

[26] Amit Sahai and Brent Waters. Revocation systems with very small private keys. Cryptology ePrint Archive, Report 2008/309, 2008.

[27] Elaine Shi, John Bethencourt, Hubert T.-H. Chan, Dawn Xiaodong Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

[28] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *ICALP (2)*, pages 560–578, 2008.

[29] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

A Definitions

We define the system requirements and security definitions for Identity-Based Encryption and Hierarchical IBE. We emphasize that our definitions are in the full model — i.e. are note selective-ID. In addition, we use a complete model of HIBE where that reflects where the private keys that an adversary requests come from.

A.1 Identity-Based Encryption

An identity-based based encryption scheme consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

Setup. The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master secret key MSK.

Encrypt$(\text{PK}, M, \mathcal{I})$**.** The encryption algorithm takes as input the public parameters PK, a message M, and an identity $\mathcal{I}$. The algorithm outputs a ciphertext CT.

Key Generation$(\text{MSK}, \mathcal{I})$**.** The key generation algorithm takes as input the master secret key MSK and an identity $\mathcal{I}$. It outputs a private key SK.

Decrypt$(\mathrm{PK}, \mathrm{CT}, \mathrm{SK})$**.** The decryption algorithm takes as input the public parameters PK, a ciphertext CT, and a secret key. If the ciphertext was an encryption to $\mathcal{I}$ and the secret key was the output of a key generation for the same identity then the algorithm will output the encrypted message M.

Security Definition for IBE

Setup. The challenger runs the Setup algorithm and gives the public parameters PK to the adversary.

Phase 1. The adversary makes repeated private key queries for any identity $\mathcal{I}$ of its choice.

Challenge. The adversary submits two equal length messages M_0 and M_1 and a challenge identity $\mathcal{I}^*$ with the restriction that $\mathcal{I}^*$ is not equal to any identity requested in the previous phase. The challenger then flips a random coin β, and encrypts M_β under $\mathcal{I}^*$. The resulting ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that an identity requested $\mathcal{I} \neq \mathcal{I}^*$.

Guess. The adversary outputs a guess β' of β.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[\beta' = \beta] - \frac{1}{2}$. We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 1. *An identity-based encryption scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.*

A.2 Hierarchical Identity-Based Encryption

A Hierarchical based encryption scheme consists of five algorithms: Setup, Encrypt, KeyGen, Decrypt, and Delegate.

Setup. The setup algorithm takes no input other than the implicit security parameter. It outputs the public parameters PK and a master secret key MSK.

Encrypt$(\mathrm{PK}, M, \vec{\mathcal{I}})$**.** The encryption algorithm takes as input the public parameters PK, a message M, and an identity $\vec{\mathcal{I}}$. The algorithm outputs a ciphertext CT.

Key Generation$(\mathrm{MSK}, \vec{\mathcal{I}})$**.** The key generation algorithm takes as input the master secret key MSK and an identity vector $\mathcal{I}$. It outputs a private key $\mathrm{SK}_{\vec{\mathcal{I}}}$.

Decrypt$(\mathrm{PK}, \mathrm{CT}, \mathrm{SK})$**.** The decryption algorithm takes as input the public parameters PK, a ciphertext CT, and a secret key. If the ciphertext was an encryption to $\vec{\mathcal{I}}$ and the secret key is for $\vec{\mathcal{I}}'$, where $\vec{\mathcal{I}}'$ is a prefix of $\vec{\mathcal{I}}$ identity then the algorithm will output the encrypted message M.

Delegate$(\mathrm{PK}, \mathrm{SK}_{\vec{\mathcal{I}}}, \mathcal{I})$ The delegate algorithm takes in a HIBE secret key for an identity vector $\vec{\mathcal{I}}$ of depth d and an identity $\mathcal{I}$. It returns a secret key for the $d+1$ depth identity $\vec{\mathcal{I}} : \mathcal{I}$.

Security Definition for HIBE We now give the security definition for HIBE. Our definition is a complete one in that when an attacker requests a key the model keeps track of which other key it was derived from. Prior models made no distinction about how keys were generated (e.g., whether they came straight from the authority or from a delegation operation of another key).

Setup. The challenger runs the Setup algorithm and gives the public parameters PK to the adversary.

The challenger will also intialize a set $S = \emptyset$, which will be the set of private keys it has created, but not given out.

Phase 1. The adversary makes repeated queries of one of three types:

Create The attacker gives the challenger an identity-vector $\vec{\mathcal{I}}$. The challenger creates a key for that vector, but does not give it to the adversary. It instead adds the key to the set S and gives the attacker a reference to it.

Delegate The attacker specifies a key $SK_{\vec{\mathcal{I}}}$ in the set S for an identity $\vec{\mathcal{I}}$, then it gives the challenger an identity $\mathcal{I}'$. The challenger runs the Delegate$(\mathrm{PK}, \mathrm{SK}_{\vec{\mathcal{I}}}, \mathcal{I}')$ algorithm to get a new secret key $\mathrm{SK}_{\vec{\mathcal{I}}:\mathcal{I}'}$ and adds this to the set S.

Reveal The attacker specifies an element of the set S for a secret key SK. The challenger removes the item from the set S and gives the attacker the secret key. We note at this point there is no need for the challenger to allow more delegate queries on the key since the attacker can run them itself.

Challenge. The adversary submits two equal length messages M_0 and M_1 and a challenge identity vector $\vec{\mathcal{I}}^*$ with the restriction that each identity vector $\vec{\mathcal{I}}$ given out in the key phase must not be a prefix of $\vec{\mathcal{I}}^*$. The challenger then flips a random coin β, and encrypts M_β under $\vec{\mathcal{I}}^*$. The resulting ciphertext CT^* is given to the adversary.

Phase 2. Phase 1 is repeated with the restriction that any *revealed* identity vector $\vec{\mathcal{I}}$ is not a prefix of $\vec{\mathcal{I}}^*$.

Guess. The adversary outputs a guess β' of β.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[\beta' = \beta] - \frac{1}{2}$. We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 2. *An hierarchical identity-based encryption scheme is secure if all polynomial time adversaries have at most a negligible advantage in the above game.*

B Proof of Lemma 3

Lemma 3 *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and* $\mathbf{Game}_{\mathsf{q}}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{Final}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. *Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision BDH game.*

Proof. We begin by noting that in both of these two games the challenge ciphertexts and all the private keys are semi-functional. Therefore, $\mathcal{B}$ only needs to be able to generate semi-functional private keys.

$\mathcal{B}$ begins by taking in a BDH instance $g, g^{c_1}, g^{c_2}, g^{c_3}, T$.

Setup The algorithm begins by choosing random exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_h, y_u \in \mathbb{Z}_p$. It then sets

$$g = g,\ g^b,\ g^{a_1},\ g^{a_2} = g^{c_2},\ g^{b \cdot a_1},\ g^{b \cdot a_2} = (g^{c_2})^b,$$

$$v = g^{y_v},\ v_1 = g^{y_{v_1}},\ v_2 = g^{y_{v_2}},\ w = g^{y_w},\ u = g^{y_u},\ h = g^{y_h},\ e(g,g)^{a_1 \alpha b} = e(g^{c_1}, g^{c_2})^{a_1 \cdot b}.$$

These parameters allow us to compute $\tau_1 = v v_1^{a_1}, \tau_1^b$; and $\tau_2 = v(g^{c_2})^{y_{v_2}}, \tau_2^b$ and publish the public key PK. Note that the master secret key g^α is not available to $\mathcal{B}$.

We point out that this setup lets $\alpha = c_1 \cdot c_2$ and $a_2 = c_2$.

Key Generation Phase 1,2 All key generations result in semi-functional keys. When a request for $\mathcal{I}$ is made, the key generation algorithm chooses random $r_1, r_2, z_1, z_2, \gamma', \mathrm{tag}_k \in \mathbb{Z}_p$ and defines $r = r_1 + r_2$. It aims to implicitly set the variable $\gamma = c_1 + \gamma'$.

It creates the key as:

$$D_1 = (g^{c_2})^{-\gamma' \cdot a_1} v^r, \quad D_2 = (g^{c_2})^{\gamma'} v_1^r g^{z_1}, \quad D_3 = (g^b)^{-z_1}, \quad D_4 = (g^{c_1})^{a_1} g^{a_1 \cdot \gamma'} v_2^r g^{z_2},$$

$$D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2 \cdot b}, \quad D_7 = g^{r_1}, \quad K = (u^{\mathcal{I}} w^{\mathrm{tag}_k} h)^{r_1}.$$

Challenge Ciphertext $\mathcal{B}$ receives a challenge identity $\vec{\mathcal{I}}^*$ and two message M_0, M_1 from the attacker. $\mathcal{B}$ will now create a challenge ciphertext that is a semi-functional ciphertext of either M_β or a random message, depending on T. It first chooses a random bit β.

$\mathcal{B}$ chooses random s_1, t and $\mathrm{tag}_c \in \mathbb{Z}_p$. It will implicitly let $s_2 = c_3$. The message $M_\beta \in \mathbb{G}_T$ is blinded as $C_0 = M_\beta \cdot T^{a_1 b}$ It then chooses random $x' \in \mathbb{Z}_p$ and will implicitly set $x = -c_3 + x'$.

$$C_1 = g^{s_1 \cdot b} + (g^{c_3})^b, \quad C_2 = g^{b \cdot a_1 \cdot s_1}, \quad C_3 = g^{a_1 \cdot s_1}, \quad C_4 = (g^{c_2})^{x' \cdot b}, \quad C_5 = (g^{c_2})^{x'},$$

$$C_6 = \tau_1^{s_1} (g^{c_3})^{y_v} (g^{c_2})^{y_{v_2} \cdot x'}, \quad C_7 = (\tau_1^b)^{s_1} (g^{c_3})^{y_v \cdot b} (g^{c_2})^{y_{v_2} \cdot x' \cdot b} w^{-t}, \quad E_1 = (u^{\mathcal{I}} w^{\mathrm{tag}_c} h)^t, \quad E_2 = g^t.$$

If T is a tuple, then we are in $\mathbf{Game}_q$, otherwise we are in $\mathbf{Game}_{\mathrm{Final}}$. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

C Hierarhical IBE Proof

We now provide our proof of security. We first define the semi-functional algorithms. Then we provide our HIBE security proof.

C.1 Semi-Functional Algorithms

We now describe the semi-functional ciphertext and key generation algorithms for HIBE. Again, we will define them as algorithms that are executed with knowledge of the secret exponents; however, in a real system they will not be used. Their main purpose is to define the structures that will be used in our proof. We define both semi-functional ciphertexts and keys in terms of a transformation on a normal ciphertext or key.

Semi-Functional Ciphertexts The algorithm first runs the encryption algorithm to generate a normal ciphertext CT for hierarchical identity $\vec{\mathcal{I}}$ of depth d and message M with $C'_1, \ldots, C'_7, E'_1, \ldots, E'_d, \tilde{E}'$ with $\text{tag}_{c1}, \ldots \text{tag}_{cd}$. Then it chooses a random $x \in \mathbb{Z}_p$. It sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$, $E_1 = E'_1, \ldots, E_d = E'_d$, $\tilde{E} = \tilde{E}'$, leaving these elements and the tag_c unchanged. It then sets

$$C_4 = C'_4 \cdot g^{ba_2x}, \quad C_5 = C'_5 \cdot g^{a_2x}, \quad C_6 = C'_6 \cdot v_2^{a_2x}, \quad C_7 = C'_7 \cdot v_2^{a_2bx}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, \ldots, E_d, \text{tag}_{c1}, \ldots \text{tag}_{cd}$.

Semi-Functional Secret Keys The algorithm first runs the encryption algorithm to generate a normal private key $\text{SK}_{\vec{\mathcal{I}}}$ for hierarchical identity $\vec{\mathcal{I}}$ of depth d with $D'_1, \ldots, D'_7, (K'_{1,1}, K'_{1,2}), \ldots, (K'_{d,1}, K'_{d,2})$ and $\text{tag}_{k1}, \ldots, \text{tag}_{kd}$. Then it chooses a random $\gamma \in \mathbb{Z}_p$. It sets $D_3 = D'_3, D_5 = D'_5, D_6 = D'_6, D_7 = D'_7, K_{i,1} = K'_{i,1}$ and $K_{i,2} = K'_{i,2}$ for all i, leaving these elements and the tag values unchanged. It then sets

$$D_1 = D'_1 g^{-a_1a_2\gamma}, \quad D_2 = D'_2 \cdot g^{a_2\gamma}, \quad D_4 = D'_4 \cdot g^{a_1\gamma}.$$

The semi-functional secret key is $\text{SK}_{\vec{\mathcal{I}}} = D_1, \ldots, D_7, (K_{1,1}, K_{1,2}), \ldots, (K_{d,1}, K_{d,2}), \text{tag}_{k1}, \ldots, \text{tag}_{kd}$.

C.2 Proof of Security

We now present the proof of security for our HIBE system. The proof will follow the same guiding principles as the IBE proof of Section 3. The most challenging aspect will occur in arguing that an attacker cannot distinguish a semi-functional key from a normal one.

It is useful to consider how we can prove security (against key distinguishing) under the first definitions of HIBE security [23, 21] where all the queries the adversary makes come from the authority's key generation algorithm. In this case we setup the parameters in an analogous manner to the previous proof by choosing $(A_1, B_1), \ldots, (A_n, B_n)$ to define n the functions $F_i(\mathcal{I}) = A_i\mathcal{I} + B$ for $1 \leq i \leq n$.

The two primary issues are how to form the k-th key and challenge ciphertext when arguing that no attacker can distinguish $\textbf{Game}_{k-1}$ from $\textbf{Game}_k$. When giving out the k-th key, a depth d key $\vec{\mathcal{I}}$, we would choose $\text{tag}_{k1}, \ldots, \text{tag}_{kd-1}$ at random and choose $\text{tag}_{kd} = F_d(\mathcal{I}_d)$. To create the challenge ciphertext for an identity $\vec{\mathcal{I}}^* = \mathcal{I}_1^* : \cdots : \mathcal{I}_{d^*}^*$ of depth d^* we will derive $\text{tag}_{ci} = F_i(\mathcal{I}_I^*)$ for each level i. All tags will appear correctly distributed. First, for $i \neq d$ the attacker will only see at most one evaluation of $F_i(\cdot)$ from the challenge ciphertext. For $F_d\cdot$ the attacker will see at most two evaluations; furthermore, if $d^* \geq d$ then $\text{tag}_{kd} = F_d(\mathcal{I}_d) \neq F_d(\mathcal{I}_d^*) = \text{tag}_{kd}$. Otherwise, the $\vec{\mathcal{I}}$ is a key that is allowed to decrypt $\vec{\mathcal{I}}^*$ and could not have been legally requested.

As we argued in Section 5, the definitions that don't track delegation cannot adequately model security for our system and we must use the definitions in Appendix A. Arguing security with this complete definition poses some challenges. We would like to argue using a sequence of games of

the keys revealed to the attacker for an attacker that makes at most q_R reveal queries. Ideally, if the k-th query revealed for $\vec{\mathcal{I}}$ was of depth d we could set its value $\text{tag}_{kd} = F_d(\mathcal{I}_d)$ according to our strategy above. However, this presents problems. If prior to revealing that key the adversary asked to reveal some descendant of it, then the descendant will have the same tag_{kd} value as the k-th revealed key; therefore, it "locks" in the tag value before we are even know it will be the key we are distinguishing on!

We deal with this in the following way. Suppose we are given an attacking algorithm $\mathcal{A}$ that makes at most q_R reveal queries and q_A create and derive queries. We will make the hybrid argument over the revealed queries from 1 to q_R. However, at hybrid k we will a guess that the jth key that was created or derived will be the kth one revealed. If the guess is wrong the reduction algorithm $\mathcal{B}$ will need to abort; however, it will be correct $1/q_A$ of the time

Again, we organize our proof as a sequence of games. The first game defined will be the real identity-based encryption game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next (under a complexity assumption).

$\text{Game}_{\text{Real}}$: The actual IBE security game defined in Appendix A.

Game_i: The real security game with the following two exceptions: 1) The challenge ciphertext will be a semi-functional ciphertext on the challenge identity $\vec{\mathcal{I}}^*$. 2) The first i private key queries will return semi-functional private keys. The rest of the keys will be normal. In the security game the keys will be changed from normal to semi-functional right before they are revealed. Therefore, if the j-th key revealed for $j > i$ is a descendant of one of the first i keys revealed; it will be a normal key.

For an adversary that makes at most q_R reveal queries we will be interested in $\mathbf{Game}_0, \ldots,$ $\mathbf{Game}_{q_R}$. We note that in $\mathbf{Game}_0$ the challenge ciphertext is semi-functional, but all keys are normal and in $\mathbf{Game}_{q_R}$ all private keys are semi-functional.

$\text{Game}_{\text{Final}}$: The real security game with the following exceptions: 1) The challenge ciphertext is a semi-functional encryption of a *random* group element of $\mathbb{G}_T$. 2) *All* of the private key queries result in semi-functional keys.

We now prove a set of Lemmas that argue about the distinguishablity of these games. For each proof we need to build a reduction simulator that both answers private key queries and creates a challenge ciphertext. We let $\mathbf{Game}_{\text{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ denote an algorithm $\mathcal{A}$'s advantage in the real game.

Lemma 4. *Suppose that there exists an algorithm $\mathcal{A}$ where $\mathbf{Game}_{Real}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_0\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision-Linear problem. We now describe how it executes the Setup, Key Phase; and Challenge phases of the HIBE game with $\mathcal{A}$.

Setup(λ, n) The algorithm chooses random exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_p$ and random group elements $u_1, \ldots, u_n, w, h_1, \ldots, h_n \in \mathbb{G}$. It then sets $g = g, g^{a_1} = f, g^{a_2} = \nu$; intuitively a_1, a_2 are the exponents that the reduction cannot know itself.

Finally, it sets the variables as:

$$g^b,\ g^{b\cdot a_1} = f^b,\ g^{b\cdot a_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

Using this it can calculate $\tau_1, \tau_2, \tau_1^b, \tau_2^b$ and $e(g,g)^{\alpha a_1 b} = e(g,f)^{\alpha \cdot b}$ in order to publish the public parameters PK. We also note that using α it can compute the master secret key for itself.

Key Generation Phase 1,2 Since $\mathcal{B}$ has the actual master secret key MSK it simply runs the key generation to generate the keys in both phases. Note that the MSK it has only allows for the creation of normal keys.

Challenge ciphertext $\mathcal{B}$ receives two messages M_0, M_1 and challenge identity $\vec{\mathcal{I}}^*$ of depth d. It then flips a coin β. We describe the creation of the challenge ciphertext in two steps. First, it creates a normal ciphertext using the real algorithm by calling Encrypt(PK, $\vec{\mathcal{I}}^*, M_\beta$), which outputs a ciphertext $\mathrm{CT} = C'_0, \ldots, C'_7, E'_1, \ldots, E'_d, \tilde{E}', \mathrm{tag}_{c1}, \ldots, \mathrm{tag}_{cd}$. Let s'_1, s'_2, t' be the random exponents used in creating the ciphertext.

Then we modify components of our ciphertext as follows. It sets

$$C_0 = C'_0 \cdot \big(e(g^{c_1}, f) \cdot e(g, f^{c_2})\big)^{b\cdot\alpha}, \quad C_1 = C'_1 \cdot (g^{c_1})^b, \quad C_2 = C'_2 \cdot (f^{c_2})^{-b}, \quad C_3 = C'_3 \cdot (f^{-c_2}), \quad C_4 = C'_4 \cdot (T)^b,$$

$$C_5 = C'_5 \cdot T, \quad C_6 = C'_6 \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}, \quad C_7 = C'_7 \cdot \big((g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}\big)^b,$$

$$E_1 = E'_1, \ldots, E_d = E'_d,\ \tilde{E} = \tilde{E}'.$$

The returned ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, \ldots, E_d, \tilde{E}, \mathrm{tag}_{c1}, \ldots, \mathrm{tag}_{cd}$

If T is a tuple, then this assignment implicitly sets $s_1 = -c_2 + s'_1, s_2 = s'_2 + c_1 + c_2$ and $s = s_1 + s_2 = c_1 + s'_1 + s'_2$. If $T = \nu^{c_1+c_2}$ it will have the same distribution as a standard ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Lemma 5. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q_R reveal queries and $\mathbf{Game}_{\mathsf{k}-1}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{\mathsf{k}}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$ for some k where $1 \le k \le q$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision-Linear problem. We now describe how it executes the Setup, Key Phase; and Challenge phases of the HIBE game with $\mathcal{A}$.

Setup Algorithm $\mathcal{B}$ first chooses random exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_{u_1}, \ldots, y_{u_n}, y_{h_1}, \ldots, y_{h_n}$. It then defines

$$g = g, g^b = f, g^{b\cdot a_1} = f^{a_1},\ g^{b\cdot a_2} = f^{a_2},\ v = \nu^{-a_1\cdot a_2},\ v_1 = \nu^{a_2}\cdot g^{y_{v_1}},\ v_2 = \nu^{a_1}\cdot g^{y_{v_2}},\ e(g,g)^{\alpha\cdot a_1 b} = e(f,g)^{\alpha\cdot a_1}.$$

Now it can create

$$\tau_1 = v v_1^{a_1} = g^{y_{v_1} a_1}, \quad \tau_2 = v v_1^{a_2} = g^{y_{v_2} a_2}, \quad \tau_1^b = v v_1^{a_1} = f^{y_{v_1} a_1}, \quad \tau_2^b = v v_1^{a_2} = f^{y_{v_2} a_2}.$$

Finally, $\mathcal{B}$ chooses random $(A_1, B_1), \ldots, (A_n, B_n) \in \mathbb{Z}_p$. It then sets

$$w = fg^{y_w}, \quad u_1 = f^{-A_1} g^{y_{u_1}}, \ldots, u_n = f^{-A_n} g^{y_{u_n}}, \; h_1 = f^{-B_1} g^{y_{h_1}}, \ldots, h_n = f^{-B_n} g^{y_{h_n}}.$$

This will define all the public parameters of the system. Note that by virtue of knowing α, the algorithm $\mathcal{B}$ will know the regular master secret key.

We again highlight the importance of the function $F_i(\mathcal{I}_i) = A_i \cdot \mathcal{I} + B_i$. For $\text{tag}_{ci} = F_i(\mathcal{I})$ we have $(u_i^{\mathcal{I}_i} w_i^{\text{tag}_c} h_i) = f^{\text{tag}_{ci} - A_i \cdot \mathcal{I}_i - B_i} g^{\mathcal{I} \cdot y_{u_i} + y_{h_i} + \text{tag}_{ci} \cdot y_w} = g^{\mathcal{I} \cdot y_u + y_{h_i} + \text{tag}_{ci} \cdot y_w}$. In this case $\mathcal{B}$ will know the discrete log base g of the function. We also note that A_i, B_i are initially information theoretically hidden from the adversary. Since it is a pairwise independent function, if the adversary is given $F_i(\mathcal{I}_i)$ for some identity, the, all values in $\mathbb{Z}_p$ are equally likely for $F_i(\mathcal{I}_i')$ for some $\mathcal{I}_i \neq \mathcal{I}_i'$ for all $1 \leq i \leq n$.

Key Gen Phases 1,2 We break the Key Generation into three cases. Key Generation is done the same regardless of whether we are in phase 1 or 2. In this game of security we need to explain both how key delegate and create queries are handled and how key reveal queries are handled.

The algorithm $\mathcal{B}$ begins by choosing a value κ uniformly at random between 1 and Q_A. $\mathcal{B}$ is guessing that the k-th query revealed will be the κ key either created directly or created by delegation. We now describe how the $\mathcal{B}$ handles create, delegate, and reveal requests.

We first describe how it handles create and delegate queries. The algorithm initially creates a counter j for the number of combined create and delegate queries. Initially, j is set to 0. When either a delegate or create request is made j is first incremented by 1. $\mathcal{B}$ handles this in one of two way.

Case 1: $j \neq \kappa$

If this is a create query for an identity $\vec{\mathcal{I}}$ of depth, d, then the algorithm chooses random tags $\text{tag}_{k1}, \ldots, \text{tag}_{kd}$ and associates them and $\vec{\mathcal{I}}$ to the j-th member of the set S of created keys.

Suppose this is a delegate query to delegate from a previous key of identity $\vec{\mathcal{I}}$ of depth $d-1$ to $\vec{\mathcal{I}} : \mathcal{I}_d$ of depth d. The algorithm chooses *one* new tag tag_{kd}, the other d tags $\text{tag}_{k1}, \ldots, \text{tag}_{kd-1}$ are copied from the element that we are delegating from. These tags and $\vec{\mathcal{I}} : \mathcal{I}_{d+1}$ are associated with the j-th element of S.

Case 2: $j = \kappa$

If this is a create query for an identity $\vec{\mathcal{I}}$ of depth d then the algorithm chooses random tags $\text{tag}_{k1}, \ldots, \text{tag}_{kd-1}$. Then it sets $\text{tag}_{kd} = F_d(\mathcal{I}_d)$. It associates the tags and $\vec{\mathcal{I}}$ to the j-th member of the set S of created keys.

Suppose this is a delegate query to delegate from a previous key of identity $\vec{\mathcal{I}}$ of depth $d-1$ to $\vec{\mathcal{I}} : \mathcal{I}_d$ of depth d. The algorithm sets *one* new tag $\text{tag}_{kd} = F_d(\mathcal{I}_d)$ and the other d tags $\text{tag}_{k1}, \ldots, \text{tag}_{kd-1}$ are copied from the element that we are delegating from. These tags and $\vec{\mathcal{I}} : \mathcal{I}_d$ are associated with the j-th element of S.

Now we show how to execute a reveal query of an existing element of S. At this point any element of S has assoicated with it an identity and tag values, but no actual key. Since private keys are completely re-randomized, other than the tag values at each delegate invocation we can construct these keys during the actual reveal phase. Consider the i-th reveal query made by $\mathcal{A}$.

Case 1: $i > k$

Suppose the attacker ask for the j-th key to be revealed. When i is greater than k our algorithm $\mathcal{B}$ will generate a normal key for the requested identity $\vec{\mathcal{I}}$. Since it has the master secret key MSK it can run that algorithm, using the tag values from the j-th element added to S.

Case 2: $i < k$
Suppose the attacker ask for the j-th key to be revealed. When i is less than k our algorithm $\mathcal{B}$ will generate a semi-functional key for the requested identity $\mathcal{I}$. It first creates a normal key using MSK and the tag values from the j-th element added to S. Then it makes it semi-functional using the procedure from above in Subsection 3.2. It can run this procedure since it knows $g^{a_1 a_2}$.
Case 3: $i = k$
Suppose the attacker ask for the j-th key to be revealed. If $j \neq \kappa$ then the $\mathcal{B}$ aborts the simulation and guesses whether T is a tuple randomly.

The algorithm first runs the key generation algorithm to generate a normal private key $\mathrm{SK}_{\vec{\mathcal{I}}}$ for identity $\vec{\mathcal{I}}$ of depth d with $D'_1, \ldots, D'_7, (K'_{1,1}, K'_{1,2}), \ldots, (K'_{d,1}, K'_{d,2})$ using the tag_k values already assigned. Recall that $\mathrm{tag}_{kd} = F(\mathcal{I}_d)$. Let r'_1, r'_2, z'_1, z'_2 be the random exponents used.

It then sets

$$D_1 = D'_1 \cdot T^{-a_1 \cdot a_2},\ D_2 = D'_2 \cdot T^{a_2}(g^{c_1})^{y_{v_1}},\ D_3 = D'_3 \cdot (f^{c_2})^{y_{v_1}},\ D_4 = D'_4 \cdot T^{a_1}(g^{c_1})^{y_{v_2}},$$

$$D_5 = D'_5 \cdot (f^{c_2})^{y_{v_2}},\ D_6 = D'_6 \cdot f^{c_2},\ D_7 = D'_7 \cdot (g^{c_1}),$$

$$(K_{1,1} = K'_{1,1},\ K_{1,2} = K'_{1,2}), , \ldots, (K_{d-1,1} = K'_{d-1,1},\ K_{d-1,2} = K'_{d-1,2})$$

$$(K_{d,1} = K'_{d,1} \cdot (g^{c_1})^{\mathcal{I}_d \cdot y_{u_d} + y_{h_d} + \mathrm{tag}_{k\,d} \cdot y_w}, K_{d,2} = K'_{d,2} \cdot (g^{c_1}).)$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, (K_{1,1}, K_{1,2}), \ldots, (K_{d,1}, K_{d,2}), \mathrm{tag}_{k1}, \ldots, k_d$. We emphasize that the fact that $\mathrm{tag}_k = F(\mathcal{I})$ allowed us to created the component $K_{d,1}$. In addition, we note that we implicitly set $z_1 = z'_1 - y_{v_1} c_2$ and $z_2 = z'_2 - y_{v_2} c_2$ in order to be able to create D_2 and D_4.

If T is a linear tuple of the form $T = \nu^{c_1 + c_2}$ then the k-th query results in a normal key under randomness $r_1 = r'_1 + c_1$ and $r_2 = r'_2 + c_2$. Otherwise, if T is a random group element, then we can write $T = \nu^{c_1 + c_2} g^{\gamma}$ for random $\gamma \in \mathbb{Z}_p$. This forms a semi-functional key where γ is the added randomness to make it semi-functional. The reduction willl not abort $1/q_A$ amount of the time and the abort condition will be independent of the adversary's success.

Challenge Ciphertext Algorithm $\mathcal{B}$ is given a challenge identity $\vec{\mathcal{I}}^*$ of depth d^* and messages M_0, M_1. Then it flips a coin β.

In this phase $\mathcal{B}$ needs to be able to generate a semi-functional challenge ciphertext. One problem is that $\mathcal{B}$ does not have the group element v_2^b so it cannot directly create such a ciphertext. However, in the case where $\mathrm{tag}^*_{ci} = F_i(\mathcal{I}^*_i)$ for all i where $1 \leq i \leq d^*$ it will have a different method of doing so.

$\mathcal{B}$ first runs the normal encryption algorithm to generate a normal ciphertext CT for identity $\mathcal{I}^*$ and message M^*. During this run it uses $\mathrm{tag}^*_{ci} = F_i(\mathcal{I}^*_i)$ for all i less than or equal to d^*. It then gets a standard ciphertext $C'_1, \ldots, C'_7, E'_1, \ldots, E'_{d^*}$ under random exponents s'_1, s'_2, t'

To make it semi-functional it chooses a random $x \in \mathbb{Z}_p$. It first sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$ leaving these elements and the $\mathrm{tag}_c{}^*$ unchanged. It then sets

$$C_4 = C'_4 \cdot f^{a_2 \cdot x}, \quad C_5 = C'_5 \cdot g^{a_2 \cdot x}, \quad C_6 = C'_6 \cdot v_2^{a_2 x}, \quad C_7 = C'_7 \cdot f^{y_{v_2} \cdot x a_2} \nu^{-a_1 a_2 \cdot x \cdot y_w},$$

$$E_1 = E'_1 \cdot (\nu^{\mathcal{I}_1 \cdot y_{u_1} + y_{h_1} + \mathrm{tag}_{ci} \cdot y_w})^{a_1 a_2 x}, \ldots, E_{d^*} = E'_{d^*} \cdot (\nu^{\mathcal{I}_{d^*} \cdot y_{u_{d^*}} + y_{h_{d^*}} + \mathrm{tag}_{c\,d^*} \cdot y_w})^{a_1 a_2 x},$$

$$\tilde{E} = \tilde{E}' \cdot \nu^{a_1 a_2 \cdot x}$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, \ldots, E_{d^*}, \tilde{E}, \text{tag}_{c1}, \ldots, \text{tag}_{cd^*}$.

Intuitively, the algorithm implicitly sets $g^t = g^{t'} + \nu^{a_1 a_2 x}$. This allows for the cancellation of the term $v_2^{a_1 a_2 bx}$ by w^{-t} in constructing C_7. Normally, this would be problematic for the generation of $E_1, \ldots, E_{d^*}$; however since $\text{tag}_{ci}^* = F_i(\mathcal{I}_i^*)$ for all i the algorithm $\mathcal{B}$ is able to create these terms.

If T is a tuple, then we are in $\mathbf{Game}_{k-1}$, otherwise we are in $\mathbf{Game}_k$. We highlight that even if the challenge ciphertext depth d^* is greater than or equal to k-th keys depth d, the adversary cannot detect any special relationship between tag_{cd}^* and tag_{kd}^* since $F_d(\mathcal{I}) = A_d \cdot \mathcal{I} + B_d$ is a pairwise independent function and A, B are hidden from its view.

$\mathcal{B}$ receives a bit β' and outputs 0 if $\beta = \beta'$.

Lemma 6. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and* $\mathbf{Game}_{\mathsf{q}}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{Final}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. *Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision BDH game.*

Proof. We begin by noting that in both of these two games the challenge ciphertexts and all the private keys are semi-functional. Therefore, $\mathcal{B}$ only needs to be able to generate semi-functional private keys.

$\mathcal{B}$ begins by taking in a BDH instance $g, g^{c_1}, g^{c_2}, g^{c_3}, T$.

Setup The algorithm begins by choosing random exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_{h_1}, \ldots, y_{h_n}, y_{u_1}, \ldots, y_{u_n} \in \mathbb{Z}_p$. It then sets

$$g = g,\ g^b,\ g^{a_1},\ g^{a_2} = g^{c_2},\ g^{b \cdot a_1},\ g^{b \cdot a_2} = (g^{c_2})^b,$$

$$v = g^{y_v},\ v_1 = g^{y_{v_1}},\ v_2 = g^{y_{v_2}},\ w = g^{y_w},$$

$$(u_1 = g^{y_{u_1}}, \ldots, u_n = g^{y_{u_n}}), (h_1 = g^{y_{h_1}}, \ldots, h_n = g^{y_{h_n}}), e(g,g)^{a_1 \alpha b} = e(g^{c_1}, g^{c_2})^{a_1 \cdot b}.$$

These parameters allow us to compute $\tau_1 = v v_1^{a_1}, \tau_1^b$; and $\tau_2 = v (g^{c_2})^{y_{v_2}}, \tau_2^b$ and publish the public key PK. Note that the master secret key g^α is not available to $\mathcal{B}$.

We point out that this setup lets $\alpha = c_1 \cdot c_2$ and $a_2 = c_2$.

Key Generation Phase 1,2 All key generations result in semi-functional keys. When a request for $\vec{\mathcal{I}}$ of depth d is made, the key generation algorithm chooses random $r_1, r_2, z_1, z_2, \gamma', \text{tag}_{k1}, \ldots, \text{tag}_{kd} \in \mathbb{Z}_p$ and defines $r = r_1 + r_2$. In addition, it chooses random $\mu_1, \ldots, \mu_d \in \mathbb{Z}_p$ with the constraint that they sum to r_1. It aims to implicitly sets the variable $\gamma = c_1 + \gamma'$.

It creates the key as:

$$D_1 = (g^{c_2})^{-\gamma' \cdot a_1} v^r, \quad D_2 = (g^{c_2})^{\gamma'} v_1^r g^{z_1}, \quad D_3 = (g^b)^{-z_1}, \quad D_4 = (g^{c_1})^{a_1} g^{a_1 \cdot \gamma'} v_2^r g^{z_2},$$

$$D_5 = (g^b)^{-z_2}, \quad D_6 = g^{r_2 \cdot b}, \quad D_7 = g^{r_1},$$

$$(K_{1,1} = (u_1^{\mathcal{I}_1} w^{\text{tag}_{k1}} h_1)^{\mu_1},\ K_{1,2} = g^{\mu_1}), \ldots, (K_{d,1} = (u_d^{\mathcal{I}_d} w^{\text{tag}_{kd}} h_d)^{\mu_d},\ K_{d,2} = g^{\mu_d}.)$$

Challenge Ciphertext $\mathcal{B}$ receives a challenge identity $\vec{\mathcal{I}}^*$ of depth d and two message M_0, M_1 from the attacker. $\mathcal{B}$ will now create a challenge ciphertext that is a semi-functional ciphertext of either M_β or a random message, depending on T. It first chooses a random bit β.

$\mathcal{B}$ chooses random s_1, t and $\text{tag}_{c1}, \ldots, \text{tag}_{cd} \in \mathbb{Z}_p$. It will implicitly let $s_2 = c_3$. The message $M_\beta \in \mathbb{G}_T$ is blinded as $C_0 = M_\beta \cdot T^{a_1 b}$. It then chooses random $x' \in \mathbb{Z}_p$ and will implicitly set $x = -c_3 + x'$.

$$C_1 = g^{s_1 \cdot b} + (g^{c_3})^b, \quad C_2 = g^{b \cdot a_1 \cdot s_1}, \quad C_3 = g^{a_1 \cdot s_1}, \quad C_4 = (g^{c_2})^{x' \cdot b}, \quad C_5 = (g^{c_2})^{x'},$$

$$C_6 = \tau_1^{s_1} (g^{c_3})^{y_v} (g^{c_2})^{y_{v_2} \cdot x'}, \quad C_7 = (\tau_1^b)^{s_1} (g^{c_3})^{y_v \cdot b} (g^{c_2})^{y_{v_2} \cdot x' \cdot b} w^{-t},$$

$$E_1 = (u_1^{\mathcal{I}_1} w^{\text{tag}_{c1}} h_1)^t, \ldots, E_d = (u_d^{\mathcal{I}_d} w^{\text{tag}_{cd}} h_d)^t, \quad \tilde{E} = g^t.$$

If T is a tuple, then we are in $\mathbf{Game}_q$, otherwise we are in $\mathbf{Game}_{\text{Final}}$. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Theorem 2. *If the decisional Linear and decisional BDH assumptions hold, then no poly-time algorithm can break our HIBE system.*

Proof. Any attacker's advantage in $\mathbf{Game}_{\text{Final}}\, \mathsf{Adv}_{\mathcal{A}}$ in the final game must be 0 since it completely hides the bit β. By the sequence of games we established and Lemmas 4,5,6 an attacker's advantage in the real game $\mathbf{Game}_{\text{Real}}\, \mathsf{Adv}_{\mathcal{A}}$ must be negligibly close to 0.

D Broadcast Encryption

We now show how to use our techniques to give a secure broadcast encryption scheme with ciphertext overhead of a constant number of group elements. Our system is the only one under static (i.e. non q-based) assumptions to achieve this. We point out that only recently Gentry and Waters [22] gave a short ciphertext broadcast encryption system that was even adaptively secure. One feature of our methodology is that it is relatively simple to leverage our techniques to get adaptive security. Our broadcast encryption system is actually perfectly correct and does not use tags.

We refer the reader to [22] for the adaptive security definition of broadcast encryption.

D.1 Construction

Setup(λ, n) The setup algorithm takes as input a security paramter and n, the number of users in the system. The authority first chooses a group $\mathbb{G}$ of prime order p. Next, it chooses generators $g, v, v_1, v_2, w, u_1, \ldots, u_n \in \mathbb{G}$ and exponents $a_1, a_2, b, \alpha \in \mathbb{Z}_p$. Let $\tau_1 = v v_1^{a_1}, \tau_2 = v v_2^{a_2}$. It publishes the public parameters PK as the group description $\mathbb{G}$ along with:

$$g^b,\ g^{a_1},\ g^{a_2}, g^{b \cdot a_1},\ g^{b \cdot a_2},\ \tau_1, \tau_2, \tau_1^b, \tau_2^b,\ w,\ u_1, \ldots, u_n, e(g,g)^{\alpha \cdot a_1 \cdot b}.$$

The master secret key MSK consists of $g, g^\alpha, g^{\alpha \cdot a_1}, v, v_1, v_2$ as well as the public parameters. The identity space for the described scheme will be $\mathbb{Z}_p$, although we note in practice one can apply a collision resistant function to identities of arbitrary lengths.

Encrypt$(\mathrm{PK}, S \in \{1, 2, \ldots, n\}, M)$ The encryption algorithm takes as input the public paraemters, a set S of indicies to encrypt to, and a message M. The encryption algorithm chooses random $s_1, s_2, t \in \mathbb{Z}_p$. Let $s = s_1 + s_2$. It then blinds $M \in \mathbb{G}_T$ as $C_0 = M \cdot (e(g,g)^{\alpha a_1 \cdot b})^{s_2}$ and creates:

$$C_1 = (g^b)^{s_1+s_2},\ C_2 = (g^{b \cdot a_1})^{s_1}, \quad C_3 = (g^{a_1})^{s_1},\ C_4 = (g^{b \cdot a_2})^{s_2},\ C_5 = (g^{a_2})^{s_2},\ C_6 = \tau_1^{s_1}\tau_2^{s_2},\ C_7 = (\tau_1^b)^{s_1}(\tau_2^b)^{s_2}w^{-t},$$

$$E_1 = (\prod_{i \in S} u_i)^t,\ E_2 = g^t.$$

The ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, E_2$.

KeyGen$(\mathrm{MSK}, k \in [1, n])$ The KeyGeneration algorithm takes as input the master secret key and an index $k \in [1, n]$ and outputs the k-th user's private key. The authority chooses random $r_1, r_2, z_1, z_2 \in \mathbb{Z}_p$. Let $r = r_1 + r_2$.

Then it creates:

$$D_1 = g^{\alpha \cdot a_1} v^r,\ D_2 = g^{-\alpha} v_1^r g^{z_1},\ D_3 = (g^b)^{-z_1},\ D_4 = v_2^r g^{z_2},\ D_5 = (g^b)^{-z_2},\ D_6 = g^{r_2 \cdot b},\ D_7 = g^{r_1}$$

$$K = (u_k w)^{r_1}, \quad \forall_{i \neq k}\ K_i = u_i^{r_1}.$$

The secret key is $\mathrm{SK}_k = D_1, \ldots, D_7, K,\ \forall_{i \neq k}\ K_i$.

Decrypt$(\mathrm{CT}, S, \mathrm{SK}_k)$ The decryption algorithm will take as input a ciphertext CT and a set S of users that the ciphertext is encrypted to as well as its key SK_k. The algorithm will be able to decrypt with user k's secret key a ciphertext if $k \in S$.

We break the decryption algorithm into a set of calculations. First, it computes:

$$\begin{aligned} A_1 &= e(C_1, D_1) \cdot e(C_2, D_2) \cdot e(C_3, D_3) \cdot e(C_4, D_4) \cdot e(C_5, D_5) \\ &= e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(v, g)^{b(s_1+s_2)r} e(v_1, g)^{a_1 b s_1 r} e(v_2, g)^{a_2 b s_2 r}. \end{aligned}$$

Recall that $r = r_1 + r_2$. Next, it computes

$$\begin{aligned} A_2 &= e(C_6, D_6) \cdot e(C_7, D_7) \\ &= e(v, g)^{b(s_1+s_2)r} e(v_1, g)^{a_1 b s_1 r} e(v_2, g)^{a_2 b s_2 r} \cdot e(g, w)^{-r_1 t}. \end{aligned}$$

Taking, $A_3 = A_1/A_2 = e(g,g)^{\alpha \cdot a_1 \cdot b \cdot s_2} \cdot e(g, w)^{r_1 \cdot t}$ leaves us with one more cancellation to get the message blinding factor.

$$A_4 = \big(e(E_2, K \prod_{\substack{i \in S \\ i \neq k}} K_i)/e(E_1, D_7)\big) = e(g, w)^{r_1 \cdot t}.$$

Finally, we can recover the message by computing

$$C_0/(A_3/A_4) = M.$$

Altogether, decryption requires nine applications of the pairing algorithm.

D.2 Semi-Functional Algorithms

We now describe the semi-functional ciphertext and key generation algorithms.

Semi-Functional Ciphertexts The algorithm first runs the encryption algorithm to generate a normal ciphertext CT for set S and message M with $C'_1, \ldots, C'_7, E'_1, E'_2$. Then it chooses a random $x \in \mathbb{Z}_p$. It sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3, E_1 = E'_1, E_2 = E'_2$, leaving these elements unchanged. It then sets

$$C_4 = C'_4 \cdot g^{ba_2x}, \quad C_5 = C'_5 \cdot g^{a_2x}, \quad C_6 = C'_6 \cdot v_2^{a_2x}, \quad C_7 = C'_7 \cdot v_2^{a_2bx}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2,$.

Semi-Functional Secret Keys The algorithm first runs the encryption algorithm to generate a normal private key SK_k for user $k \in [1, n]$ with $D'_1, \ldots, D'_7, K_i \forall i \neq k$. Then it chooses a random $\gamma \in \mathbb{Z}_p$. It sets $D_3 = D'_3, D_5 = D'_5, D_6 = D'_6, D_7 = D'_7, K_i = K'_i \forall i \neq k$ leaving these elements unchanged. It then sets

$$D_1 = D'_1 g^{-a_1 a_2 \gamma}, \quad D_2 = D'_2 \cdot g^{a_2\gamma}, \quad D_4 = D'_4 \cdot g^{a_1\gamma}.$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \ \forall_{i \neq k} \ K_i$.

D.3 Proof of Security

We organize our proof as a sequence of games. The first game defined will be the real identity-based encryption game and the last one will be one in which the adversary has no advantage unconditionally. We will show that each game is indistinguishable from the next (under a complexity assumption). As stated before the crux of our strategy is to move to a security game where both the challenge ciphertext and private keys are semi-functional. At this point any keys the challenger gives out are not useful in decrypting the ciphertext. We first define the games as:

$\mathbf{Game_{Real}}$: The actual broadcast encryption security game.

$\mathbf{Game}_i$: Let $\mathcal{K}$ be the set of private keys the adversary requests during the protocol. This game is the real security game with the following two exceptions: 1) The challenge ciphertext will be a semi-functional ciphertext on the challenge set S^*. 2) For any index $j \leq i$ $j \in \mathcal{K}$ the game will return a semi-functional private key SK_j. The rest of the keys in $\mathcal{K}$ will be normal.

For an system of n users we will be interested in $\mathbf{Game}_0, \ldots, \mathbf{Game}_n$. We note that in $\mathbf{Game}_0$ the challenge ciphertext is semi-functional, but all keys are normal and in $\mathbf{Game}_n$ all private keys are semi-functional.

$\mathbf{Game_{Final}}$: The real security game with the following exceptions: 1) The challenge ciphertext is a semi-functional encryption of a *random* group element of $\mathbb{G}_T$. 2) *All* of the private key queries result in semi-functional keys.

We now prove a set of Lemmas that argue about the distinguishablity of these games. For each proof we need to build a reduction simulator that both answers private key queries and creates a challenge ciphertext. We let $\mathbf{Game}_{\mathrm{Real}}\,\mathsf{Adv}_{\mathcal{A}}$ denote an algorithm $\mathcal{A}$'s advantage in the real game.

Lemma 7. *Suppose that there exists an algorithm $\mathcal{A}$ where $\mathbf{Game}_{Real}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_0\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision-Linear problem. We now describe how it executes the Setup, Key Phase and Challenge phases of the IBE game with $\mathcal{A}$.

Setup The algorithm chooses random exponents $b, \alpha, y_v, y_{v_1}, y_{v_2} \in \mathbb{Z}_p$ and random group elements $u_1, \ldots, u_n, w \in \mathbb{G}$. It then sets $g = g, g^{a_1} = f, g^{a_2} = \nu$; intuitively a_1, a_2 are the exponents that the reduction cannot know itself.

Finally, it sets the variables as:

$$g^b,\ g^{b \cdot a_1} = f^b,\ g^{b \cdot a_2} = \nu^b, v = g^{y_v}, v_1 = g^{y_{v_1}}, v_2 = g^{y_{v_2}}.$$

Using this it can calculate $\tau_1, \tau_2, \tau_1^b, \tau_2^b$ and $e(g,g)^{\alpha a_1 b} = e(g,f)^{\alpha \cdot b}$ in order to publish the public parameters PK. We also note that using α it can compute the master secret key for itself.

Key Generation Phase 1,2 Since $\mathcal{B}$ has the actual master secret key MSK it simply runs the key generation to generate the keys in both phases. Note that the MSK it has only allows for the creation of normal keys.

Challenge ciphertext $\mathcal{B}$ receives two messages M_0, M_1 and challenge set S^*. It then flips a coin β. We describe the creation of the challenge ciphertext in two steps. First, it creates a normal ciphertext using the real algorithm by calling Encrypt$(\mathrm{PK}, S^*, M_\beta)$, which outputs a ciphertext $\mathrm{CT} = C'_0, \ldots, C'_7, E'_1, E'_2$. Let s'_1, s'_2, t' be the random exponents used in creating the ciphertext.

Then we modify components of our ciphertext as follows. It sets

$$C_0 = C'_0 \cdot \big(e(g^{c_1}, f) \cdot e(g, f^{c_2})\big)^{b \cdot \alpha}, \quad C_1 = C'_1 \cdot (g^{c_1})^b, \quad C_2 = C'_2 \cdot (f^{c_2})^{-b}, \quad C_3 = C'_3 \cdot (f^{-c_2}), \quad C_4 = C'_4 \cdot (T)^b,$$

$$C_5 = C'_5 \cdot T, \quad C_6 = C'_6 \cdot (g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}, \quad C_7 = C'_7 \cdot \big((g^{c_1})^{y_v} \cdot (f^{c_2})^{-y_{v_1}} \cdot T^{y_{v_2}}\big)^b, \quad E_1 = E'_1, \quad E_2 = E'_2.$$

The returned ciphertext is $\mathrm{CT} = C_0, \ldots, C_7, E_1, E_2$.

If T is a tuple, then this assignment implicitly sets $s_1 = -c_2 + s'_1, s_2 = s'_2 + c_1 + c_2$, and $s = s_1 + s_2 = c_1 + s'_1 + s'_2$. If $T = \nu^{c_1 + c_2}$ it will have the same distribution as a standard ciphertext; otherwise, it will be distributed identically to a semi-functional ciphertext. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Lemma 8. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and* $\mathbf{Game}_{\mathsf{k}-1}\,\mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{\mathsf{k}}\,\mathsf{Adv}_{\mathcal{A}} = \epsilon$ *for some k where $1 \leq k \leq q$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision Linear game.*

Proof. We observe that the only way of distinguishing between these two games is if $\mathcal{A}$ actually requests the k-th key. Otherwise, its views in the two games are equivalent and it will have absolutely 0 advantage in distinguishing. Using this observation we prove that even when $\mathcal{A}$ requests the k-th private key, his advantage in distinguishing these games is negligible.

Our algorithm $\mathcal{B}$ begins by taking in an instance $(\mathbb{G}, g, f, \nu, g^{c_1}, f^{c_2}, T)$ of the decision-Linear problem. We now describe how it executes the Setup, Key Phase, and Challenge phases of the Broadcast Encryption game with $\mathcal{A}$.

Setup Algorithm $\mathcal{B}$ first chooses random exponents $\alpha, a_1, a_2, y_{v_1}, y_{v_2}, y_w, y_{u_1}, \ldots, y_{u_n}$. It then defines

$$g = g, g^b = f, g^{b \cdot a_1} = f^{a_1},\ g^{b \cdot a_2} = f^{a_2},\ v = \nu^{-a_1 \cdot a_2},\ v_1 = \nu^{a_2} \cdot g^{y_{v_1}},\ v_2 = \nu^{a_1} \cdot g^{y_{v_2}},\ e(g,g)^{\alpha \cdot a_1 b} = e(f,g)^{\alpha \cdot a_1}.$$

Now it can create

$$\tau_1 = v v_1^{a_1} = g^{y_{v_1} a_1}, \quad \tau_2 = v v_1^{a_2} = g^{y_{v_2} a_2}, \quad \tau_1^b = v v_1^{a_1} = f^{y_{v_1} a_1}, \quad \tau_2^b = v v_1^{a_2} = f^{y_{v_2} a_2}.$$

Finally, it sets

$$w = f g^{y_w}, \quad u_k = f^{-1} g^{y_{u_k}}, \quad \forall_{i \neq k}\ u_i = g^{y_{u_i}}.$$

This will define all the public parameters of the system. Note that by virtue of knowing α, the algorithm $\mathcal{B}$ will know the regular master secret key.

Key Gen Phases 1,2 We now describe how the system responds to a private key request. Consider a request for the i-th user's key made by $\mathcal{A}$.
Case 1: $i > k$
When i is less than k our algorithm $\mathcal{B}$ will generate a normal key for the requested user i. Since it has the master secret key MSK it can run that algorithm.
Case 2: $i < k$
When i is greater than k our algorithm $\mathcal{B}$ will generate a semi-functional key for the requested user i. It first creates a normal key using MSK. Then it makes it semi-functional using the procedure from above in Subsection 3.2. It can run this procedure since it knows $g^{a_1 a_2}$.
Case 3: i = k
The algorithm first runs the encryption algorithm to generate a normal private key SK_k for user i with $D_1', \ldots, D_7', K, \forall_{j \neq k}\ K_j$. Let r_1', r_2', z_1', z_2' be the random exponents used.

It then sets

$$D_1 = D_1' \cdot T^{-a_1 \cdot a_2},\ D_2 = D_2' \cdot T^{a_2} (g^{c_1})^{y_{v_1}},\ D_3 = D_3' \cdot (f^{c_2})^{y_{v_1}},\ D_4 = D_4' \cdot T^{a_1} (g^{c_1})^{y_{v_2}},$$

$$D_5 = D_5' \cdot (f^{c_2})^{y_{v_2}},\ D_6 = D_6' \cdot f^{c_2},\ D_7 = D_7' \cdot (g^{c_1}),\ K = K' \cdot (g^{c_1})^{y_{u_k} + y_w},\ \forall_{j \neq k}\ K_j = K_j' (g^{c_1})^{y_{u_j}}.$$

The semi-functional secret key is $\mathrm{SK} = D_1, \ldots, D_7, K, \forall_{j \neq k}\ K_j$ In addition, we note that we implicitly set $z_1 = z_1' - y_{v_1} c_2$ and $z_2 = z_2' - y_{v_2} c_2$ in order to be able to create D_2 and D_4.

If T is a Linear tuple of the form $T = \nu^{c_1 + c_2}$ then the k-th query results in a normal key under randomness $r_1 = r_1' + c_1$ and $r_2 = r_2' + c_2$. Otherwise, if T is a random group element, then we can write $T = \nu^{c_1 + c_2} g^{\gamma}$ for random $\gamma \in \mathbb{Z}_p$. This forms a semi-functional key where γ is the added randomness to make it semi-functional.

Challenge Ciphertext Algorithm $\mathcal{B}$ is given a challenge set S^* and messages M_0, M_1. Then it flips a coin β.

In this phase $\mathcal{B}$ needs to be able to generate a semi-functional challenge ciphertext. The fact that k will not be in this set enables the algorithm to create a semi-functional ciphertext.

$\mathcal{B}$ first runs the normal encryption algorithm to generate a normal ciphertext CT for set S^* and message M^*. It then gets a standard ciphertext $C_1', \ldots, C_7', E_1', E_2'$ under random exponents s_1', s_2', t'

To make it semi-functional it chooses a random $x \in \mathbb{Z}_p$. It first sets $C_1 = C'_1, C_2 = C'_2, C_3 = C'_3$ leaving these elements unchanged. It then sets

$$C_4 = C'_4 \cdot f^{a_2 \cdot x}, \quad C_5 = C'_5 \cdot g^{a_2 \cdot x}, \quad C_6 = C'_6 \cdot v_2^{a_2 x}, \quad C_7 = C'_7 \cdot f^{y_{v_2} \cdot x a_2} \nu^{-a_1 a_2 \cdot x \cdot y_w}$$

$$E_1 = E'_1 \cdot \prod_{i \in S^*} (\nu^{\mathcal{I} \cdot y_{u_i} y_h})^{a_1 a_2 x} \quad E_2 = E'_2 \cdot \nu^{a_1 a_2 \cdot x}.$$

The semi-functional ciphertext is $C_1, \ldots, C_7, E_1, E_2$.

Intuitively, the algorithm implicitly sets $g^t = g^{t'} + \nu^{a_1 a_2 x}$. This allows for the cancellation of the term $v_2^{a_1 a_2 b x}$ by w^{-t} in constructing C_7.

If T is a tuple, then we are in $\mathbf{Game}_{k-1}$, otherwise we are in $\mathbf{Game}_k$. $\mathcal{B}$ receives a bit β' and outputs 0 if $\beta = \beta'$.

Lemma 9. *Suppose that there exists an algorithm $\mathcal{A}$ that makes at most q queries and $\mathbf{Game}_{\mathsf{q}} \mathsf{Adv}_{\mathcal{A}} - \mathbf{Game}_{Final} \mathsf{Adv}_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ that has advantage ϵ in the decision BDH game.*

Proof. We begin by noting that in both of these two games both the challenge ciphertexts and all the private keys are semi-functional. Therefore, $\mathcal{B}$ only needs to be able to generate semi-functional private keys.

$\mathcal{B}$ begins by taking in a BDH instance $g, g^{c_1}, g^{c_2}, g^{c_3}, T$.

Setup The algorithm begins by choosing random exponents $a_1, b, y_v, y_{v_1}, y_{v_2}, y_w, y_{u_1}, \ldots, y_{u_n}, \in \mathbb{Z}_p$. It then sets

$$g = g,\ g^b,\ g^{a_1},\ g^{a_2} = g^{c_2},\ g^{b \cdot a_1},\ g^{b \cdot a_2} = (g^{c_2})^b,$$

$$v = g^{y_v},\ v_1 = g^{y_{v_1}},\ v_2 = g^{y_{v_2}},\ w = g^{y_w},\ u_1 = g^{y_{u_1}}, \ldots, u_n = g^{y_{u_n}}, e(g,g)^{a_1 \alpha b} = e(g^{c_1}, g^{c_2})^{a_1 \cdot b}.$$

These parameters allow us to compute $\tau_1 = v v_1^{a_1}, \tau_1^b$; and $\tau_2 = v (g^{c_2})^{y_{v_2}}, \tau_2^b$ and publish the public key PK. Note that the master secret key g^α is not available to $\mathcal{B}$.

We point out that this setup lets $\alpha = c_1 \cdot c_2$ and $a_2 = c_2$.

Key Generation Phases 1,2 All key generations result in semi-functional keys. When a request for key i is made, the key generation algorithm chooses random $r_1, r_2, z_1, z_2, \gamma' \in \mathbb{Z}_p$ and defines $r = r_1 + r_2$. It aims to implicitly sets the variable $\gamma = c_1 + \gamma'$.

It creates the key as:

$$D_1 = (g^{c_2})^{-\gamma' \cdot a_1} v^r,\ D_2 = (g^{c_2})^{\gamma'} \cdot v_1^r g^{z_1},\ D_3 = (g^b)^{-z_1},\ D_4 = (g^{c_1})^{a_1} g^{a_1 \cdot \gamma'} v_2^r g^{z_2},$$

$$D_5 = (g^b)^{-z_2},\ D_6 = g^{r_2 \cdot b},\ D_7 = g^{r_1},\ K = (u_i \cdot w)^{r_1},\ \forall_{j \neq i}\ K_j = u_j^{r_1}.$$

Challenge Ciphertext $\mathcal{B}$ receives a challenge set S^* and two message M_0, M_1 from the attacker. $\mathcal{B}$ will now create a challenge ciphertext that is a semi-functional ciphertext of either M_β or a random message, depending on T. It first chooses a random bit β.

$\mathcal{B}$ chooses random $s_1, t \in \mathbb{Z}_p$. It will implicitly let $s_2 = c_3$. The message $M_\beta \in \mathbb{G}_T$ is blinded as $C_0 = M_\beta \cdot T^{a_1 b}$ It then chooses random $x' \in \mathbb{Z}_p$ and will implicitly set $x = -c_3 + x'$.

$$C_1 = g^{s_1 \cdot b} + (g^{c_3})^b,\ C_2 = g^{b \cdot a_1 \cdot s_1},\ C_3 = g^{a_1 \cdot s_1},\ C_4 = (g^{c_2})^{x' \cdot b},\ C_5 = (g^{c_2})^{x'},$$

$$C_6 = \tau_1^{s_1} (g^{c_3})^{y_v} (g^{c_2})^{y_{v_2} \cdot x'},\ C_7 = (\tau_1^b)^{s_1} (g^{c_3})^{y_v \cdot b} (g^{c_2})^{y_{v_2} \cdot x' \cdot b} w^{-t},\ E_1 = (\prod_{i \in S} u_i)^t,\ E_2 = g^t.$$

If T is a tuple, then we are in $\mathbf{Game}_q$, otherwise we are in $\mathbf{Game}_{\text{Final}}$. $\mathcal{B}$ receives a bit β' and outputs 0 iff $\beta = \beta'$.

Theorem 3. *If the decisional Linear and decisional BDH assumptions holds, then no poly-time algorithm can break our Broadcast Encryption system.*

Proof. Any attacker's advantage in $\mathbf{Game}_{\text{Final}}\, \mathsf{Adv}_{\mathcal{A}}$ in the final game must be 0 since it completely hides the bit β. By the sequence of games we established and Lemmas 7,8,9 an attacker's advantage in the real game $\mathbf{Game}_{\text{Real}}\, \mathsf{Adv}_{\mathcal{A}}$ must be negligibly close to 0.

E IBE with Short Parameters from Subgroups

Setup The authority picks four primes p_1, p_2, p_3, p_4 and let's $N = p_1 p_2 p_3 p_4$. It then generates the group as $\mathcal{G}(N) \to \mathbb{G}$.

Next it chooses a random exponent $\alpha \in \mathbb{Z}_p$, random group elements: $g_{p_1}, u_{p_1}, h_{p_1}, w_{p_1}, \in \mathbb{G}_1$, and $X_{p_3} \in \mathbb{G}_3$. The public key is published as:

$$g_{p_1},\ u_{p_1},\ h_{p_1},\ w_{p_1},\ X_{p_3},\ e(g_{p_1}, g_{p_1})^\alpha$$

KeyGen($\mathcal{I}$) The authority chooses a random exponent $r \in Z_N$, a random key tag, tag $\in \mathbb{Z}_N$, and a random group element $R_{p_4} \in \mathbb{G}_4$. It produces the private key as:

$$D_0 = g_{p_1}^\alpha (u_{p_1}^{\mathcal{I}} h_{p_1})^r R_{p_4} \quad D_A = g_{p_1}^r \quad D_B = (u_{p_1} h_{p_1})^r (w_{p_1}^{\text{tag}})^r$$

Encrypt-KEM($\mathcal{I}$) The encryptor first generates random $R_{p_3}, R'_{p_3} \in \mathbb{G}_3$ (generated from X_{p_3} and $s \in \mathbb{Z}_N$ along with a random ciphertext "tag" value tag$'$ in $\mathbb{Z}_N$. The ciphertext is generated as:

$$C_0 = g_{p_1}^s R_{p_3} \quad C_1 = (u_{p_1}^{\mathcal{I}} h_{p_1})^s w_{p_1}^{\text{tag}' \cdot s} R'_{p_3}$$

The key extracted is:

$$K = e(g_{p_1}, g_{p_1})^{\alpha \cdot s}$$

Decrypt$(\mathrm{CT}, K_{\mathcal{I}})$ The decryption algorithm will work in the case when the private key tag $\mathrm{tag} \neq \mathrm{tag}' \pmod{p}_1$, where tag and tag$'$ are the private key and ciphertext tags respectively.

The decryption algorithm first computes:

$$T_1 = \left(e(C_1, D_A)^{\mathrm{tag}} e(C_0, D_B)^{-\mathrm{tag}'}\right)^{\frac{1}{\mathrm{tag}-\mathrm{tag}'}} = e(g_{p_1}, u_{p_1}^{\mathcal{I}} h_{p_1})^{rs}$$

Next, it computes

$$T_0 = e(C_0, D_0) = e(g_{p_1}, g_{p_1})^{\alpha \cdot s} e(g_{p_1}, u_{p_1}^{\mathcal{I}} h_{p_1})^{rs}$$

The KEM key can now be recovered by computing T_0/T_1.

E.1 Security Proof Sketch

We will show the security of the scheme based on a sequence of games. Intuitively, the proof will proceed as follows. The first game, $\mathbf{Game}_0$ is defined to be security game played on the real system. The next game $\mathbf{Game}_1$ has ciphertexts with the following distribution when create for challenge identity $\mathcal{I}^*$:

$$C_0 = g_{p_1}^s R_{p_3} R_{p_2} \quad C_1 = (u_{p_1}^{\mathcal{I}^*} h_{p_1})^s w_{p_1}^s R'_{p_3} R'_{p_2}$$

Notice they are identical to the ciphertexts given before with the exception that both C_0, C_1 have a random element of the $\mathbb{G}_{p_2}$ subgroup multiplied in. This will not effect decryption since all private keys (at this stage) do not have a component in the p_2 subgroup. We will argue that if a subgroup decision variant holds then no adversary can distinguish between $\mathbf{Game}_0$ and $\mathbf{Game}_1$. This is the relatively easy part of the reduction.

Next we define $\mathbf{Game}_i$ for $i = 1, \ldots, Q$, where Q is maximum number of private key queries made by the adversary. In these sequence of games we will slowly change the structure of the private keys given to the adversary. In particular, we will add a random element of the order p_2 subgroup to the private key component D_0. In game $\mathbf{Game}_i$ the challenge ciphertexts are generated the same way as in $\mathbf{Game}_1$. The private keys for the jth query where $j \geq i$ are generated as before; however, for $j < i$ the query of an idenity $\mathcal{I}$ generates a key as follows.

$$D_0 = g_{p_1}^{\alpha} (u_{p_1}^{\mathcal{I}} h_{p_1})^r R_{p_4} Y_{p_2} \quad D_A = g_{p_1}^r \quad D_B = (u_{p_1} h_{p_1})^r (w_{p_1}^{\mathrm{tag}})^r$$

Notice that these keys have the same distribution as the other keys except that a random element Y_{p_2} of $\mathbb{G}_2$ is multiplied into D_0.

We want to argue that no adversary can distinguish between $\mathbf{Game}_i$ and $\mathbf{Game}_{i+1}$. Intuitively, any private key of this new form will decrypt well-formed ciphertexts correctly. However, we might worry about how the key in question interacts with the challenge ciphertext. In particular, if the simulator can hand out a private key for any identity and also create a malformed challenge ciphertext for any identity $\mathcal{I}^*$, what would stop the simulator from learning the structure of the ith key by simply building the key for $\mathcal{I}^*$ itself. We resolve this paradox by building the reduction in the following manner. During setup the simulator will implicitly embed a degree one polynomial $f(x) = ax + b \pmod{N}$. In the reduction the simulator will create keys for $j > i$ or $j < i$ as fits either $\mathbf{Game}_i$ or $\mathbf{Game}_{i+1}$. However, for the challenge ciphertext it will need to form it such that the challenge ciphertext tag tag$'$ follows $\mathrm{tag}' = f(\mathcal{I}^*)$. Moreover, when the i-th query is for $\mathcal{I}$ the private key has $\mathrm{tag} = f(\mathcal{I})$. The simulator will not be able to attempt to decrypt the challenge ciphertext with the key in question since for $\mathcal{I}^*$ the tags would be the same.

Indistinguishability of Games 0 and 1 We begin by defining the extended subgroup assumption as follows.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving extended subgroup decision problem in a composite group $\mathbb{G}$ if

$$\Big| \Pr\left[\mathcal{B}\big(g_{p_1} \in G_{p_1}, X_{p_3} \in G_{p_3}, X_{p_4}, \in G_{p_4}, \quad T = Y_{p_3} \in \mathbb{G}_{p_3} = 0\right] - \\ \Pr\left[\mathcal{B}\big(g_{p_1} \in G_{p_1}, X_{p_3} \in G_{p_3}, X_{p_4}, \in G_{p_4}, \quad T = Y_{p_3} Y_{p_2} \in \mathbb{G}_{p_3 \times p_2}\big) = 0\right] \Big| \geq \epsilon$$

The problem basically states that if we are given generators in the p_1, p_3, p_4, but not p_2 subgroup then we cannot distinguish a random element in the $\mathbb{G}_{p_3}$ subgroup from the $\mathbb{G}_{p_3 \times p_2}$ subgroup. If we note that the $p_3 \times p_2$ subgroup is simply embedded in a larger group one can see this is actually equivalent to the standard decision subgroup assumption.

In the reduction the public key is created by choosing random $\alpha, a, b, c \in \mathbb{Z}_N$ and setting

$$g_{p_1},\ u_{p_1} = g_{p_1}^a,\ h_{p_1} = g_{p_1}^a,\ w_{p_1} = g_{p_1}^a,\ X_{p_3},\ e(g_{p_1}, g_{p_1})^\alpha$$

For each new private key for an $\mathcal{I}$ we choose fresh random t, r. Then create random $R_{p_4} = X_{p_4}^t$ and generate the key as:

$$D_0 = g_{p_1}^\alpha (u_{p_1}^{\mathcal{I}} h_{p_1})^r R_{p_4} \quad D_A = g_{p_1}^r \quad D_B = (u_{p_1} h_{p_1})^r (w_{p_1}^{\text{tag}})^r$$

The challenge ciphertext is created by choosing random $s, z \in Z_N$ and constructing:

$$C_0 = g_{p_1}^s X_{p_3} T \quad C_1 = (u_{p_1}^{\mathcal{I}} h_{p_1})^s w_{p_1}^s T^z$$

Finally, the derived key is $e(g_{p_1}, g_{p_1})^{\alpha s}$.

If T is random in the group G_3 then we are in **Game**0; otherwise we are in game **Game**$_1$.

Indistinguishability of Game$_i$ and Game$_{i+1}$ We now need to define a new Assumption1.

We begin by defining the extended subgroup assumption as follows.

An algorithm $\mathcal{B}$ that outputs $z \in \{0,1\}$ has advantage ϵ in solving our new Assumption1 in a composite group $\mathbb{G}$ if

$$\Big| \Pr\left[\mathcal{B}\big(g_{p_1}, w_{p_1} \in G_{p_1}, X_{p_3} \in G_{p_3}, X_{p_4}, \in G_{p_4}, g_{p_1}^t \in G_{p_1}, R_{p_2} R_{p_4}, g_{p_1}^s X_{p_2}, \quad T = u_{p_1}^t Y_{p_4}\right] - \\ \Pr\left[\mathcal{B}\big(g_{p_1}, w_{p_1} \in G_{p_1}, X_{p_3} \in G_{p_3}, X_{p_4}, \in G_{p_4}, g_{p_1}^t \in G_{p_1}, R_{p_2} R_{p_4}, g_{p_1}^s X_{p_2}, \quad T = u_{p_1}^t Y_{p_4} Y_{p_2}\big) = 0\right] \Big| \geq \epsilon$$

In the reduction the public key is created by choosing random $\alpha, a, b, c, d \in \mathbb{Z}_N$ and setting

$$g_{p_1},\ u_{p_1} = w_{p_1}^a g_{p_1}^c,\ h_{p_1} = w_{p_1}^b g_{p_1}^d,\ w_{p_1} = w_{p_1}\ X_{p_3},\ e(g_{p_1}, g_{p_1})^\alpha = e(g_{p_1}, g_{p_1}^s X_{p_2})^\alpha$$

At this point we define $f(\mathcal{I}) = -(a\mathcal{I} + b)$. The tag for the challenge ciphertext is set as $\text{tag}' = f(\mathcal{I}^*)$. The challenge ciphertext is created for $\mathcal{I}^*$ by choosing random $z_1, z_2 \in Z_N$ and constructing:

$$C_0 = (g_{p_1}^s X_{p_2}) X_{p_3}^{z_1} \quad C_1 = (g_{p_1}^s X_{p_2})^{c\mathcal{I}^* + d} X_{p_3}^{z_2}$$

The derived key is $e(g_{p_1}^s X_{p_2}, g_{p_1})^\alpha$. Notice that the choice of the tag canceled out the $w_{p_1}^s$ terms. Also note that C_1 and C_2 have indepedently distributed components of $\mathbb{G}_{p_2}, \mathbb{G}_{p_3}$ (this is do do the fact that c, d are random mod N and not just mod p_1).

Next, we generate keys for q j-th query where $j > i$. The simulator chooses a random tag. For each new private key for an $\mathcal{I}$ we choose fresh random t, r. Then create random $Z = X_{p_4}^t$ and generates the key as:

$$D_0 = g_{p_1}^\alpha (u_{p_1}^{\mathcal{I}} h_{p_1})^r Z \quad D_A = g_{p_1}^r \quad D_B = (u_{p_1} h_{p_1})^r (w_{p_1}^{\text{tag}})^r$$

Now, we generate keys for q j-th query where $j < i$. These are the keys that have already been "transformed" (and are known to be). The simulator chooses a random tag. These keys must have a random blinding component in both the $\mathbb{G}_{p_2}$ and $\mathbb{G}_{p_4}$ subgroups in D_0. For each new private key for an $\mathcal{I}$ we choose fresh random t, r. Then create random $Z = (R_{p_2} R_{p_4})^t$ and generates the key as:

$$D_0 = g_{p_1}^\alpha (u_{p_1}^{\mathcal{I}} h_{p_1})^r Z \quad D_A = g_{p_1}^r \quad D_B = (u_{p_1} h_{p_1})^r (w_{p_1}^{\text{tag}})^r$$

Finally, we come to generating the "challenge key" for the i-th query. The form of this key will depend upon the output from the assumption. The tag for a call to $\mathcal{I}$ is set as $\text{tag} = -f(\mathcal{I})$. The key is created as follows.

$$D_0 = g_{p_1}^\alpha (g_{p_1}^t)^{c\mathcal{I}+d} T^{a\mathcal{I}+b} \quad D_A = g_{p_1}^t \quad D_B = (g_{p_1}^t)^{c\mathcal{I}+d}$$

Notice, that we again use the tag for a cancellation.

If T is from the first distribution then we are in $\mathbf{Game}_i$; otherwise, we are in game $\mathbf{Game}_{i+1}$.

Finishing it Off We sketch the final part of the proof. Essentially, we need that given $g_{p_1}^\alpha R_{p_2}$ and $g_{p_1}^s X_{p_2}$ it is hard to distinguish $e(g_{p_1}, g_{p_1})^{\alpha s}$ from a random group element of $\mathbb{G}_T$. This seems pretty straightforward.