New Techniques for Dual System Encryption and Fully Secure HIBE with Short Ciphertexts

Allison Lewko* and Brent Waters**

University of Texas, Austin
alewko@cs.utexas.edu, bwaters@cs.utexas.edu

Abstract. We construct a fully secure HIBE scheme with short ciphertexts. The previous construction of Boneh, Boyen, and Goh was only proven to be secure in the selective model, under a non-static assumption which depended on the depth of the hierarchy. To obtain full security, we apply the dual system encryption concept recently introduced by Waters. A straightforward application of this technique is insufficient to achieve short ciphertexts, since the original instantiation of the technique includes tags that do not compress. To overcome this challenge, we design a new method for realizing dual system encryption. We provide a system in composite order groups (of three primes) and prove the security of our scheme under three static assumptions.

1 Introduction

An IBE system is a public key system where an encryptor uses only the identity of the recipient and a set of global public parameters, so a separate public key for each entity is not required. A trusted authority holds a master secret key which allows it to create secret keys for identities and distribute them to authenticated users. A Hierarchical IBE system (HIBE) [1,2] provides more functionality by forming levels of an organizational hierarchy. A user at level k can delegate secret keys to descendant identities at lower levels, but cannot decrypt messages intended for a recipient that is not among its descendants. For example, a user with the identity "University of Texas: computer science department" can delegate a key for the identity "University of Texas: computer science department: grad student", but cannot delegate keys for identities that do not begin with "University of Texas : computer science department". A more formal definition of an HIBE system is given in Section 2.

Most previous HIBE constructions were proven secure in the selective model of security (where an attacker must declare the identity he intends to attacker before seeing the public parameters of the system), with two recent exceptions. Gentry and Halevi [3] employ the techniques of [4] to obtain full security, but at

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-3-642-11799-2_36

[*] Supported by National Defense Science and Engineering Graduate Fellowship.

[**] Supported by NSF CNS-0716199, CNS-0915361 and Air Force Office of Scientific Research (AFO SR) under the MURI award for "Collaborative policies and assured information sharing" (Project PRESIDIO).

D. Micciancio (Ed.): TCC 2010, LNCS 5978, pp. 455–479, 2010.

the cost of a strong assumption (the BDHE-Set assumption) and ciphertext size growing linearly in the depth of the hierarchy. Waters [5] obtained full security with his new dual system encryption methodology from the well-established d-BDH and decisional Linear assumptions, but also had ciphertexts with size growing linearly in the depth of the hierarchy. This fell short of the constant size ciphertexts achieved by Boneh, Boyen, and Goh [6], but their HIBE system was only proven to be selectively secure in the standard model (or fully secure in the random oracle model).

In this paper, we resolve the question of whether full security and short ciphertexts (like [6]) can be simultaneously achieved in a HIBE system. A natural approach is to combine the Waters realization of dual system encryption with the Boneh-Boyen-Goh construction. This direct combination presents two problems:

1. Tags for each level that do not compress
2. Keys that are not fully rerandomized at delegation.

In the Boneh-Boyen-Goh system, group elements corresponding to each level of an identity are compressed (multiplied together) into a constant number of ciphertext elements. The tags in the Waters system do not allow this. These tags also prevent a key from being fully rerandomized upon delegation, meaning that an attacker can tell the difference between a delegated key and one freshly generated by the key generation algorithm. This requires a security definition that keeps track of such subtleties, which substantially complicates the security proof. Removing the tags from the Waters realization of dual system encryption is a nontrivial task because the tags were used to avoid a potential paradox in the dual system proof strategy.

1.1 Our Approach

We develop a new realization of dual system encryption that does not use tags. This provides several benefits:

1. compression of ciphertext is now possible
2. negligible correctness error caused by the tags is removed
3. schemes appear very natural and closely related to prior schemes.

Before giving the details of our approach, we first review the concept of dual system encryption.

Dual System Encryption. In a dual system, ciphertexts and keys can take on two forms: normal or semi-functional. Semi-functional ciphertexts and keys are not used in the real system, they are only used in the security proof. A normal key can decrypt normal or semi-functional ciphertexts, and a normal ciphertext can be decrypted by normal or semi-functional keys. However, when a semi-functional key is used to decrypt a semi-functional ciphertext, decryption will fail. More specifically, the semi-functional components of the key and ciphertext will interact to mask the blinding factor by an additional random term. Security for dual systems is proved using a sequence of games which are shown to be indistinguishable. The first game is the real security game (with normal ciphertext and keys). In the next game, the ciphertext is semi-functional, while all the keys are normal. For an attacker that makes q key requests, games 1 through q follow. In game k, the first k keys are semi-functional while the remaining keys are normal. In game q, all the keys and the challenge ciphertext given to the attacker are semi-functional. Hence none of the given keys are useful for decrypting the challenge ciphertext. At this point, proving security becomes relatively easy.

The Waters Realization. When arguing that games k and $k-1$ are indistinguishable, we create a simulator who can use any legal identities for the challenge ciphertext and keys. This creates a potential problem. The simulator is prepared to make a semi-functional ciphertext for an identity ID and is also prepared to make the k^{th} key for identity ID, so it may seem like the simulator can determine whether key k is semi-functional for itself by test decrypting with a semi-functional ciphertext for the same identity. To resolve this paradox, the Waters IBE scheme associates random tag values with each ciphertext and key. Decryption works only when the tag values of the ciphertext and decrypting key are unequal. If the simulator attempted to test semi-functionality of key k for itself by creating a semi-functional ciphertext for the same identity, it would only be able to create one with an equal tag, and hence decryption would *unconditionally fail.* This correlation of tags is hidden from an attacker who cannot request a key with the same identity as the challenge ciphertext, so the tags look randomly distributed from the attacker's point of view.

Tags are used similarly in the Waters HIBE scheme, but here they cause two additional problems. First, there is a separate tag value associated with each level of the identity in a ciphertext or key. All these tag values must be given out in a ciphertext, so this forces ciphertext size to grow linearly with the depth of the hierarchy. Secondly, there is no method for rerandomizing the tags in key delegation. This means that a key at level $d+1$ which is delegated from a key at level d will share its first d tag values, a property which links the distribution of a key to its lineage. Some previous security definitions for HIBE [1, 2] which did not keep track of delegation paths of keys are hence invalid for such a system. Security must be argued under a more complete definition introduced in [7].

Our Realization. The additional complications of the proof and the linear ciphertext size are undesirable artifacts of building the HIBE system with the same tag techniques as the IBE system. To remove the tags, we must find a different way to resolve the paradox. Instead of having decryption unconditionally fail when the simulator attempts to test semi-functionality of the k^{th} key, we design our system so that decryption will *unconditionally succeed.* We introduce a variant of semi-functional keys which we call *nominally* semi-functional keys. These keys are semi-functional in name only, meaning that they are distributed like semi-functional keys, but are actually correlated with semi-functional ciphertexts so that when a nominally semi-functional key is used to decrypt a semi-functional ciphertext, the interaction of the two semi-functional components results in cancelation and decryption is successful. If the simulator attempts to answer its own question by creating the k^{th} key and challenge ciphertext for the same identity, the created key will be nominally semi-functional and hence test decrypting will not distinguish this from a normal key. This nominally semi-functional key will appear to be distributed like a regular semi-functional key to the attacker, who cannot request a key that can decrypt the challenge ciphertext.

With this technique, we are able to construct a fully secure IBE system with short parameters without tags, and also give a fully secure HIBE system with constant-size ciphertexts. Our proofs rely on simple (constant-size) assumptions which do not depend on the number of queries the attacker makes. Our proof for our HIBE system is considerably simplified by the fact that our keys can be fully rerandomized upon delegation, avoiding the corresponding difficulties of the Waters HIBE proof.

In our the main body we provide a construction under a group of composite order N where N is the product of three primes. In Appendix C, we provide an analog of this for prime order groups. Our analog takes advantage of asymmetric bilinear groups where there is no efficient isomorphism between G_1 and G_2.

An interesting observation arising from our work is that the existing Boneh-Boyen IBE [8] and Boneh-Boyen-Goh HIBE [6] schemes which were only proven to be selectively secure can be transformed into fully secure systems by embedding them in composite order groups. Our IBE and HIBE systems are *remarkably* similar to these schemes.

1.2 Related Work

Identity Based Encryption was introduced by Shamir [9] and first realized by Boneh and Franklin [10] and Cocks [11]. The Boneh-Franklin IBE construction [10] proved security in the random oracle model. Subsequent constructions by Canetti, Halevi, and Katz [12] and Boneh and Boyen [8] were proved secure in the standard model, but under the weaker notion of selective security. Later, Boneh and Boyen [13] and Waters [14] gave constructions which were fully secure in the standard model. The Waters system was efficient and fully secure in the standard model under the decisional Bilinear Diffie-Hellman assumption (d-BDH), but it had public parameters consisting of $\mathcal{O}(\lambda)$ group elements for security parameter λ. Gentry [4] constructed an IBE system with short public parameters and proved full security in the standard model, but used an assumption (q-ABHDE) which is substantially more complicated than d-BDH and depends on the number of queries made by the attacker. Gentry, Peikert, and Vaikuntanathan also gave an IBE construction based on lattice assumptions [15].

Hierarchical Identity Based Encryption was introduced by Horwitz and Lynn [2] and then constructed by Gentry and Silverberg [1] in the random oracle model. Boneh and Boyen [8] achieved security in the selective model without random oracles. Boneh, Boyen, and Goh [6] then gave an HIBE with constant size ciphertexts, also in the selective model under a q-based assumption. These short ciphertexts were particularly useful for applications, including forward secure encryption [12] and converting the NNL broadcast encryption system [16] into a public-key system [17]. Gentry and Halevi [3] constructed the first fully secure HIBE for polynomial depth, though also under a complex assumption. Waters [5] attained full security under the d-BDH and decisional Linear assumptions, but with ciphertext size growing linearly in the hierarchy depth. We note that Waters first instantiated this result in composite order groups. The complete definition of security for HIBE that we use in this paper was formulated by Shi and Waters [7].

1.3 Organization

In Section 2, we formally define an HIBE system and give the complete security definition, give background on bilinear groups, and state our assumptions. In Section 3, we present our IBE scheme and prove its security. In Section 4, we give our HIBE scheme and prove its security. In Section 6, we conclude and discuss open directions for further research.

2 Background

2.1 Hierarchical Identity Based Encryption

A Hierarchical Identity Based Encryption scheme has five algorithms: Setup, Encrypt, KeyGen, Decrypt, and Delegate.

Setup$(\lambda) \rightarrow PK, MSK.$ The setup algorithm takes a security parameter λ as input and outputs the public parameters PK and a master secret key MSK.

KenGen$(MSK, \boldsymbol{I}) \rightarrow SK_{\boldsymbol{I}}.$ The key generation algorithm takes the master secret key and an identity vector $\boldsymbol{I}$ as input and outputs a private key $SK_{\boldsymbol{I}}$.

Delegate$(PK, SK_{\boldsymbol{I}}, I) \rightarrow SK_{\boldsymbol{I}:I}.$ The delegation algorithm takes a secret key for the identity vector $\boldsymbol{I}$ of depth d and an identity I as input and outputs a secret key for the depth $d+1$ identity vector $\boldsymbol{I} : I$ formed by concatenating I onto the end of $\boldsymbol{I}$.

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

Decrypt$(PK, CT, SK) \rightarrow M.$ The decryption algorithm takes the public parameters PK, a ciphertext CT, and a secret key SK as input and outputs the message M, if the ciphertext was an encryption to an identity vector $\boldsymbol{I}$ and the secret key is for the same identity vector.

Notice that the decryption algorithm is only required to work when the identity vector for the ciphertext matches the secret key exactly. However, someone who has a secret key for a prefix of this identity vector can delegate to themselves the required secret key and also decrypt.

Security definition. We give the complete form of the security definition [7] which keeps track of how keys are generated and delegated. Security is defined through the following game, played by a challenger and an attacker.

Setup. The challenger runs the Setup algorithm to generate public parameters PK which it gives to the adversary. We let S denote the set of private keys that the challenger has created but not yet given to the adversary. At this point, $S = \emptyset$.

Phase 1. The adversary makes Create, Delegate, and Reveal key queries. To make a Create query, the attacker specifies an identity vector $\boldsymbol{I}$. In response, the challenger creates a key for this vector by calling the key generation algorithm, and places this key in the set S. It only gives the attacker a reference to this key, not the key itself. To make a Delegate query, the attacker specifies a key $SK_{\boldsymbol{I}}$ in the set S and specifies an identity I'. In response, the challenger appends I' to $\boldsymbol{I}$ and makes a key for this new identity by running the delegation algorithm on $SK_{\boldsymbol{I}}$ and I'. It adds this key to the set S and again gives the attacker only a reference to it, not the actual key. To make a Reveal query, the attacker specifies an element of the set S. The challenger gives this key to the attacker and removes it from the set S. We note that the attacker need no longer make any delegation queries for this key because it can run the delegation algorithm on the revealed key for itself.

Challenge. The adversary gives the challenger two messages M_0 and M_1 and a challenge identity vector $\boldsymbol{I}^*$. This identity vector must satisfy the property that no revealed identity in Phase 1 was a prefix of it. The challenger sets $\beta \in \{0, 1\}$ randomly, and encrypts M_β under $\boldsymbol{I}^*$. It sends the ciphertext to the adversary.

Phase 2. This is the same as Phase 1, with the added restriction that any revealed identity vector must not be a prefix of $\boldsymbol{I}^*$.

Guess. The adversary must output a guess β' for β.

The advantage of an adversary $\mathcal{A}$ is defined to be $Pr[\beta' = \beta] - \frac{1}{2}$.

Definition 1. *A Hierarchical Identity Based Encryption scheme is secure if all polynomial time adversaries achieve at most a negligible advantage in the security game.*

2.2 Composite Order Bilinear Groups

Composite order bilinear groups were first introduced in [18]. We define them by using a group generator $\mathcal{G}$, an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group G. In our case, $\mathcal{G}$ outputs $(N = p_1p_2p_3, G, G_T, e)$ where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1p_2p_3$, and $e : G^2 \to G_T$ is a map such that:

1. (Bilinear) $\forall g, h \in G, a, b \in \mathbb{Z}_N, e(g^a, h^b) = e(g, h)^{ab}$
2. (Non-degenerate) $\exists g \in G$ such that $e(g, g)$ has order N in G_T.

We further require that the group operations in G and G_T as well as the bilinear map e are computable in polynomial time with respect to λ. Also, we assume the group descriptions of G and G_T include generators of the respective cyclic groups. We let G_{p_1}, G_{p_2}, and G_{p_3} denote the subgroups of order p_1, p_2 and p_3 in G respectively. We note that when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $e(h_i, h_j)$ is the identity element in G_T. To see this, suppose $h_1 \in G_{p_1}$ and $h_2 \in G_{p_2}$. We let g denote a generator of G. Then, $g^{p_1 p_2}$ generates G_{p_3}, $g^{p_1 p_3}$ generates G_{p_2}, and $g^{p_2 p_3}$ generates G_{p_1}. Hence, for some α_1, α_2, $h_1 = (g^{p_2 p_3})^{\alpha_1}$ and $h_2 = (g^{p_1 p_3})^{\alpha_2}$. We note:

$$e(h_1, h_2) = e(g^{p_2 p_3 \alpha_1}, g^{p_1 p_3 \alpha_2}) = e(g^{\alpha_1}, g^{p_3 \alpha_2})^{p_1 p_2 p_3} = 1.$$

This orthogonality property of $G_{p_1}, G_{p_2}, G_{p_3}$ will be a principal tool in our constructions.

We now give our complexity assumptions. These same assumptions will be used to prove the security of our IBE and HIBE systems. We note that they are static (not dependent on the depth of the hierarchy or the number of queries made by an attacker). The first assumption is just the subgroup decision problem in the case where the group order is a product of 3 primes. In Appendix A, we show that these assumptions hold in the generic group model if finding a nontrivial factor of the group order is hard. We prove this by applying the theorems of Katz, Sahai, and Waters [19]. Their work also used composite order bilinear groups and provided a general framework for proving generic security of assumptions in this setting.

In the assumptions below, we let $G_{p_1 p_2}$, e.g., denote the subgroup of order $p_1 p_2$ in G.

Assumption 1 (Subgroup decision problem for 3 primes). Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g \xleftarrow{R} G_{p_1},\ X_3 \xleftarrow{R} G_{p_3},$$

$$D = (\mathbb{G}, g, X_3),$$

$$T_1 \xleftarrow{R} G_{p_1 p_2},\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

We note that T_1 can be written (uniquely) as the product of an element of G_{p_1} and an element of G_{p_2}. We refer to these elements as the "G_{p_1} part of T_1" and the "G_{p_2} part of T_1" respectively. We will use this terminology in our proofs.

Definition 2. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 2. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, X_1 \xleftarrow{R} G_{p_1},\ X_2, Y_2 \xleftarrow{R} G_{p_2},\ X_3, Y_3 \xleftarrow{R} G_{p_3},$$

$$D = (\mathbb{G}, g, X_1X_2, X_3, Y_2Y_3),$$

$$T_1 \xleftarrow{R} G,\ T_2 \xleftarrow{R} G_{p_1p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

We use $G_{p_1p_3}$ to denote the subgroup of order p_1p_3 in G. We note that T_1 can be (uniquely) written as the product of an element of G_{p_1}, an element of G_{p_2}, and an element of G_{p_3}. We refer to these as the "G_{p_1} part of T_1", the "G_{p_2} part of T_1", and the "G_{p_3} part of T_1", respectively. T_2 can similarly be written as the product of an element of G_{p_1} and an element of G_{p_3}.

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

Assumption 3. Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G}, \alpha, s \xleftarrow{R} \mathbb{Z}_N,$$

$$g \xleftarrow{R} G_{p_1}, X_2, Y_2, Z_2 \xleftarrow{R} G_{p_2}, X_3 \xleftarrow{R} G_{p_3},$$

$$D = (\mathbb{G}, g, g^{\alpha}X_2, X_3, g^sY_2, Z_2),$$

$$T_1 = e(g, g)^{\alpha s}, T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 4. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

3 Our IBE System

We begin by giving our new dual system encryption realization of IBE. Our construction will use composite order groups of order $N = p_1p_2p_3$ and identities in $\mathbb{Z}_N$. Remarkably, our construction looks almost exactly like the Boneh-Boyen IBE with keys additionally randomized in the subgroup G_{p_3}. This resemblance to preexisting selectively secure schemes will continue in our HIBE system as well. We regard this as a desirable feature of our approach.

We note that the subgroup G_{p_2} is not used in our actual scheme, instead it serves as our semi-functional space. Keys and ciphertexts will be semi-functional when they include terms in G_{p_2} and decryption will proceed by pairing key elements with ciphertext elements. This will give us the decryption functionality we need: when we pair a normal key with a semi-functional ciphertext or a normal ciphertext with a semi-functional key, the terms in G_{p_2} are orthogonal to terms in G_{p_1} and G_{p_3} under the pairing and will cancel out. When we pair a semi-functional key with a semi-functional ciphertext, we will get an additional term arising from the pairing of the terms in G_{p_2}.

3.1 Construction

Setup. The setup algorithm chooses a bilinear group G of order $N = p_1 p_2 p_3$ (where p_1, p_2, and p_3 are distinct primes). We let G_{p_i} denote the subgroup of order p_i in G. It then chooses $u, g, h \in G_{p_1}$ and $\alpha \in \mathbb{Z}_N$. The public parameters are published as:

$$PK = \{N, u, g, h, e(g, g)^\alpha\}.$$

The secret parameters are α and a generator of G_{p_3}.

Encrypt(M, ID). The encryption algorithm chooses $s \in \mathbb{Z}_N$ randomly and creates the ciphertext as:

$$C_0 = Me(g, g)^{\alpha s}, C_1 = (u^{ID} h)^s, C_2 = g^s.$$

KeyGen(ID, MSK). The key generation algorithm chooses $r \in \mathbb{Z}_N$ and R_3, $R_3' \in G_{p_3}$ randomly. (Random elements of G_{p_3} can be obtained by taking a generator of G_{p_3} and raising it to random exponents modulo N.) The key is formed as:

$$K_1 = g^r R_3, K_2 = g^\alpha (u^{ID} h)^r R_3'.$$

Decryption. If the ID's of the ciphertext and key are equal, the decryption algorithm computes the blinding factor as:

$$\frac{e(K_2, C_2)}{e(K_1, C_1)} = \frac{e(g, g)^{\alpha s} e(u^{ID} h, g)^{rs}}{e(u^{ID} h, g)^{rs}}.$$

3.2 Security

To prove security of our IBE system, we first define two additional structures: semi-functional keys and semi-functional ciphertexts. These will not be used in the real system, but they will be used in our proof.

Semi-functional Ciphertext. We let g_2 denote a generator of the subgroup G_{p_2}. A semi-functional ciphertext is created as follows: first, a normal ciphertext C_0', C_1', C_2' is generated by the encryption algorithm. Random exponents $x, z_c \in \mathbb{Z}_N$ are chosen. Then, C_0 is set to be C_0', C_1 is set to be $C_1' g_2^{x z_c}$, and C_2 is set to be $C_2' g_2^x$.

Semi-functional Key. A semi-functional key is created as follows: first, a normal key K_1', K_2' is generated by the key generation algorithm. Random exponents $\gamma, z_k \in \mathbb{Z}_N$ are chosen. K_1 is set to be $K_1' g_2^{\gamma}$ and K_2 is set to be $K_2' g_2^{\gamma z_k}$.

Notice that if a semi-functional key is used to decrypt a semi-functional ciphertext, the blinding factor will be obscured by an additional factor of $e(g_2, g_2)^{x\gamma(z_k - z_c)}$. If $z_c = z_k$, decryption will still work. In this case, we say that the key is *nominally* semi-functional: it has terms in G_{p_2}, but these do not hinder decryption.

Our proof of security relies on Assumptions 1, 2, 3 defined in Section 2. We will prove security by a hybrid argument using a sequence of games. The first game, Game_{Real}, will be the real security game. The next game, $\text{Game}_{Restricted}$, will be like the real security game except that the attacker cannot ask for keys for identities which are equal to the challenge identity modulo p_2. This is a stronger restriction than the real security game, where the identities must be unequal modulo N. We will retain this stronger restriction throughout the subsequent games. The reason for it will be explained in the proof. We let q denote the number of key queries the attacker makes. For k from 0 to q, we define Game_k as:

Game$_k$. This is like the restricted security game, except that the ciphertext given to the attacker is semi-functional and the first k keys are semi-functional. The rest of the keys are normal.

In Game_0, all the keys are normal and the ciphertext is semi-functional. In Game_q, the ciphertext and all of the keys are semi-functional. Our last game is Game_{Final}, which is the same as Game_q except that the ciphertext is a semi-functional encryption of a random message, not one of the two messages requested by the attacker. We will prove that each of these games is indistinguishable in the following four lemmas.

Lemma 1. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{Real}Adv_{\mathcal{A}} - Game_{Restricted}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\geq \frac{\epsilon}{2}$ in breaking either Assumption 1 or Assumption 2.*

Proof. Given g, X_3, $\mathcal{B}$ can simulate Game_{Real} with $\mathcal{A}$. With probability ϵ, $\mathcal{A}$ produces identities ID and ID^* such that $ID \neq ID^*$ modulo N and p_2 divides $ID - ID^*$. $\mathcal{B}$ uses these identities to produce a nontrivial factor of N by computing $a = \gcd(ID - ID^*, N)$. We set $b = \frac{N}{a}$. We note that p_2 divides a and $N = ab = p_1 p_2 p_3$. We consider two cases:

1. p_1 divides b
2. $a = p_1 p_2$ and $b = p_3$.

At least one of these cases must occur with probability $\geq \frac{\epsilon}{2}$. In case 1, $\mathcal{B}$ will break Assumption 1. Given g, X_3, T, $\mathcal{B}$ can determine that p_1 divides b by verifying that g^b is the identity and will then test whether T^b is the identity. If it is, then $T \in G_{p_1}$. If it is not, $T \in G_{p_1 p_2}$.

In case 2, $\mathcal{B}$ will break Assumption 2. Given g, X_1X_2, X_3, Y_2Y_3, $\mathcal{B}$ can determine that $a = p_1 p_2$ by verifying that $(X_1X_2)^a$ is the identity and will then test whether $e((Y_2Y_3)^b, T)$ is the identity. If it is, then $T \in G_{p_1p_3}$. If it is not, then $T \in G$.

Lemma 2. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{Restricted}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ first receives g, X_3, T. It simulates $\text{Game}_{Restricted}$ or Game_0 with A. It sets the public parameters as follows. It chooses random exponents $\alpha, a, b \in \mathbb{Z}_N$ and sets $g = g, u = g^a$, $h = g^b$. It sends these public parameters $\{N, u, g, h, e(g,g)^\alpha\}$ to $\mathcal{A}$. Each time $\mathcal{B}$ is asked to provide a key for an identity ID_i, it chooses random exponents r_i, t_i, and $w_i \in \mathbb{Z}_N$ and sets:

$$K_1 = g^{r_i} X_3^{t_i}, K_2 = g^\alpha (u^{ID_i} h)^{r_i} X_3^{w_i}.$$

$\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, ID. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly. The ciphertext is formed as follows:

$$C_0 = M_\beta e(T,g)^\alpha, C_1 = T^{aID+b}, C_2 = T.$$

(This implicitly sets g^s equal to the G_{p_1} part of T.) If $T \in G_{p_1p_2}$, then this is a semi-functional ciphertext with $z_c = aID + b$. We note that the value of z_c modulo p_2 is not correlated with the values of a and b modulo p_1, so this is properly distributed. If $T \in G_{p_1}$, this is a normal ciphertext. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

Lemma 3. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{k-1}Adv_{\mathcal{A}} - Game_kAdv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ first receives $g, X_1X_2, X_3, Y_2Y_3, T$. $\mathcal{B}$ picks random exponents $a, b, \alpha \in \mathbb{Z}_N$ and sets the public parameters as: $g = g, u = g^a, h = g^b, e(g,g)^\alpha$. It sends these to $\mathcal{A}$. When $\mathcal{A}$ requests the i^{th} key for ID_i when $i < k$, $\mathcal{B}$ creates a semi-functional key. It does this by choosing random exponents r_i, z_i, $t_i \in \mathbb{Z}_N$ and setting:

$$K_1 = g^{r_i}(Y_2Y_3)^{t_i}, K_2 = g^\alpha (u^{ID_i} h)^{r_i} (Y_2Y_3)^{z_i}.$$

This is a properly distributed semi-functional key with $g_2^\gamma = Y_2^{t_i}$. (We note that the values of t_i and z_i modulo p_2 and modulo p_3 are uncorrelated by the Chinese Remainder Theorem.)

For $i > k$, $\mathcal{B}$ generates normal keys by using random exponents $r_i, t_i, w_i \in \mathbb{Z}_N$ and setting:

$$K_1 = g^{r_i} X_3^{t_i}, K_2 = g^\alpha (u^{ID_i} h)^{r_i} X_3^{w_i}.$$

To create the k^{th} requested key, $\mathcal{B}$ lets $z_k = aID_k + b$, chooses a random exponent $w_k \in \mathbb{Z}_N$, and sets:

$$K_1 = T, K_2 = g^\alpha T^{z_k} X_3^{w_k}.$$

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, ID. $\mathcal{B}$ sets $\beta \in \{0, 1\}$ randomly. The challenge ciphertext is formed as:

$$C_0 = M_\beta e(X_1X_2, g)^\alpha, C_1 = (X_1X_2)^{aID+b}, C_2 = X_1X_2.$$

We note that this sets $g^s = X_1$ and $z_c = aID + b$. Since $f(ID) = aID + b$ is a pairwise independent function modulo p_2, as long as $ID_k \neq ID (\bmod p_2)$, z_k and z_c will seem randomly distributed to $\mathcal{A}$ (again, we note that the values of a and b modulo p_2 are uncorrelated with their values modulo p_1). If $ID_k \equiv ID (\bmod p_2)$, then $\mathcal{A}$ has made an invalid key request. This is where we use our additional modular restriction.

Though it is hidden from $\mathcal{A}$, this relationship between z_c and z_k is crucial: if $\mathcal{B}$ attempts to test itself whether key k is semi-functional by creating a semi-functional ciphertext for ID_k and trying to decrypt, then decryption will work whether key k is semi-functional or not, because $z_c = z_k$. In other words, the simulator $\mathcal{B}$ can only make a nominally semi-functional key k.

If $T \in G_{p_1p_3}$, then $\mathcal{B}$ has properly simulated Game$_{k-1}$. If $T \in G$, then $\mathcal{B}$ has properly simulated Game$_k$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

Lemma 4. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_qAdv_{\mathcal{A}} - Game_{Final}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ first receives $g, g^\alpha X_2, X_3, g^sY_2, Z_2, T$. $\mathcal{B}$ chooses random exponents $a, b \in \mathbb{Z}_N$ and sets the public parameters as $g = g, u = g^a, h = g^b, e(g, g)^\alpha = e(g^\alpha X_2, g)$. It sends these to $\mathcal{A}$. When $\mathcal{A}$ requests a key for identity ID_i, $\mathcal{B}$ generates a semi-functional key. It does this by choosing random exponents $c_i, r_i, t_i, w_i, \gamma_i \in \mathbb{Z}_N$ and setting:

$$K_1 = g^{r_i}Z_2^{\gamma_i}X_3^{t_i}, K_2 = g^\alpha X_2(u^{ID_i}h)^{r_i}Z_2^{c_i}X_3^{w_i}.$$

$\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, ID. $\mathcal{B}$ sets $\beta \in \{0, 1\}$ randomly. It forms the challenge ciphertext as:

$$C_0 = M_\beta T, C_1 = (g^sY_2)^{aID+b}, C_2 = g^sY_2.$$

This sets $z_c = aID + b$. We note that the value of z_c only matters modulo p_2, whereas $u = g^a$ and $h = g^b$ are elements of G_{p_1}, so when a and b are chosen randomly modulo N, there is no correlation between the values of a and b modulo p_1 and the value $z_c = aID + b$ modulo p_2.

If $T = e(g, g)^{\alpha s}$, then this is a properly distributed semi-functional ciphertext with message M_β. If T is a random element of G_T, then this is a semi-functional ciphertext with a random message. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

We have now proven the following theorem:

Theorem 1. *If Assumptions 1, 2, and 3 hold, then our IBE system is secure.*

Proof. If Assumptions 1, 2, and 3 hold, then we have shown by the previous lemmas that the real security game is indistinguishable from Game_{Final}, in which the value of β is information-theoretically hidden from the attacker. Hence the attacker can attain no advantage in breaking the IBE system.

4 Our HIBE System

We build upon our IBE system and extend our techniques to give an HIBE system with short ciphertexts. The absence of tags allows us to compress the ciphertext into a constant number of group elements and also to rerandomize keys fully upon delegation. This dramatically simplifies our proof of security. Our construction again uses composite order groups of order $N = p_1p_2p_3$, and looks almost exactly like the Boneh-Boyen-Goh HIBE system with keys additionally randomized in subgroup G_{p_3}. G_{p_2} will be our semi-functional space, which is not used in the real system.

4.1 Construction

Setup. The setup algorithm chooses a bilinear group G or order $N = p_1p_2p_3$. We let ℓ denote the maximum depth of the HIBE. The setup algorithm chooses $g, h, u_1, \ldots, u_\ell \in G_{p_1}$, $X_3 \in G_{p_3}$, and $\alpha \in \mathbb{Z}_N$. The public parameters are published as:

$$PK = \{N, g, h, u_1, \ldots, u_\ell, X_3, e(g,g)^\alpha\}.$$

The secret parameter is α.

Encrypt($M, (ID_1, \ldots, ID_j)$). The encryption algorithm chooses $s \in Z_N$ randomly. It sets:

$$C_0 = Me(g,g)^{\alpha s}, C_1 = \left(u_1^{ID_1} \cdots u_j^{ID_j} h\right)^s, C_2 = g^s.$$

KeyGen($MSK, (ID_1, \ldots, ID_j)$). The key generation algorithm chooses $r \in \mathbb{Z}_N$ randomly and also chooses random elements $R_3, R_3', R_{j+1}, \ldots, R_\ell$ of G_{p_3}. It sets:

$$K_1 = g^r R_3, K_2 = g^\alpha \left(u_1^{ID_1} \cdots u_j^{ID_j} h\right)^r R_3', E_{j+1} = u_{j+1}^r R_{j+1}, \ldots, E_\ell = u_\ell^r R_\ell.$$

Delegate. Given a key $K_1', K_2', E_{j+1}', \ldots, E_\ell'$ for $(ID_1, \ldots, ID_j)$, the delegation algorithm creates a key for $(ID_1, \ldots, ID_{j+1})$ as follows. It chooses a random $r' \in \mathbb{Z}_N$ and random elements of G_{p_3} denoted, e.g., by $\tilde{R}_3$. The new key is set as:

$$K_1 = K_1' g^{r'} \tilde{R}_3,$$

$$K_2 = K_2' \left(u_1^{ID_1} \cdots u_j^{ID_j} h\right)^{r'} (E_{j+1}')^{ID_{j+1}} u_{j+1}^{r' ID_{j+1}} \tilde{R}_3',$$

$$E_{j+2} = E_{j+2}' u_{j+2}^{r'} \tilde{R}_{j+2}, \ldots, E_\ell = E_\ell' u_\ell^{r'} \tilde{R}_\ell.$$

We note that this new key is fully rerandomized: its only tie to the previous key is in the values $ID_1, \ldots, ID_j$.

Decrypt. The decryption algorithm assumes that the key and ciphertext both correspond to the same identity $(ID_1, \ldots, ID_j)$. If the key identity is a prefix of this instead, then the decryption algorithm starts by running the key delegation algorithm to create a key with identity matching the ciphertext identity exactly. The decryption algorithm then computes the blinding factor as:

$$\frac{e(K_2, C_2)}{e(K_1, C_1)} = \frac{e(g,g)^{\alpha s} e(u_1^{ID_1} \cdots u_j^{ID_j} h, g)^{rs}}{e(g, u_1^{ID_1} \cdots u_j^{ID_j} h)^{rs}} = e(g,g)^{\alpha s}.$$

4.2 Security

To prove security of our HIBE system, we again rely on the static Assumptions 1, 2, and 3. We first define two additional structures: semi-functional ciphertexts and semi-functional keys. These will not be used in the real system, but will be used in our proof.

Semi-functional Ciphertext. We let g_2 denote a generator of G_{p_2}. A semi-functional ciphertext is created as follows: first, we use the encryption algorithm to form a normal ciphertext C'_0, C'_1, C'_2. We choose random exponents $x, z_c \in \mathbb{Z}_N$. We set:

$$C_0 = C'_0, C_1 = C'_1 g_2^{x z_c}, C_2 = C'_2 g_2^x.$$

Semi-functional Keys. To create a semi-functional key, we first create a normal key K'_1, K'_2, E'_{j+1}, ..., E'_ℓ using the key generation algorithm. We choose random exponents $\gamma, z_k, z_{j+1}, \ldots, z_\ell \in \mathbb{Z}_N$. We set:

$$K_1 = K'_1 g_2^{\gamma}, K_2 = K'_2 g_2^{\gamma z_k}, E_{j+1} = E'_{j+1} g_2^{\gamma z_{j+1}}, \ldots, E_\ell = E'_\ell g_2^{\gamma z_\ell}.$$

We note that when a semi-functional key is used to decrypt a semi-functional ciphertext, the decryption algorithm will compute the blinding factor multiplied by the additional term $e(g_2, g_2)^{x\gamma(z_k - z_c)}$. If $z_c = z_k$, decryption will still work. In this case, the key is nominally semi-functional.

Our proof of security will again be structured as a hybrid argument over a sequence of games. The first game, Game_{Real}, is the real HIBE security game. The next game, $\text{Game}_{Real'}$, is the same as the real game except that all key queries will be answered by fresh calls to the key generation algorithm (the challenger will not be asked to delegate keys in a particular way). The next game, $\text{Game}_{Restricted}$ is the same as $\text{Game}_{Real'}$ except that the attacker cannot ask for keys for identities which are prefixes of the challenge identity modulo p_2. We will retain this restriction in all subsequent games. We let q denote the number of key queries the attacker makes. For k from 0 to q, we define Game_k as:

$Game_k$. This is like $\text{Game}_{Restricted}$, except that the ciphertext given to the attacker is semi-functional and the first k keys are semi-functional. The rest of the keys are normal.

In Game_0, only the challenge ciphertext is semi-functional. In Game_q, the challenge ciphertext and all of the keys are semi-functional. We define Game_{Final} to be like Game$_q$, except that the challenge ciphertext is a semi-functional encryption of a random message, not one of the messages provided by the attacker. We will show these games are indistinguishable in five lemmas. The proofs are very similar to the proofs for our IBE system, and can be found in Appendix B.

5 Moving to Prime Order Groups

In Appendix C we show an analog of our previous construction in prime order groups. The prime order group construction we give takes advantage of asymmetric groups where there is a pairing function $e : G_1 \times G_2 \rightarrow G_T$, but there is not believed to be an efficient isomorphism from either G_1 to G_2 or G_2 to G_1.

Our prime construction can be viewed as an analog of the composite order one where we "emulate" the three subgroups with multiple group elements to create three subspaces. Our "emulation" technique uses some ideas from the Waters [5] prime order group realization; however, we are able to "squeeze" things down by using asymmetric groups.

A potential future direction is be to realize our methods in prime order groups without relying on the lack of isomorphism for security. A natural approach would be to use an "unsqueezed" version of our techniques. It is possible that this approach might give a reduction with more cancelations that in turn provides security from even simpler assumptions.

6 Conclusions and Open Directions

We have given the first HIBE system with constant size ciphertext that is fully secure in the standard model from simple assumptions. In doing so, we discovered that instantiations of the selectively secure Boneh-Boyen IBE and Boneh-Boyen-Goh HIBE schemes in composite order bilinear groups can be proved to be fully secure using the dual encryption technique of Waters. We overcame the initial challenges introduced by the use of tags in the original Waters IBE and HIBE systems by introducing the concept of nominally semi-functional keys. Our work further demonstrates the power and versatility of the dual system encryption technique, which we believe will have many future applications.

We leave it as an open problem to transfer our IBE and HIBE systems into prime order groups with security proven from standard assumptions such as the decisional Linear assumption and d-BDH. This kind of translation was previously achieved by Waters [5] for his IBE and HIBE systems, which were originally constructed in composite order groups.

References

1. Gentry, C., Silverberg, A.: Hierarchical id-based cryptography. In: Zheng, Y. (ed.) ASIACRYPT 2002. LNCS, vol. 2501, pp. 548–566. Springer, Heidelberg (2002)
2. Horwitz, J., Lynn, B.: Toward hierarchical identity-based encryption. In: Knudsen, L.R. (ed.) EUROCRYPT 2002. LNCS, vol. 2332, pp. 466–481. Springer, Heidelberg (2002)

3. Gentry, C., Halevi, S.: Hierarchical identity based encryption with polynomially many levels. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 437–456. Springer, Heidelberg (2009)
4. Gentry, C.: Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 445–464. Springer, Heidelberg (2006)
5. Waters, B.: Dual system encryption: realizing fully secure ibe and hibe under simple assumptions. In: Halevi, S. (ed.) CRYPTO 2009. LNCS, vol. 5677, pp. 619–636. Springer, Heidelberg (2009)
6. Boneh, D., Boyen, X., Goh, E.: Hierarchical identity based encryption with constant size ciphertext. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 440–456. Springer, Heidelberg (2005)
7. Shi, E., Waters, B.: Delegating capabilities in predicate encryption systems. In: Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.) ICALP 2008, Part II. LNCS, vol. 5126, pp. 560–578. Springer, Heidelberg (2008)
8. Boneh, D., Boyen, X.: Efficient selective-ID secure identity-based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 223–238. Springer, Heidelberg (2004)
9. Shamir, A.: Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 47–53. Springer, Heidelberg (1985)
10. Boneh, D., Franklin, M.: Identity based encryption from the weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001)
11. Cocks, C.: An identity based encryption scheme based on quadratic residues. In: Honary, B. (ed.) Cryptography and Coding 2001. LNCS, vol. 2260, p. 360. Springer, Heidelberg (2001)
12. Canetti, R., Halevi, S., Katz, J.: A forward-secure public-key encryption scheme. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 255–271. Springer, Heidelberg (2003)
13. Boneh, D., Boyen, X.: Secure identity based encryption without random oracles. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 443–459. Springer, Heidelberg (2004)
14. Waters, B.: Efficient identity-based encryption without random oracles. In: Cramer, R. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 114–127. Springer, Heidelberg (2005)
15. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Proceedings of the 40th annual ACM Symposium on Theory of Computing, pp. 197–206. ACM, New York (2008)
16. Naor, D., Naor, M., Lotspiech, J.: Revocation and tracing schemes for stateless receivers. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 41–62. Springer, Heidelberg (2001)
17. Dodis, Y., Fazio, N.: Public key broadcast encryption for stateless receivers. In: Feigenbaum, J. (ed.) DRM 2002. LNCS, vol. 2696, pp. 61–80. Springer, Heidelberg (2003)
18. Boneh, D., Goh, E., Nissim, K.: Evaluating 2-DNF formulas on ciphertexts. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 325–341. Springer, Heidelberg (2005)
19. Katz, J., Sahai, A., Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 146–162. Springer, Heidelberg (2008)

A Generic Security of Our Complexity Assumptions

We now prove our three complexity assumptions hold in the generic group model, as long as it is hard to find a nontrivial factor of the group order, N. We adopt the notation of [19] to express our assumptions. We fix generators $g_{p_1}, g_{p_2}, g_{p_3}$ of the subgroups $G_{p_1}, G_{p_2}, G_{p_3}$ respectively. Every element of G can then be expressed as $g_{p_1}^{a_1} g_{p_2}^{a_2} g_{p_3}^{a_3}$ for some values of a_1, a_2, a_3. We denote an element of G by (a_1, a_2, a_3). The element $e(g_{p_1}, g_{p_1})^{a_1} e(g_{p_2}, g_{p_2})^{a_2} e(g_{p_3}, g_{p_3})^{a_3}$ in G_T will be denoted by $[a_1, a_2, a_3]$. We use capital letters to denote random variables, and we reuse random variables to denote relationships between elements. For example, $X = (X_1, Y_1, Z_1)$ is a random element of G, and $Y = (X_1, Y_2, Z_2)$ is another random element that shares the same component in the G_{p_1} subgroup.

Given random variables $X, \{A_i\}$ expressed in this form, we say that X is *dependent* on $\{A_i\}$ if there exists values $\lambda_i \in \mathbb{Z}_n$ such that $X = \sum_i \lambda_i A_i$ as formal random variables. Otherwise, we say that X is *independent* of $\{A_i\}$. We note the following two theorems from [19]:

Theorem 2. *(Theorem A.1 of [19]) Let $N = \prod_{i=1}^{m} p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{A_i\}$ be random variables over G, and let $\{B_i\}, T_0, T_1$ be random variables over G_T, where all random variables have degree at most t. Consider the following experiment in the generic group model:*

An algorithm is given $N, \{A_i\}$, and $\{B_i\}$. A random bit b is chosen, and the adversary is given T_b. The algorithm outputs a bits b', and succeeds if $b' = b$. The algorithm's advantage is the absolute value of the difference between its success probability and $\frac{1}{2}$.

Say each of T_0 and T_1 is independent of $\{B_i\} \cup \{e(A_i, A_j)\}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ in the above experiment, $\mathcal{A}$ can be used to find a nontrivial factor of N (in time polynomial in λ and the running time of $\mathcal{A}$) with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.

Theorem 3. *(Theorem A.2 of [19]) Let $N = \prod_{i=1}^{m} p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{A_i\}, T_0, T_1$ be random variables over G, and let $\{B_i\}$ be random variables over G_T, where all random variables have degree at most t. Consider the same experiment as in the theorem above.*

Let $S := \{i | e(T_0, A_i) \neq e(T_1, A_i)\}$ (where inequality refers to inequality as formal polynomials). Say each of T_0 and T_1 is independent of $\{A_i\}$, and furthermore that for all $k \in S$ it holds that $e(T_0, A_k)$ is independent of $\{B_i\} \cup \{e(A_i, A_j)\} \cup \{e(T_0, A_i)\}_{i \neq k}$, and $e(T_1, A_k)$ is independent of $\{B_i\} \cup \{e(A_i, A_j)\} \cup \{e(T_1, A_i)\}_{i \neq k}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ, the algorithm can be used to find a nontrivial factor of N (in time polynomial in λ and the running time of $\mathcal{A}$) with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.

We apply these theorems to prove the security of our assumptions in the generic group model.

Assumption 1. We apply Theorem 3. We can express this assumption as:

$$A_1 = (1,0,0), A_2 = (0,0,1),$$

$$T_0 = (X_1, X_2, 0), T_1 = (X_1, 0, 0).$$

We note that $S = \emptyset$ in this case. It is clear that T_0 and T_1 are both independent of $\{A_1, A_2\}$ because X_1 does not appear in A_1 or A_2. Thus, Assumption 1 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 2. We apply Theorem 3. We can express this assumption as:

$$A_1 = (1,0,0), A_2 = (X_1,1,0), A_3 = (Y_1,0,0), A_4 = (0,X_2,1),$$

$$T_0 = (Z_1, Z_2, Z_3), T_1 = (Z_1, 0, Z_3).$$

We note that $S = \{2,4\}$ in this case. It is clear that T_0 and T_1 are both independent of $\{A_i\}$ since Z_1 does not appear in the A_i's, for example. We see that $e(T_0, A_2)$ is independent of $\{e(A_i,A_j)\} \cup \{e(T_0,A_i)\}_{i\neq 2}$ because it is impossible to obtain X_1Z_1 in the first coordinate of a combination of elements of $\{e(A_i,A_j)\} \cup \{e(T_0,A_i)\}_{i\neq 2}$. This also allows us to conclude that $e(T_1,A_2)$ is independent of $\{e(A_i,A_j)\} \cup \{e(T_1,A_i)\}_{i\neq 2}$. We similarly note that $e(T_0,A_4)$ is independent of $\{e(A_i,A_j)\} \cup \{e(T_0,A_i)\}_{i\neq 4}$ and $e(T_1,A_4)$ is independent of $\{e(A_i,A_j)\} \cup \{e(T_1,A_i)\}_{i\neq 4}$ because we cannot obtain Z_3 in the third coordinate. Thus, Assumption 2 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 3. We apply Theorem 2. We can express this assumption as:

$$A_1 = (1,0,0), A_2 = (B,1,0), A_3 = (0,0,1), A_4 = (S,X_2,0), A_5 = (0,Y_2,0),$$

$$T_0 = [BS,0,0], T_2 = [Z_1,Z_2,Z_3].$$

T_1 is independent of $\{e(A_i,A_j)\}$ because Z_1, Z_2, Z_3 do not appear in $\{A_i\}$. T_0 is independent of $\{e(A_i,A_j)\}$ because the only way to obtain BS in the first coordinate is to take $e(A_2,A_4)$, but then we are left with an X_2 in the second coordinate that cannot be canceled. Thus, Assumption 3 is generically secure, assuming it is hard to find a nontrivial factor of N.

B HIBE Security Proof

Lemma 5. *For any algorithm $\mathcal{A}$, $Game_{Real}Adv_{\mathcal{A}} = Game_{Real'}Adv_{\mathcal{A}}$.*

Proof. We note that keys are identically distributed whether they are produced by the key delegation algorithm from a previous key or from a fresh call to the key generation algorithm. Thus, in the attacker's view, there is no difference between these games.

Lemma 6. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{Real'}Adv_{\mathcal{A}} - Game_{Restricted}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage $\geq \frac{\epsilon}{2}$ in breaking either Assumption 1 or Assumption 2.*

Proof. This proof is identical to the proof of Lemma 5.

Lemma 7. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{Restricted}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ first receives g, X_3, T. It simulates Game$_{Real}$ or Game$_0$ with A. It sets the public parameters as follows. It chooses random exponents $\alpha, a_1, \ldots, a_\ell, b \in \mathbb{Z}_N$ and sets $g = g, u_i = g^{a_i}$ for i from 1 to ℓ and $h = g^b$. It sends these public parameters $\{N, g, u_1, \ldots, u_\ell, h, e(g,g)^\alpha\}$ to $\mathcal{A}$. Each time $\mathcal{B}$ is asked to provide a key for an identity $(ID_1, \ldots, ID_j)$, it chooses random exponents r, t, w, $v_{j_1}, \ldots, v_\ell \in \mathbb{Z}_N$ and sets:

$$K_1 = g^r X_3^t, K_2 = g^\alpha (u_1^{ID_1} \cdot u_j^{ID_j} h)^r X_3^w, E_{j+1} = u_{j+1}^r X_3^{v_{j+1}}, \ldots, E_\ell = u_\ell^r X_3^{v_\ell}.$$

$\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, $(ID_1^*, \ldots, ID_j^*)$. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly. The ciphertext is formed as follows:

$$C_0 = M_\beta e(T,g)^\alpha, C_1 = T^{a_1 ID_1^* + \cdots a_j ID_j^* + b}, C_2 = T.$$

(This implicitly sets g^s equal to the G_{p_1} part of T.) If $T = \in G_{p_1 p_2}$, then this is a semi-functional ciphertext with $z_c = a_1 ID_1^* + \cdots + a_j ID_j^* + b$. If $T \in G_{p_1}$, this is a normal ciphertext. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

Lemma 8. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{k-1}Adv_{\mathcal{A}} - Game_kAdv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ first receives $g, X_1X_2, X_3, Y_2Y_3, T$. $\mathcal{B}$ picks random exponents $a_1, \ldots, a_\ell$, $b \in \mathbb{Z}_N$ and sets the public parameters as: $g = g, u_i = g^{a_i}, h = g^b, e(g,g)^\alpha$. It sends these to $\mathcal{A}$. When $\mathcal{A}$ requests the i^{th} key for $(ID_1, \ldots, ID_j)$ when $i < k$, $\mathcal{B}$ creates a semi-functional key. It does this by choosing random exponents r, z, t, $z_{j+1}, \ldots, z_\ell \in \mathbb{Z}_N$ and setting:

$$K_1 = g^r (Y_2Y_3)^t, K_2 = g^\alpha (u_1^{ID_1} \cdots u_j^{ID_j} h)^r (Y_2Y_3)^z,$$

$$E_{j+1} = u_{j+1}^r (Y_2Y_3)^{z_{j+1}}, \ldots, E_\ell = u_\ell^r (Y_2Y_3)^{z_\ell}.$$

This is a properly distributed semi-functional key with $g_2^\gamma = Y_2^t$.

For $i > k$, $\mathcal{B}$ generates normal keys by calling the usual key generation algorithm.

To create the k^{th} requested key for $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ lets $z_k = a_1 ID_1 + \cdots a_j ID_j + b$, chooses random exponents $w_k, w_{j+1}, \ldots, w_\ell \in \mathbb{Z}_N$, and sets:

$$K_1 = T, K_2 = g^\alpha T^{z_k} X_3^{w_k}, E_{j+1} = T^{a_{j+1}} X_3^{w_{j+1}}, \ldots, E_\ell = T^{a_\ell} X_3^{w_\ell}.$$

If $T \in G_{p_1p_3}$, this is a normal key with g^r equal to the G_{p_1} part of T. If $T \in G$, this is a semi-functional key.

At some point, $\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, $(ID_1^*, \ldots, ID_j^*)$. $\mathcal{B}$ sets $\beta \in \{0,1\}$ randomly. The challenge ciphertext is formed as:

$$C_0 = M_\beta e(X_1X_2, g)^\alpha, C_1 = (X_1X_2)^{a_1ID_1^* + \cdots + a_jID_j^* + b}, C_2 = X_1X_2.$$

We note that this sets $g^s = X_1$ and $z_c = a_1ID_1^* + \cdots a_jID_j^* + b$. Since the k^{th} key is not a prefix of the challenge key modulo p_2, z_k and z_c will seem randomly distributed to $\mathcal{A}$. Though it is hidden from $\mathcal{A}$, this relationship between z_c and z_k is crucial: if $\mathcal{B}$ attempts to test itself whether key k is semi-functional by creating a semi-functional ciphertext for this identity and trying to decrypt, then decryption will work whether key k is semi-functional or not, because $z_c = z_k$. In other words, the simulator can only create a nominally semi-functional key k.

If $T \in G_{p_1p_3}$, then $\mathcal{B}$ has properly simulated Game$_{k-1}$. If $T \in G$, then $\mathcal{B}$ has properly simulated Game$_k$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

Lemma 9. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_qAdv_{\mathcal{A}} - Game_{Final}Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ first receives $g, g^\alpha X_2, X_3, g^sY_2, Z_2, T$. $\mathcal{B}$ chooses random exponents $a_1, \ldots, a_\ell, b \in \mathbb{Z}_N$ and sets the public parameters as $g = g, u_1 = g^{a_1}, \ldots, u_\ell = g^{a_\ell}, h = g^b, e(g,g)^\alpha = e(g^\alpha X_2, g)$. It sends these to $\mathcal{A}$. When $\mathcal{A}$ requests a key for identity $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ generates a semi-functional key. It does this by choosing random exponents c, r, t, w, z, z_{j+1}, $\ldots, z_\ell$, $w_{j+1}, \ldots, w_\ell \in \mathbb{Z}_N$ and setting:

$$K_1 = g^r Z_2^z X_3^t, K_2 = g^\alpha X_2 Z_2^c (u_1^{ID_1} \cdots u_j^{ID_j} h)^r X_3^w,$$

$$E_{j+1} = u_{j+1}^r Z_2^{z_{j+1}} X_3^{w_{j+1}}, \ldots, E_\ell = u_\ell^r Z_2^{z_\ell} X_3^{w_\ell}.$$

$\mathcal{A}$ sends $\mathcal{B}$ two messages, M_0 and M_1, and a challenge identity, $(ID_1^*, \ldots, ID_j^*)$. $\mathcal{B}$ sets $\beta \in \{0,1\}$ randomly. It forms the challenge ciphertext as:

$$C_0 = M_\beta T, C_1 = (g^sY_2)^{a_1ID_1^* + \cdots a_jID_j^* + b}, C_2 = g^sY_2.$$

This sets $z_c = a_1ID_1^* + \cdots + a_jID_j^* + b$. We note that the value of z_c only matters modulo p_2, whereas $u_1 = g^{a_1}, \ldots, u_\ell = g^{a_\ell}$, and $h = g^b$ are elements of G_{p_1}, so when $a_1, \ldots, a_\ell$ and b are chosen randomly modulo N, there is no correlation between the values of $a_1, \ldots, a_\ell, b$ modulo p_1 and the value $z_c = a_1ID_1^* + \cdots + a_jID_j^* + b$ modulo p_2.

If $T = e(g,g)^{\alpha s}$, then this is a properly distributed semi-functional ciphertext with message M_β. If T is a random element of G_T, then this is a semi-functional ciphertext with a random message. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to distinguish between these possibilities for T.

We have now proven the following theorem:

Theorem 4. *If Assumptions 1, 2, and 3 hold, then our HIBE system is secure.*

Proof. If Assumptions 1, 2, and 3 hold, then we have shown by the previous lemmas that the real security game is indistinguishable from Game_{Final}, in which the value of β is information-theoretically hidden from the attacker. Hence the attacker can attain no advantage in breaking the HIBE system.

C IBE in Prime Order Groups

Our construction essentially replaces each single group element in our composite order construction with a 3-tuple of group elements. This 3-tuple is inspired by a simplification of the Waters dual encryption IBE system [5]. We prove security under 3 new static assumptions. We leave it as an open problem to obtain security from the decisional Linear and d-BDH assumptions. One approach would be to use more of the Waters system (with less simplification).

C.1 Construction

For our construction, we employ prime order groups G_1, G_2, G_T of order p such that there is an efficient bilinear map $e : G_1 \times G_2 \rightarrow G_T$ but no efficient isomorphism between G_1 and G_2. We use subscripts to clarify which elements are in G_1 and which are in G_2, for example, $g_1 \in G_1$.

Setup. Our setup algorithm chooses groups G_1, G_2, G_T of order p as above. It chooses $g_1, u_1, h_1 \in G_1, g_2 \in G_2$ randomly. It sets u_2 and h_2 so that the discrete log of u_2, h_2 base g_2 is equal to the discrete log of u_1, h_1 base g_1 respectively. It chooses $a, \alpha \in Z_p$ randomly. It chooses $v_2, v_2', f_2 \in G_2$ randomly and sets $\tau \in Z_p$ to satisfy $f_2^{\tau} = v_2 (v_2')^a$. It publishes the public parameters as:

$$\{g_1, u_1, h_1, g_1^a, u_1^a, h_1^a, g_1^{\tau}, u_1^{\tau}, h_1^{\tau}, e(g_1, g_2)^{\alpha}\}.$$

The master secret key is $g_2, \alpha, v_2, v_2', u_2, h_2, f_2$.

Encrypt(M, ID). The encryption algorithm randomly chooses $s \in Z_p$ and creates the ciphertext as:

$$C_0 = Me(g_1, g_2)^{\alpha s}, C_{1,1} = (u_1^{ID} h_1)^s, C_{1,2} = (u_1^{ID} h)^{as}, C_{1,3} = (u_1^{ID} h_1)^{-s\tau},$$

$$C_{2,1} = g_1^s, C_{2,2} = g_1^{as}, C_{2,3} = g_1^{\tau s}.$$

KeyGen(ID, MSK). The key generation algorithm chooses random values $y, c_1, c_2 \in \mathbb{Z}_p$. It creates the key as:

$$K_{1,1} = g_2^y v_2^{c_1}, K_{1,2} = (v_2')^{c_1}, K_{1,3} = f_2^{c_1},$$

$$K_{2,1} = g_2^{\alpha} (u_2^{ID} h_2)^y v_2^{c_2}, K_{2,2} = (v_2')^{c_2}, K_{2,3} = f_2^{c_2}.$$

Decryption. If the ID's of the ciphertext and key are equal, the decryption algorithm computes the blinding factor as:

$$\frac{e(C_{2,1},K_{2,1})e(C_{2,2},K_{2,2})e(C_{2,3},K_{2,3})}{e(C_{1,1},K_{1,1})e(C_{1,2},K_{1,2})e(C_{1,3}K_{1,3})}.$$

C.2 Complexity Assumptions

We state the assumptions we will rely on in our security proof. These are non-standard assumptions, but we emphasize that they are static.

Assumption 1. Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $a, b, s \in \mathbb{Z}_p$ be chosen randomly. Given

$$\{f_1, f_1^{bs}, f_1^s, f_1^a, f_1^{ab^2}, f_1^b, f_1^{b^2}, f_1^{as}, f_1^{b^2 s}, f_1^{b^3}, f_1^{b^3 s}, T \in G_1, f_2, f_2^b \in G_2\},$$

it should be hard to distinguish $T = f_1^{asb^2}$ from random.

Assumption 2. Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $d, b, c, x \in \mathbb{Z}_p$ be chosen randomly. Given

$$\{f_1, f_1^d, f_1^{d^2}, f_1^{bx}, f_1^{dbx}, f_1^{d^2 x} \in G_1, f_2, f_2^d, f_2^b, f_2^c \in G_2, T \in G_2\},$$

it should be hard to distinguish $T = f_2^{bc}$ from random.

Assumption 3. Let $f_1 \in G_1$ and $f_2 \in G_2$ be chosen randomly. Let $d, b, c \in \mathbb{Z}_p$ be chosen randomly. Given

$$\{f_1, f_1^a, f_1^b, f_1^c \in G_1, f_2, f_2^a, f_2^b, f_2^c \in G_2, T \in G_T\},$$

it should be hard to distinguish $T = e(f_1, f_2)^{abc}$ from random.

C.3 Security

We first define semi-functional keys and ciphertexts.

Semi-functional Ciphertext. We let f_1, v_1' denote elements of G_1 such that the discrete log of v_1' base f_1 is the same as the discrete log of v_2' base f_2. We let t, z_c denote random exponents in $\mathbb{Z}_p$. A semi-functional ciphertext is created as follows: first, a normal ciphertext $C_0', C_{1,1}', C_{1,2}', C_{1,3}', C_{2,1}', C_{2,2}', C_{2,3}'$ is created. Then, C_0 is set to be C_0', $C_{1,1} = C_{1,1}'$, $C_{1,2} = C_{1,2}' f_1^{tz_c}, C_{1,3} = C_{1,3}'(v_1')^{-tz_c}$, $C_{2,1} = C_{2,1}'$, $C_{2,2} = C_{2,2}' f_1^t$, $C_{2,3} = C_{2,3}'(v_1')^{-t}$.

Semi-functional Key. A semi-functional key is created as follows: a normal key $K_{1,1}'$, $K_{1,2}'$, $K_{1,3}'$, $K_{2,1}', K_{2,2}', K_{2,3}'$ is generated. Random exponents $w, z_k \in \mathbb{Z}_p$ are chosen. Then we set: $K_{1,1} = K_{1,1}' f_2^{-aw}, K_{1,2} = K_{1,2}' f_2^w, K_{1,3} = K_{1,3}'$, $K_{2,1} = K_{2,1}' f_2^{-awz_k}, K_{2,2} = K_{2,2}' f_2^{wz_k}, K_{2,3} = K_{2,3}'$.

We note that when a semi-functional key is paired with a normal ciphertext or a normal key is paired with a semi-functional ciphertext, decryption still works. When a semi-functional key is paired with a semi-functional ciphertext, the blinding factor is obscured by an additional term: $e(f_1, f_2)^{tw(z_k - z_c)}$. (When $z_k = z_c$, decryption will still work.)

We will prove security through a hybrid argument over a sequence of games. Game$_{Real}$ is the real security game. Game$_0$ is like the real security game, except with a semi-functional ciphertext. *Game$_k$* for k from 1 to q (where q is the number of queries by the attacker) is like Game$_0$, except that the first k requested keys are semi-functional and the rest are normal. In Game$_{Final}$, the semi-functional encryption is of a random message instead of one of the requested messages. We will rely on Assumptions 1, 2, 3 as defined in the subsection above. We prove security through the following 3 lemmas.

Lemma 10. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{Real}Adv_{\mathcal{A}} - Game_0 Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ is given

$$\{f_1, f_1^{bs}, f_1^s, f_1^a, f_1^{ab^2}, f_1^b, f_1^{b^2}, f_1^{as}, f_1^{b^2 s}, f_1^{b^3}, f_1^{b^3 s}, T \in G_1, f_2, f_2^b \in G_2\}.$$

It chooses random exponents $\alpha, A, B, y_g, y_u, y_h, y_v' \in \mathbb{Z}_p$ and sets the parameters as:

$$g_1 = f_1^{b^2} f_1^{y_g}, u_1 = (f_1^{b^2})^A f_1^{y_u}, h_1 = (f_1^{b^2})^B f_1^{y_h}, g_1^a, u_1^a, h_1^a,$$

$$f_2 = f_2, v_2 = f_2^b, v_2' = f_2^{y_v'}, \tau = b + a y_v'.$$

Here, a is from the assumption and g_1^a, u_1^a, h_1^a can be computed from f_1^a and $f_1^{ab^2}$. We note that $\mathcal{B}$ can also compute g_1^τ, u_1^τ, and h_1^τ using $f_1^{b^3}$, $f_1^{b_2 a}$, f_1^a, and f_1^b. It can also compute $e(g_1, g_2)^\alpha$ using $f_1^{b^2}, f_2^b$, and $f_1^{b^3}$.

To construct a normal key for ID, $\mathcal{B}$ chooses random exponents $c_1', c_2', y \in \mathbb{Z}_p$ and sets $f_2^{c_1} = f_2^{c_1'}(f_2^b)^{-y}$ and $f_2^{c_2} = f_2^{c_2'}(f_2^b)^{-yAID+B-\alpha}$. Then the key can be formed as:

$$K_{1,1} = f_2^{y_g y}(f_2^b)^{c_1'}, K_{1,2} = (f_2^{c_1})^{y_v'}, K_{1,3} = f_2^{c_1},$$

$$K_{2,1} = f_2^{\alpha y_g}(f_2^b)^{c_2'} f_2^{y(y_u ID + y_h)}, K_{2,2} = (f_2^{c_2})^{y_v'}, K_{2,3} = f_2^{c_2}.$$

To construct the challenge ciphertext for M_β and ID^*, $\mathcal{B}$ sets $s = s$ from the assumption. Then $C_{1,1}$ and $C_{2,1}$ can be computed from f_1^s and $f_1^{b^2 s}$. Next,

$$C_{1,2} = T^{AID+B}(f_1^{as})^{y_I D + y_h}, C_{2,2} = T(f_1^{as})^{y_g}.$$

We can create $C_{2,3}$ as:

$$C_{2,3} = f_1^{b^3 s}(f_1^{bs})^{y_g} T^{y_v'}(f_1^{as})^{y_g y_v'}.$$

$C_{1,3}$ can similarly be constructed using $T^{y_v'(AID+B)}$. We note that A, B are information-theoretically hidden from the attacker.

Lemma 11. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_{k-1}Adv_{\mathcal{A}} - Game_kAdv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ is given $\{f_1, f_1^d, f_1^{d^2}, f_1^{bx}, f_1^{dbx}, f_1^{d^2x} \in G_1, f_2, f_2^d, f_2^b, f_2^c \in G_2, T \in G_2\}$. It chooses random exponents $\alpha, a, A, B, y_u, y_h, y_v \in \mathbb{Z}_p$. It sets the parameters as follows:

$$g_1 = f_1^d, u_1 = (f_1^d)^A f_1^{y_u}, h_1 = (f_1^d)^B f_1^{y_h}, g_1^a, u_1^a, h_1^a, g_2 = f_2^d, u_2 = (f_2^d)^A f_2^{y_u},$$

$$h_2 = (f_2^d)^B f_2^{y_h}, v_2' = f_2^b, v_2 = f_2^d f_2^{-ba} f_2^{y_v}, f_2 = f_2.$$

This sets $\tau = d - ba + y_v + ab = d + y_v$, so the simulator $\mathcal{B}$ can also compute $g_1^\tau, u_1^\tau, h_1^\tau$ and send all of the public parameters to $\mathcal{A}$.

To make normal keys for key queries $< k$, $\mathcal{B}$ can choose $y, c_1, c_2 \in \mathbb{Z}_p$ randomly and generate the keys from the MSK. To make semi-functional keys for key queries $> k$, $\mathcal{B}$ can choose $y, c_1, c_2, w, z_k \in \mathbb{Z}_p$ randomly to generate the semi-functional key.

To make the challenge key k for ID, $\mathcal{B}$ chooses $y', c_2' \in \mathbb{Z}_p$ randomly and implicitly sets $y = -c + y'$, $c_1 = c$, $c_2 = c(AID + B) + c_2'$. The key can then be formed as:

$$K_{1,1} = (f_2^d)^{y'} T^{-a} (f_2^c)^{y_v}, K_{1,2} = T, K_{1,3} = f_2^c,$$

$$K_{2,1} = g_2^\alpha T^{-a(AID+B)} (f_2^b)^{-ac_2'} (f_2^d)^{y'(AID+B)+c_2'}$$
$$(f_2^c)^{y_u ID + y_h + y_v(AID+B)} f_2^{y'(y_u ID + y_h) + c_2' y_v},$$

$$K_{2,2} = T^{AID+B} (f_2^b)^{c_2'}, K_{2,3} = (f_2^c)^{AID+B} f_2^{c_2'}.$$

We note that this sets $z_k = AID + B$.

At some point, $\mathcal{A}$ sends two messages, M_0, M_1, to $\mathcal{B}$ along with a challenge identity ID^*. $\mathcal{B}$ chooses $\beta \in \{0, 1\}$ randomly and generates a semi-functional ciphertext for M_β and ID^* as follows. $\mathcal{B}$ chooses a random exponent $s' \in \mathbb{Z}_p$ and implicitly sets $s = bx + s'$, $t = -d^2x$. The ciphertext is formed as follows:

$$C_0 = M_\beta e(f_1^{dbx}, f_2^d) e(g_1, g_2)^{\alpha s'}, C_{1,1} = (f_1^{dbx})^{AID^*+B} (f_1^{bx})^{y_u ID^* + B} (u_1^{ID^*} h_1)^{s'},$$

$$C_{1,2} = (f_1^{dbx})^{a(AID^*+B)} (f_1^{d^2x})^{-(AID^*+B)} (u_1^{ID^*} h_1)^{as'},$$

$$C_{1,3} = (f_1^{dbx})^{-y_v(AID^*+B)} (f_1^{d^2})^{-s'(AID^*+B)} (f_1^d)^{-y_v s'(AID^*+B)},$$

$$C_{2,1} = f_1^{dbx} (f_1^d)^{s'}, C_{2,2} = (f_1^{dbx})^a (f_1^d)^{s'a} (f_1^{d^2x})^{-1},$$

$$C_{2,3} = (f_1^{dbx})^{-y_v} (f_1^{d^2})^{-s'} (f_1^d)^{-y_v s'}.$$

We note that $z_c = AID^* + B$. Since A and B are information-theoretically hidden from the attacker, this will seem properly distributed to the attacker. If $T = f_2^{bc}$, then $\mathcal{B}$ has properly simulated Game$_{k-1}$, and if T is random, then $\mathcal{B}$ has properly simulated Game$_k$.

Lemma 12. *Suppose there exists an algorithm $\mathcal{A}$ such that $Game_q Adv_{\mathcal{A}} - Game_{Final} Adv_{\mathcal{A}} = \epsilon$. Then we can build an algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ is given

$$\{f_1, f_1^d, f_1^{d^2}, f_1^{bx}, f_1^{dbx}, f_1^{d^2x} \in G_1, f_2, f_2^d, f_2^b, f_2^c \in G_2, T \in G_2\}.$$

It will implicitly set $\alpha = ab$, $s = c$, and $a = a$. $\mathcal{B}$ chooses random exponents $y_g, y_u, y_v, y_v, y'_v \in \mathbb{Z}_p$. It sets the parameters as:

$$g_1 = f_1^{y_g}, u_1 = f_1^{y_u}, h_1 = f_1^{y_h}, v_2 = f_2^{y_v}, v'_2 = f_2^{y'_v}, g_2 = f_2^{y_g}$$

and sets $\tau = y_v + a y'_v$. From this, it can calculate the rest of the public parameters as:

$$g_1^a = (f_1^a)^{y_g}, u_1^a = (f_1^a)^{y_u}, h_1^a = (f_1^a)^{y_h}, g_1^\tau = (g_1^a)^{y'_v} g_1^{y_v},$$

$$u_1^\tau = (u_1^a)^{y'_v} u_1^{y_v}, h_1^\tau = (h_1^a)^{y'_v} h_1^{y_v}, e(g_1, g_2)^\alpha = e(f_1^a, f_2^b)^{y_g^2}.$$

To make semi-functional keys, $\mathcal{B}$ must cancel the term g_2^α in $K_{2,1}$ since this is unknown. To do this, the simulator randomly chooses $w, c_1, c_2, y, \gamma \in \mathbb{Z}_p$ and implicitly sets $wz_k = b + \gamma$.

To make the challenge ciphertext for M_β and ID^*, $\mathcal{B}$ sets $s = c$ and chooses random values $\delta, \delta' \in \mathbb{Z}_p$. It implicitly sets $ca + t = \delta$ and $ca(y_u ID^* + y_h) + tz_c = \delta'$ and $acy_g + t = \delta$.

$$C_0 = M_\beta T, C_{1,1} = (f_1^c)^{y_u ID^* + y_h}, C_{1,2} = f_1^{\delta'}, C_{1,3} = (f_1^c)^{-y_v(y_u ID^* + y_h)} f_1^{-y'_v \delta'},$$

$$C_{2,1} = (f_1^c)^{y_g}, C_{2,2} = f_1^{\delta}, C_{2,3} = (f_1^c)^{-y_v y_g} (f_1^{\delta})^{-y'_v}.$$