Unbounded HIBE and Attribute-Based Encryption

Allison Lewko[1] * and Brent Waters[2] **

[1] University of Texas Austin
alewko@cs.utexas.edu
[2] University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. In this work, we present HIBE and ABE schemes which are "unbounded" in the sense that the public parameters do not impose additional limitations on the functionality of the systems. In all previous constructions of HIBE in the standard model, a maximum hierarchy depth had to be fixed at setup. In all previous constructions of ABE in the standard model, either a small universe size or a bound on the size of attribute sets had to be fixed at setup. Our constructions avoid these limitations. We use a nested dual system encryption argument to prove full security for our HIBE scheme and selective security for our ABE scheme, both in the standard model and relying on static assumptions. Our ABE scheme supports LSSS matrices as access structures and also provides delegation capabilities to users.

1 Introduction

Hierarchical Identity-Based Encryption (HIBE) systems [29, 26] and Attribute-Based Encryption (ABE) systems [40] offer users more levels of flexibility in sharing and managing sensitive data than are provided by Identity-Based and Public Key Encryption systems. In a hierarchical identity-based encryption scheme, user identities are arranged in an organizational hierarchy. Anyone can encrypt a message to any identity in the system using the public parameters. An identity at level k in the hierarchy can use its secret key to delegate secret keys to its subordinates, but cannot decrypt any messages which are intended for recipients other than itself and its subordinates. In a Key-Policy Attribute-Based Encryption (KP-ABE) system [28], users have secret keys which are associated with access policies over a universe of attributes and ciphertexts are associated with sets of attributes. A user can decrypt a message encrypted to a set of attributes S only if S satisfies the access policy of the user's key.

* Supported by National Defense Science and Engineering Graduate Fellowship.
** Supported by NSF CNS-0915361, and CNS-0952692, the MURI program under AFOSR Grant No: FA9550-08-1-0352. Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), a Google Faculty Research award, and the Alfred P. Sloan Foundation.

Both HIBE and ABE systems are designed to accommodate certain changes in the needs of users over time, but current constructions have some inherent limitations. For instance, new users can enter an HIBE system and collect secret keys without requiring any change to the public parameters or the keys of users already present. However, for all previous constructions in the standard model, the identities of new users must fit within the hierarchy depth specified by the public parameters. More precisely, the size of the public parameters grows linearly with the maximum depth of the hierarchy, and it is impossible to add new levels to the hierarchy once the public parameters are fixed. In the ABE setting, the particular access policies and attribute sets employed by users may change over time, but current constructions in the standard model do not allow complete versatility in the choice of attributes and policies once the public parameters have been set. In "small universe" constructions (e.g. [28, 31]), a polynomially sized universe of attributes must be fixed at setup, and the size of the public parameters grows linearly with the size of the chosen attribute universe. In "large universe" constructions (e.g. [28]), the attribute universe is exponentially large, but the size of a set S used for encryption is bounded by a parameter n which is fixed at setup. The size of the public parameters grows linearly with n.

This places an undesirable burden on someone wishing to deploy an HIBE or ABE system to be used in practice. If the setup parameters are chosen to be too small, the system will not achieve the desired longevity and will need to be completely re-initialized when users exhaust its overly restrictive structure. If the setup parameters are chosen to be too large, then the public parameters of the system will be needlessly large and this will cause unnecessary inefficiency.

Removing these restrictions from previous approaches appears to be quite challenging. For example, many standard model HIBE constructions employ structures similar to the Boneh-Boyen HIBE in [9] (e.g. [11, 10, 45, 34] fall roughly into this framework). At a high level, these systems all rely on hash functions H which map identity vectors to group elements in a particular way. More specifically, we suppose that a user at level j in the hierarchy is associated with an identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$. The hash function H uses d fixed group elements $u_1, \ldots, u_d$ in a bilinear group G of order p (for example). Upon receiving an identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ as input, H somehow chooses k vectors $\boldsymbol{v}^1 = (v_1^1, \ldots, v_d^1), \ldots, \boldsymbol{v}^k = (v_1^k, \ldots, v_d^k) \in \mathbb{Z}_p^d$, where k is a function of j and the maximum depth of the hierarchy. In particular, k will be strictly less than d. It then outputs group elements of the form

$$\left(u_1^{v_1^1} \cdot u_2^{v_2^1} \cdots u_d^{v_d^1}\right), \ldots, \left(u_1^{v_1^k} \cdot u_2^{v_2^k} \cdots u_d^{v_d^k}\right).$$

In forming the secret keys or ciphertexts, these group elements are typically each raised to the same random exponent in $\mathbb{Z}_p$.

If we try to apply this approach without bounding the maximum depth of the hierarchy, then for some identity vectors, we will need to produce $\geq d$ samples of the form above, and each will be raised to the same exponent $s \in \mathbb{Z}_p$. This causes insecurity - since our vectors $\boldsymbol{v}^1, \ldots, \boldsymbol{v}^k$ reside in a d-dimensional space,

most collections of d of them will be linearly independent, and will span $\mathbb{Z}_p^d$. This will allow an attacker to create a new sample

$$\left(u_1^{v_1^*} \cdot u_2^{v_2^*} \cdots u_d^{v_d^*}\right)^s$$

for any vector $(v_1^*, \ldots, v_d^*)$ that it wants, by taking its received samples, raising them to appropriate powers, and multiplying the results. For this reason, achieving unbounded HIBE systems by relying on these sorts of hash functions seems unlikely.

Our Contribution Using new techniques, we obtain "unbounded" HIBE and ABE schemes. Our HIBE scheme can accommodate arbitrary hierarchy depths from public parameters which consist of only a constant number of group elements. This eliminates the need to decide maximum hierarchy depth at setup and reduces the size of the public parameters. We prove our scheme fully secure in the standard model, relying on static, generically secure assumptions in composite order bilinear groups. Our ABE scheme has a large attribute universe and imposes no bound on the size of attribute sets used for encryption. It also has public parameters which are a constant number of group elements. It supports LSSS matrices as access structures, and additionally provides delegation capabilities to users. Our ABE scheme is proven selectively secure[3] from the same static, generically secure assumptions in composite order bilinear groups.

Our Techniques We overcome the limitations of previous constructions by employing a secret-sharing technique and introducing fresh "local" randomness at each level of the keys and ciphertexts. Thus, instead of needing to create too many samples from a bounded dimensional vector space with the same randomness, we will be creating many samples which each have *new randomness*. This avoids the insecurity of the previous approach described above.

To create a secret key for a user in our HIBE and ABE systems, we first split the master secret into shares that will be associated with the components of the user's identity vector or the rows of its access matrix. Each share is then blinded by randomness which is freshly chosen for each share and links the share to its corresponding identity or attribute.

The main obstacle to proving the security of our schemes is the low amount of entropy provided by the short public parameters. This poses a challenge for both partitioning proof techniques and the more recently introduced technique of dual system encryption [45]. To successfully execute a partitioning proof, we would need to program the public parameters to allow cancelations when the simulator attempts to make a certain key or keys. However, the small number of degrees of freedom available in the public parameters make it difficult to program in keys of arbitrary depth. To use a dual system encryption proof, we must execute an information-theoretic argument in a low entropy context - this is a challenge, but

[3] This is a weaker model of security where the attacker must specify what it will be challenged on before seeing the public parameters.

a surmountable one. We ultimately accomplish this by introducing a *nested dual system encryption approach* which allows us to make our information-theoretic argument in a very localized context, where the limited entropy of the public parameters is sufficient.

In a dual system encryption scheme, ciphertexts and keys can take two forms: normal and semi-functional. Normal keys can decrypt both normal and semi-functional ciphertexts, while semi-functional keys can only decrypt normal ciphertexts. Security is proven through a hybrid argument over a sequence of games, where first the challenge ciphertext is changed to semi-functional, and then the keys are changed to semi-functional one by one. At the end of this process, the simulator does not need to produce keys and ciphertexts which decrypt properly, and now security can be proven directly. However, we must avoid a potential paradox: at the point in the game sequence where a key is being changed to semi-functional, the simulator should not be able to test the nature of the key for itself by testing decryption on a semi-functional ciphertext. This can be enforced with nominal semi-functionality, meaning that if the simulator tries to make a semi-functional ciphertext which can be decrypted by the key of unknown type, then the key, ciphertext pair will actually be correlated so that decryption will succeed regardless of semi-functionality. In other words, even if semi-functional terms are present, they will cancel out upon decryption with the semi-functional ciphertext and hence be undetectable to the simulator. This nominal semi-functionality should be hidden from an attacker who cannot request keys capable of decrypting the ciphertext it receives.

The limited entropy of the public parameters in our systems does not enable us to hide nominal semi-functionality from the attacker if we try to change a key from normal to semi-functional in a single step. To overcome this, we introduce the concept of *ephemeral semi-functionality* for keys and ciphertexts. Ephemeral semi-functionality for keys is a temporary state which serves as an intermediate step between normalcy and semi-functionality. Ephemeral semi-functionality for ciphertexts is a temporary state of enhanced semi-functionality - ephemeral semi-functional keys can still decrypt semi-functional ciphertexts, but ephemeral semi-functional ciphertexts can only be decrypted by normal keys. Our proof employs a nested hybrid structure, where first the ciphertext is changed to semi-functional, then one key at a time is first changed to ephemeral semi-functional, then the ciphertext is changed to ephemeral semi-functional, and then the single key and ciphertext are both changed to semi-functional.

We note that a key first becomes incapable of decrypting ciphertexts when both are ephemeral semi-functional, and there is only *one* ephemeral semi-functional key at a time. This allows us to employ a information-theoretic argument to hide nominality in a more local context, where we need only be concerned with a single key. Even with this nested approach, accomplishing the game transitions with low entropy is still an intricate process - we employ additional inner hybrid steps to gradually change the distributions of keys and ciphertexts. In the KP-ABE setting, we also change to the selective security model.

Related Work Identity-Based Encryption was conceived by Shamir in [41] and first constructed by Boneh and Franklin [12] and Cocks [23]. These were proven secure in the random oracle model. Canetti, Halevi, and Katz [16] and Boneh and Boyen [9] then provided systems which were proven selectively secure in the standard model. Fully secure solutions in the standard model were later provided by Boneh and Boyen [10] and Waters [44]. The Waters system was efficient and proven from the well-established decisional Bilinear Diffie-Hellman assumption, but had public parameters consisting of $O(\lambda)$ group elements, where λ is the security parameter. The system provided by Gentry [24] had short public parameters and was proven secure in the standard model, but relied on a "q-type" assumption (meaning that the number of terms in the assumption depends on the number of queries q made by an attacker). Using dual system encryption, Waters [45] provided an efficient IBE system with short public parameters proven fully secure under the decisional linear and decisional bilinear Diffie-Hellman assumptions. In the random oracle model, additional schemes were provided by Boneh, Gentry, and Hamburg [13] under the quadratic residuosity assumption and by Gentry, Peikert, and Vaikuntanathan [25] under lattice-based assumptions.

Hierarchical Identity-Based Encryption was first introduced by Horwitz and Lynn [29] and constructed by Gentry and Silverberg [26] in the random oracle model. Selectively-secure constructions in the standard model were then provided by Boneh and Boyen [9] and Boneh, Boyen, and Goh [11]. The scheme of Boneh, Boyen, and Goh achieved short ciphertexts (ciphertext size independent of the hierarchy depth). Gentry and Halevi gave a fully secure construction for polynomial depth, relying on a complex assumption. Waters [45] provided a fully secure scheme from the decisional linear and decisional bilinear Diffie-Hellman assumptions. Lewko and Waters [34] provided a construction with short ciphertext, also achieving full security from static assumptions. Lattice-based HIBE systems were constructed by Cash, Hofheinz, Kiltz, and Peikert [17] and Agrawal, Boneh, and Boyen [1]. Agrawal, Boneh, and Boyen [2] constructed a lattice HIBE scheme where the dimension of the delegated lattices does not grow with the levels of the hierarchy. The lattice systems are proven either secure in the random oracle model or selectively secure in the standard model. Chatterjee and Sarkar [20] defined a couple of new security models for HIBE, and also suggested an HIBE system in a new, much weaker security model which can support arbitrary depths (i.e. a maximum depth is not fixed at setup). However, this system does not achieve even selective security - the authors point out that there is a simple attack against it in the standard selective security model.

Attribute-Based Encryption was introduced by Sahai and Waters [40]. Subsequently, Goyal, Pandey, Sahai, and Waters [28] defined two forms of ABE: Key-Policy ABE (where keys are associated with access policies and ciphertexts are associated with sets of attributes) and Ciphertext-Policy ABE (where ciphertexts are associated with access policies and keys are associated with sets of attributes). Several constructions of selectively secure KP-ABE and CP-ABE systems followed (e.g. [8, 21, 27, 28, 38, 39, 46]). Fully secure constructions were recently provided by Lewko, Okamoto, Sahai, Takashima, and Waters [31] and

Okamoto and Takashima [37]. The works of Chase [18] and Chase and Chow [19] considered the problem of ABE in a setting with multiple authorities. The related concept of Predicate Encryption was introduced by Katz, Sahai and Waters [30] and further studied in [31, 36, 37, 42]. Other works have considered related problems without addressing collusion resistance [3–5, 15, 35, 43].

The methodology of dual system encryption was introduced by Waters [45] and later used in [34, 31, 37, 22, 32] to obtain adaptive security (and also leakage resilience in [22, 32]) for IBE, HIBE, and ABE systems. The abstractions we provide for dual system encryption in the HIBE and ABE settings are similar to the abstractions provided in [32], except that we do not consider leakage resilience and also provide only selective security in the ABE case.

2 Dual System Encryption HIBE

We now define a Dual System Encryption HIBE scheme. (This is similar to the abstraction given in [32], but things are simpler in our case because we do not consider leakage resilience.) In addition to the five algorithms of a regular HIBE scheme (Setup, Encrypt, KeyGen, Decrypt, and Delegate), a Dual System Encryption HIBE scheme also has algorithms KeyGenSF and EncryptSF, which produce semi-functional keys and ciphertexts, respectively. Unlike the Setup, Encrypt, KeyGen, Decrypt, and Delegate algorithms, the KeyGenSF and EncryptSF algorithms need not run in polynomial time (given only their input parameters), since they are used only for the proof of security and are not used in the normal operation of the system. Notice that decryption will work as before unless both the secret key and ciphertext are semi-functional, in which case decryption will always fail.

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

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

EncryptSF($M, \boldsymbol{\mathcal{I}}$, PP) $\rightarrow \widetilde{\text{CT}}$ The semi-functional encryption algorithm takes a message M, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters PP as input. It produces a semi-functional ciphertext $\widetilde{\text{CT}}$.

KeyGen(MSK, $\boldsymbol{\mathcal{I}}$, PP) $\rightarrow \text{SK}_{\boldsymbol{\mathcal{I}}}$ The key generation algorithm takes the master secret key MSK, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters as input and outputs a secret key $\text{SK}_{\boldsymbol{\mathcal{I}}}$ for that identity vector.

*KeyGenSF(*MSK, $\boldsymbol{\mathcal{I}}$, PP) $\rightarrow \widetilde{\text{SK}}_{\boldsymbol{\mathcal{I}}}$ The semi-functional key generation algorithm takes the master secret key MSK, an identity vector $\boldsymbol{\mathcal{I}}$, and the public parameters as input. It produces a semi-functional secret key $\widetilde{SK}_{\boldsymbol{\mathcal{I}}}$ for $\boldsymbol{\mathcal{I}}$.

Decrypt$(\mathrm{CT}, \mathrm{PP}, \mathrm{SK}_{\boldsymbol{\mathcal{I}}}) \rightarrow M$ The decryption algorithm takes a ciphertext CT, the public parameters PP, and a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ as input. If the identity vector of the secret key $\boldsymbol{\mathcal{I}}$ is a prefix of the identity vector used to encrypt the ciphertext and the key and ciphertext are *not both semi-functional*, the decryption algorithm outputs the message M.

Delegate$(\mathrm{SK}_{\boldsymbol{\mathcal{I}}}, \mathcal{I}', \mathrm{PP}) \rightarrow \mathrm{SK}_{\boldsymbol{\mathcal{I}}:\mathcal{I}'}$ The delegation algorithm takes a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ for identity vector $\boldsymbol{\mathcal{I}}$, an identity $\mathcal{I}'$, and the public parameters PP as input. It outputs a secret key $\mathrm{SK}_{\boldsymbol{\mathcal{I}}:\mathcal{I}'}$ for the identity vector $\boldsymbol{\mathcal{I}} : \mathcal{I}'$, which denotes the concatenation of $\boldsymbol{\mathcal{I}}$ and $\mathcal{I}'$.

2.1 Security Properties for Dual System Encryption HIBE

We define four security properties for a dual system encryption HIBE. We will show that a system which has these four properties is a secure HIBE. To define these properties, we define the following variations of the security game for HIBE, which we call Game HIBE. In this game, the attacker may make Create, Delegate, and Reveal queries. In response to Create queries, the challenger creates the specified key. In response to Delegate queries, the challenger applies the delegation algorithm to produce the requested key from a specified superior key. In response to Reveal queries, the challenger gives the requested key to the attacker. For background on HIBE and its security definition and proofs of the theorems in this section, see the full version of this paper [33].

We first define Game HIBE_{WD} to be the same as Game HIBE, except without delegation. More precisely, instead of making Create, Delegate, and Reveal queries, the attacker simply makes KeyGen queries - i.e. it provides the challenger with an identity vector, the challenger creates a secret key for this identity vector by calling KeyGen, and then gives the secret key to the attacker. The only restriction is that no queried identity vectors can be prefixes of the challenge identity vector provided for the challenge ciphertext.

We next define Game HIBE_{C} to be the same as Game HIBE_{WD}, except that the challenge ciphertext is generated by a call to EncryptSF instead of Encrypt (i.e. a semi-functional ciphertext is given to the attacker). We also define Game HIBE_{SF} to be the same as Game HIBE_{C}, except that the challenger replaces all KeyGen calls with calls to KeyGenSF. In other words, the challenge ciphertext and all the secret keys given to the attacker will be semi-functional.

Delegation Invariance We say a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *delegation invariance* if for any PPT algorithm $\mathcal{A}$, there exists another PPT algorithm $\mathcal{A}'$ such that the advantage of $\mathcal{A}$ in Game HIBE is negligibly close to the advantage of $\mathcal{A}'$ in Game HIBE_{WD}. (Here, $\mathcal{A}$ makes Create, Delegate, and Reveal queries, while $\mathcal{A}'$ makes KeyGen queries.) We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE}(\lambda) - Adv_{\mathcal{A}'}^{HIBE_{WD}}(\lambda) \right| = negl(\lambda).$$

Semi-functional Ciphertext Invariance We say a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *semi-functional ciphertext invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_{WD} is negligibly close to its advantage in Game HIBE_C. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_{WD}}(\lambda) - Adv_{\mathcal{A}}^{HIBE_C}(\lambda) \right| = negl(\lambda).$$

Semi-functional Key Invariance We say a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *semi-functional key invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_C is negligibly close to its advantage in Game HIBE_{SF}. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_C}(\lambda) - Adv_{\mathcal{A}}^{HIBE_{SF}}(\lambda) \right| = negl(\lambda).$$

Semi-functional Security We say a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *semi-functional security* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_{SF} is negligible. We denote this by:

$$Adv_{\mathcal{A}}^{HIBE_{SF}}(\lambda) = negl(\lambda).$$

Theorem 1. *If a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has delegation invariance, semi-functional ciphertext invariance, semi-functional key invariance, and semi-functional security, then Π = (Setup, Encrypt, KeyGen, Decrypt, Delegate) is a secure HIBE scheme.*

2.2 An Alternative Security Property

The semi-functional key invariance property can be difficult to prove directly. For this reason, we define an alternative property, *one semi-functional key invariance*, which is more convenient to work with and which implies semi-functional key invariance through a hybrid argument.

To define one semi-functional key invariance, we must define an additional game, Game HIBE_b (where b represents a bit that can take value 0 or 1). In this game, when the attacker requests a key, it specifies whether it wants a normal or semi-functional key. If the attacker requests a normal key, the challenger makes a call to KeyGen to generate the key and returns it to the attacker. If the attacker requests a semi-functional key, the challenger makes a call to KeyGenSF to generate the key and returns it to the attacker. At some point, the attacker specifies a *challenge key*. In response, the challenger provides a normal key if $b = 0$ and a semi-functional key if $b = 1$. When the attacker requests the challenge ciphertext, it is given a semi-functional ciphertext (under the usual restriction that no key given to the attacker can be for an identity

vector which is a prefix of the identity vector of the ciphertext). Note that the only difference between Game HIBE_0 and Game HIBE_1 is the nature of a single key specified by the attacker.

One Semi-functional Key Invariance We say a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has *one semi-functional key invariance* if for any PPT algorithm $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game HIBE_0 is negligibly close to its advantage in Game HIBE_1. We denote this by:

$$\left| Adv_{\mathcal{A}}^{HIBE_0}(\lambda) - Adv_{\mathcal{A}}^{HIBE_1}(\lambda) \right| = negl(\lambda).$$

Theorem 2. *If a dual system encryption HIBE scheme Π_D = (Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt, Delegate) has one semi-functional key invariance, then it has semi-functional key invariance.*

3 Complexity Assumptions

Our construction will use composite order bilinear groups, first introduced in [14]. Additional background about these groups can be found in the full version. We let G denote a bilinear group order $N = p_1 p_2 p_3$, which is a product of three distinct primes, and we let $e : G \times G \to G_T$ denote the bilinear map.

In the assumptions below, we let G_{p_1} denote the subgroup of order p_1 in G, for example. We note that if $g_i \in G_{p_i}$ and $g_j \in G_{p_j}$ for $i \neq j$, then $e(g_i, g_j) = 1$. We use the notation $X \xleftarrow{R} S$ to express that X is chosen uniformly randomly from the finite set S. We note that except for Assumption 2, all of these assumptions are special cases of the General Subgroup Decision Assumption defined in [7]. Informally, the General Subgroup Decision Assumption can be described as follows: in a bilinear group of order $N = p_1 p_2 \ldots p_n$, there is a subgroup of order $\prod_{i \in S} p_i$ for each subset $S \subseteq \{1, \ldots, n\}$. We let S_0, S_1 denote two such subsets. It should be hard to distinguish a random element from the subgroup corresponding to S_0 from a random element of the subgroup corresponding to S_1, even if one is given random elements from subgroups corresponding to other sets S_i which satisfy either that $S_0 \cap S_i = \emptyset = S_1 \cap S_i$ or $S_0 \cap S_i \neq \emptyset \neq S_1 \cap S_i$. The formal statements of our precise assumptions are below. Assumption 1 here is a slightly weaker form of Assumption 1 in [34], and Assumptions 2 and 4 here also appeared in [34]. In our proofs, we will also invoke Assumption 4 with the roles of p_2 and p_3 reversed.

Assumption 1 Given a group generator $\mathcal{G}$, we define the following distribution:

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

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

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

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

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

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

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$
$$g \xleftarrow{R} G_{p_1},\ g_2, X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ \alpha, s \xleftarrow{R} \mathbb{Z}_N$$
$$D = (\mathbb{G}, g, g_2, g_3, g^{\alpha} X_2, g^{s} Y_2),$$
$$T_1 = e(g, g)^{\alpha s},\ T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 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 say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

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

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$
$$g, X_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ X_3 \xleftarrow{R} G_{p_3}$$
$$D = (\mathbb{G}, g, g_2, X_1 X_3),$$
$$T_1 \xleftarrow{R} G_{p_1},\ T_2 \xleftarrow{R} G_{p_1 p_3}.$$

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

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

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

Assumption 4 Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$
$$g, X_1 \xleftarrow{R} G_{p_1},\ X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3, Y_3 \xleftarrow{R} G_{p_3}$$
$$D = (\mathbb{G}, g, g_3, X_1 X_2, Y_2 Y_3),$$
$$T_1 \xleftarrow{R} G_{p_1 p_3},\ T_2 \xleftarrow{R} G.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be:

$$Adv4_{\mathcal{G},\mathcal{A}}(\lambda) := \Big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\Big|.$$

We say that $\mathcal{G}$ satisfies Assumption 4 if $Adv4_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

4 Our HIBE Construction

We now present our dual system encryption HIBE scheme. Our system is constructed in a composite order bilinear group whose order N is the product of three distinct primes. We assume that our identity vectors have components which are elements of $\mathbb{Z}_N$. In the semi-functional algorithms below, we let g_2 denote a generator of G_{p_2} and g_3 denote a generator of G_{p_3}.

We will assume that identity vectors are encoded such that if identity vector $\boldsymbol{\mathcal{I}}$ is not a prefix of identity vector $\boldsymbol{\mathcal{I}}^*$, then the *last* component of $\boldsymbol{\mathcal{I}}$ is not equal to *any* component of $\boldsymbol{I}^*$. In other words, when $\boldsymbol{\mathcal{I}} = (\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is not a prefix of $\boldsymbol{\mathcal{I}}^* = (\mathcal{I}_1^*, \ldots, \mathcal{I}_\ell^*)$, we assume that $\mathcal{I}_j \neq \mathcal{I}_k^*$ for all $k \in \{1, \ldots, \ell\}$. A simple scheme to achieve this encoding is to replace an arbitrary vector of component identities, $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ by concatenating in each entry with all the previous entries: $(\mathcal{I}_1, \mathcal{I}_1||\mathcal{I}_2, \ldots, \mathcal{I}_1||\mathcal{I}_2||\cdots||\mathcal{I}_j)$. This creates entries which grow in length, but we can avoid this by applying a collision-resistant hash function to each of them.

The main idea of our construction is to employ a secret-sharing approach across the levels of our secret keys. A user's secret key involves a sharing of the master secret key α as a sum of exponents, where each piece of the sum is additionally blinded by a random term which is unique to that piece. In other words, each share of α is blinded by randomness which is "local" to that share. To successfully decrypt, a user must effectively unblind each share, which can only be accomplished by a user with a j^{th} level identity vector which matches the ciphertext identity vector in *all of the components one through* j. If a user's identity vector fails to match in component $k \leq j$, then the user will fail to recover the k^{th} share needed, thus preventing successful decryption. In essence, each level of the key and ciphertext closely resembles an instance of the Boneh-Boyen IBE scheme [9] with an added layer of local randomness between the shares of the master secret key and the terms involving the identities. These instances share the same public parameters, which we are able to accommodate by using fresh local randomness in the levels of the key and ciphertext.

4.1 Construction

Setup$(\lambda) \rightarrow \mathrm{PP}, \mathrm{MSK}$ The setup algorithm takes in the security parameter λ and chooses a bilinear group G of order $N = p_1 p_2 p_3$, where p_1, p_2, p_3 are distinct primes. We let G_{p_i} denote the subgroup of order p_i in G. The algorithm then chooses g, u, h, v, w uniformly randomly from G_{p_1}, and α uniformly randomly from $\mathbb{Z}_N$. It sets the public parameters as:

$$PP := \{N, G, g, u, h, v, w, e(g,g)^\alpha\}.$$

The master secret key is α.

Encrypt$(M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}), \rightarrow \mathrm{CT}$ The encryption algorithm chooses s, $t_1, \ldots, t_j$ uniformly randomly from $\mathbb{Z}_N$. It creates the ciphertext as:

$$C := Me(g,g)^{\alpha s},\ C_0 := g^s,$$

$$C_{i,1} := w^s v^{t_i},\ C_{i,2} := g^{t_i},\ C_{i,3} := (u^{\mathcal{I}_i} h)^{t_i} \quad \forall i \in \{1, \ldots, j\}.$$

EncryptSF($M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}$*)* $\rightarrow \widetilde{\mathrm{CT}}$ The semi-functional encryption algorithm first calls the Encrypt algorithm to obtain a normal ciphertext, $\mathrm{CT} = \{C', C'_0, C'_{i,1}, C'_{i,2}, C'_{i,3}\ \forall i\}$. It then chooses random values $\gamma, \delta \in \mathbb{Z}_N$. It forms the semi-functional ciphertext $\widetilde{\mathrm{CT}}$ as:

$$C := C',\ C_0 := C'_0 \cdot g_2^{\gamma},$$
$$C_{i,1} := C'_{i,1} \cdot g_2^{\delta},\ C_{i,2} := C'_{i,2},\ C_{i,3} := C'_{i,3} \quad \forall i \in \{1, \ldots, j\}.$$

Notice that the additional term g_2^{δ} on $C_{i,1}$ is the *same for each value of* i.

KeyGen($(\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}$*)* $\rightarrow \mathrm{SK}_{\boldsymbol{\mathcal{I}}}$ The key generation algorithm chooses uniformly random values $r_1, \ldots, r_j, y_1, \ldots, y_j$ from $\mathbb{Z}_N$. It also chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ subject to the constraint that $\alpha = \lambda_1 + \lambda_2 + \cdots + \lambda_j$. The secret key is created as:

$$K_{i,0} := g^{\lambda_i} w^{y_i},\ K_{i,1} := g^{y_i},\ K_{i,2} := v^{y_i} (u^{\mathcal{I}_i} h)^{r_i},\ K_{i,3} := g^{r_i} \quad \forall i \in \{1, \ldots, j\}.$$

KeyGenSF($(\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}$*)* $\rightarrow \widetilde{\mathrm{SK}}_{\boldsymbol{\mathcal{I}}}$ The first time this algorithm is called, it chooses random values $\sigma, \psi \in \mathbb{Z}_N$. These values will be stored and used on each invocation of the algorithm.

To create a semi-functional key, the semi-functional key generation algorithm first calls the KeyGen algorithm to obtain a normal key, $\mathrm{SK}_{\boldsymbol{\mathcal{I}}} = \{K'_{i,0}, K'_{i,1}, K'_{i,2}, K'_{i,3}\ \forall i\}$. It then chooses a random value $\tilde{y}_j \in \mathbb{Z}_N$ and creates the semi-functional key as:

$$K_{i,0} := K'_{i,0},\ K_{i,1} := K'_{i,1},\ K_{i,2} := K'_{i,2},\ K_{i,3} := K'_{i,3} \quad \forall i \in \{1, \ldots, j-1\},$$
$$K_{j,0} := K'_{j,0} \cdot (g_2 g_3)^{\psi \tilde{y}_j},\ K_{j,1} := K'_{j,1} \cdot (g_2 g_3)^{\tilde{y}_j},$$
$$K_{j,2} := K'_{j,2} \cdot (g_2 g_3)^{\sigma \tilde{y}_j},\ K_{j,3} := K'_{j,3}.$$

We note that the $\tilde{y}_j$ terms are chosen to be freshly random for each key, while the values σ, ψ are shared by all semi-functional keys. We also note that the exponents modulo p_3 here are uncorrelated from the exponents modulo p_2 by the Chinese Remainder Theorem. It is also important to observe that the semi-functional components (the added terms in G_{p_2} and G_{p_3}) only appear in the *last level* of the key.

Delegate($\mathrm{PP}, \mathrm{SK}, \mathcal{I}_{j+1}$*)* $\rightarrow \mathrm{SK}'$ The delegation algorithm takes in a secret key $\mathrm{SK} = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}\ \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a level $j+1$ identity $\mathcal{I}_{j+1}$. It produces a secret key SK' for $(\mathcal{I}_1, \ldots, \mathcal{I}_{j+1})$ as follows. It chooses $y'_1, \ldots, y'_{j+1}$ and $r'_1, \ldots, r'_{j+1} \in \mathbb{Z}_N$ uniformly at random, $\lambda'_1, \ldots, \lambda'_{j+1} \in \mathbb{Z}_N$ randomly up to the constraint that $\lambda'_1 + \cdots + \lambda'_{j+1} = 0$ and computes:

$$K'_{i,0} := K_{i,0} \cdot g^{\lambda'_i} \cdot w^{y'_i},\ K'_{i,1} := K_{i,1} \cdot g^{y'_i},\ K'_{i,2} := K_{i,2} \cdot v^{y'_i} (u^{\mathcal{I}_i} h)^{r'_i},$$
$$K'_{i,3} := K_{i,3} \cdot g^{r'_i}\ \forall i \in \{1, \ldots, j+1\},$$

where $K_{j+1,1}, K_{j+1,2}, K_{j+1,3}$ are defined to be the identity element in G.

Decryption$(CT, \mathrm{SK}) \to M$ The decryption algorithm takes in a secret key $\mathrm{SK} = \{K_{i,0}, K_{i,1}, K_{i,2}, K_{i,3}\ \forall i \in \{1, \ldots, j\}\}$ for $(\mathcal{I}_1, \mathcal{I}_2, \ldots, \mathcal{I}_j)$ and a ciphertext CT encrypted to $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$. Assuming $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$ is a prefix of $(\mathcal{I}_1, \ldots, \mathcal{I}_\ell)$, the message is decrypted as follows. The decryption algorithm computes:

$$B := \prod_{i=1}^{j} \frac{e(C_0, K_{i,0})e(C_{i,2}, K_{i,2})}{e(C_{i,1}, K_{i,1})e(C_{i,3}, K_{i,3})}.$$

The message is then computed as $M = C/B$.

Correctness We observe that:

$$B = \prod_{i=1}^{j} \frac{e(g,g)^{s\lambda_i} e(g,w)^{sy_i} e(g,v)^{t_i y_i} e(g, u^{\mathcal{I}_i} h)^{t_i r_i}}{e(w,g)^{sy_i} e(v,g)^{t_i y_i} e(u^{\mathcal{I}_i} h, g)^{t_i r_i}},$$

which is equal to:

$$= \prod_{i=1}^{j} e(g,g)^{s\lambda_i} = e(g,g)^{s\alpha},$$

since $\sum_{i=1}^{j} \lambda_i = \alpha$. Thus, $M = C/B$.

5 Security

We prove that our dual system encryption HIBE scheme has delegation invariance, semi-functional ciphertext invariance, one semi-functional key invariance, and semi-functional security. By theorems 1 and 2, this implies that our HIBE system is secure. More formally, we prove the following theorem:

Theorem 3. *Under Assumptions 1-4, our HIBE system is fully secure.*

Proving delegation invariance, semi-functional ciphertext invariance, and semi-functional security is relatively straightforward, and these proofs can be found in the full version. The truly challenging part of the proof will be proving one semi-functional key invariance, and this is where we introduce our key technical innovations.

5.1 One Semi-functional Key Invariance

The primary challenge in proving one semi-functional key invariance for our system is that the repetition of the same public parameters for each level of the keys and ciphertexts severely limits our ability to simulate properly distributed semi-functional keys and ciphertexts as we are changing the form of the challenge key. In a typical dual system encryption argument, we must ensure as we are changing the form of one key that the simulator cannot determine the nature of the key for itself. Since the simulator must be prepared to make a semi-functional

ciphertext for any identity vector and also must be prepared to use any identity vector for the challenge key, it seems that a simulator could learn for itself whether or not the challenge key is semi-functional by trying to decrypt a semi-functional ciphertext. This potential paradox can be avoided by ensuring that a simulator can only make a nominally semi-functional key, meaning that even if semi-functional terms are present on the challenge key, they will be correlated with the semi-functional ciphertext and cancel out upon decryption. We would then argue that nominality is hidden from an attacker who cannot request keys capable of decrypting.

If we attempt to change the challenge key in our system from normal to semi-functional in a single or very small number of steps using generalized subgroup decision assumptions, then the very limited entropy available in the public parameters seems to prevent us from maintaining the proper distributions of the semi-functional keys and semi-functional ciphertext without revealing nominality. In other words, it appears to be difficult for the simulator to prevent information-theoretic exposure of unwanted correlations between the semi-functional components of the keys and ciphertext it creates.

To overcome this difficulty, we employ a *nested dual system encryption approach* and introduce the concept of *ephemeral semi-functionality*[4]. Instead of trying to directly change the challenge key from normal to semi-functional, we will first change it from normal to *ephemeral semi-functional*. An ephemeral semi-functional key will come from a new distribution which serves as an intermediary stage between the normal and semi-functional distributions. We note that an ephemeral semi-functional key can still correctly decrypt semi-functional ciphertexts, and that its form only differs from a normal key on its last level.

After changing the challenge key from normal to ephemeral semi-functional, we will then change the ciphertext to also be ephemeral semi-functional. Ephemeral semi-functional ciphertexts will come from a new distribution of ciphertexts, and will *not* be decryptable by ephemeral semi-functional keys. This is where we confront the potential paradox of dual system encryption: we will make sure that the simulator can only make challenge key and ciphertext pairs which are *nominally ephemeral semi-functional*, meaning that the distributions of the challenge key and ciphertext will be correlated so that even if the ephemeral semi-functional terms are present in both the key and ciphertext, they will cancel out upon decryption. This correlation will be hidden from an attacker who cannot request a key capable of decrypting the ciphertext.

To accomplish this information-theoretic hiding with such low entropy in our public parameters, we will make a hybrid argument in which we change the ciphertext form one level at a time. Since there are only ephemeral semi-functional terms on one level of one key, it is now sufficient to hide a correlation between one level of the ciphertext and one level of one key: this can be accomplished with the use of a pairwise independent function. Once we have obtained an ephemeral

[4] We choose not to include this concept in our abstraction for dual system encryption HIBE, because its use here is motivated by the particular challenge of short public parameters and we imagine dual system encryption HIBE as a broader framework.

semi-functional challenge key and an ephemeral semi-functional ciphertext, we are able to change the challenge key to be semi-functional in the usual sense and also return the ciphertext to its usual semi-functional state.

Essentially, using ephemeral semi-functionality helps us overcome the challenge presented by low entropy in the public parameters because it allows us to move the information-theoretic argument that nominality is hidden from the attacker to a setting where we are really only concerned with *one key*. Since the other semi-functional keys come from a different distribution, we can prevent them from leaking information about the simulated ephemeral distribution that would break the information-theoretic argument.

We now define the distributions of ephemeral semi-functional keys and ciphertexts. We do this by defining two new algorithms, EncryptESF and KeyGenESF. Like the algorithms EncryptSF and KeyGenSF, these do not need to run in polynomial time (given only their input parameters). We note that the EncryptESF algorithm takes in an additional parameter σ: this is because the ciphertexts it produces will share the value σ with the semi-functional keys created by KeyGenSF. As in the original semi-functional algorithms, we let g_2 denote a generator of G_{p_2} and g_3 denote a generator of G_{p_3}.

EncryptESF$(M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}, \sigma) \to \widetilde{\mathrm{CT}}_E$ The ephemeral semi-functional encryption algorithm first calls the Encrypt algorithm to obtain a normal ciphertext
$\mathrm{CT} = \{C', C'_0, C'_{i,1}, C'_{i,2}, C'_{i,3}\ \forall i \in \{1, \ldots, j\}\}$. It then chooses random values $\gamma, \delta, a', b', t_1, \ldots, t_j \in \mathbb{Z}_N$ and forms the ephemeral semi-functional ciphertext $\widetilde{\mathrm{CT}}_E$ as:

$$C := C', C_0 := C'_0 \cdot g_2^{\gamma},$$

$$C_{i,1} := C'_{i,1} \cdot g_2^{\delta} \cdot g_2^{\sigma t_i},\ C_{i,2} := C'_{i,2} \cdot g_2^{t_i},\ C_{i,3} := C'_{i,3} \cdot g_2^{(a'\mathcal{I}_i + b')t_i} \quad \forall i \in \{1, \ldots, j\}.$$

KeyGenESF$((\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}, \sigma) \to \widetilde{\mathrm{SK}}_E$ The ephemeral semi-functional key generation algorithm first calls the KeyGen algorithm to obtain a normal key
$\mathrm{SK} = \{K'_{i,0}, K'_{i,1}, K'_{i,2}, K'_{i,3}\ \forall i \in \{1, \ldots, j\}\}$. It chooses random values $\tilde{r}_1, \tilde{r}_2 \in \mathbb{Z}_N$ and forms the ephemeral semi-functional key $\widetilde{\mathrm{SK}}_E$ as:

$$K_{i,0} := K'_{i,0},\ K_{i,1} := K'_{i,1},\ K_{i,2} := K'_{i,2},\ K_{i,3} := K'_{i,3}\ \forall i \in \{1, \ldots, j-1\},$$

$$K_{j,0} := K'_{j,0},\ K_{j,1} = K'_{j,1},\ K_{j,2} := K'_{j,2} \cdot (g_2 g_3)^{\tilde{r}_1},\ K_{j,3} := K'_{j,3} \cdot (g_2 g_3)^{\tilde{r}_2}.$$

We note that an ephemeral semi-functional key can decrypt a semi-functional ciphertext, but cannot decrypt an ephemeral semi-functional ciphertext, while an ephemeral semi-functional ciphertext can only be decrypted by normal keys.

Sequence of Games We prove one semi-functional key invariance of our dual system encryption HIBE scheme via a hybrid argument over the following sequence of games. We begin with Game HIBE_0, where the ciphertext is semi-functional and the challenge key is normal. We will end with Game HIBE_1,

where the ciphertext is semi-functional and the challenge key is semi-functional. We define the following intermediary games. In these games, the distributions of the challenge key and ciphertext vary, while the distribution of the requested normal and semi-functional keys are the same as in Games HIBE_0 and HIBE_1.

Game HIBE$'_0$ This game is exactly like Game HIBE_0, except for the added restriction that the last component of the challenge key identity vector cannot be equal to any of the components of the challenge ciphertext identity vector modulo p_3 (note that we were already requiring this modulo N - now we make the stronger requirement that the identities must remain unequal when we reduce modulo p_3). (This added restriction will be needed to apply pairwise independence arguments in $\mathbb{Z}_{p_3}$.)

Game EK In Game EK, the ciphertext is still semi-functional, and the challenge key is now ephemeral semi-functional. We retain the added restriction on the identities modulo p_3.

Game EC In Game EC, *both* the ciphertext and challenge are ephemeral semi-functional. We retain the added restriction on the identities modulo p_3.

Game HIBE$'_1$ This game is exactly like the Game HIBE_1, but with the added restriction on the identities modulo p_3.

In the full version, we prove that we can transition from Game HIBE_0 to Game HIBE'_0, to Game EK, to Game EC, to Game HIBE'_1, and finally to Game HIBE_1 without the attacker's advantage changing by a non-negligible amount.

6 Key-Policy Attribute-Based Encryption

We now present our construction for KP-ABE. Our public parameters consist of a constant number of elements from a bilinear group of composite order N, while our attribute universe is $\mathbb{Z}_N$. Ciphertexts in our system are associated with sets of attributes, while secret keys are associated with LSSS access matrices. Our construction is closely related to our HIBE construction. The main changes are that attributes have now replaced identities, and the master secret key α is now shared according to the LSSS matrix, instead of as a sum. We follow the convention that to share a value α, one employs a vector $\boldsymbol{\alpha}$ with first coordinate equal to α, and the shares are obtained by multiplying the rows of the LSSS matrix by the sharing vector $\boldsymbol{\alpha}$. A subset of rows is capable of reconstructing the shared secret if and only if their span includes the vector $(1, 0, \ldots, 0)$.

6.1 Construction

Setup$(\lambda) \rightarrow \text{PP}, \text{MSK}$ The setup algorithm takes in the security parameter λ and chooses a suitable bilinear group G of order $N = p_1 p_2 p_3$, a product of

three distinct primes. It chooses $\alpha \in \mathbb{Z}_N$ uniformly randomly, and also chooses uniformly random elements g, u, h, v, w from the subgroup G_{p_1}. It sets the public parameters as:

$$\text{PP} := \{N, G, g, u, h, v, w, e(g,g)^\alpha\}.$$

The MSK is α, and the universe U of attributes is $\mathbb{Z}_N$.

Encrypt($M, S \subseteq U, \text{PP}) \to \text{CT}$ The encryption algorithm takes in a message M, a set of attributes S, and the public parameters. We let ℓ denote the size of the set S, and we let $s_1, \ldots, s_\ell \in \mathbb{Z}_N$ denote the elements of S. The encryption algorithm chooses uniformly random values $s, t_1, \ldots, t_\ell \in \mathbb{Z}_N$ and computes the ciphertext as:

$$C := Me(g,g)^{\alpha s},\ C_0 := g^s,$$
$$C_{s_i,1} := w^s v^{t_i},\ C_{s_i,2} := g^{t_i},\ C_{s_i,3} := (u^{s_i} h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

(We also assume the set of S is given as part of the ciphertext.)

KeyGen($\text{MSK}, \text{PP}, (A, \rho)) \to \text{SK}$ The key generation algorithm takes in the master secret key α, the public parameters, and a LSSS matrix (A, ρ), where A is an $n \times m$ matrix over $\mathbb{Z}_N$, and ρ maps each row of A to an attribute in $\mathbb{Z}_N$. The key generation algorithm chooses a random vector $\boldsymbol{\alpha} \in \mathbb{Z}_N^m$ with first coordinate equal to α and random values $r_1, \ldots, r_n, y_1, \ldots, y_n \in \mathbb{Z}_N$. For each $x \in \{1, \ldots, n\}$, we let A_x denote the x^{th} row of A, and we let $\rho(x)$ denote that attribute associated with this row by the mapping ρ. We let $\lambda_x := A_x \cdot \boldsymbol{\alpha}$ denote the share associated with the row A_x of A. The secret key is formed as[5]:

$$K_{x,0} := g^{\lambda_x} w^{y_x}, K_{x,1} := g^{y_x}, K_{x,2} := v^{y_x} (u^{\rho(x)} h)^{r_x}, K_{x,3} := g^{r_x}\ \forall x \in \{1, \ldots, n\}.$$

Decrypt($\text{SK}, \text{CT}) \to M$ The decryption algorithm takes in a ciphertext CT for attribute set S and a secret key SK for access matrix (A, ρ). If the attributes of the ciphertext satisfy the policy of the secret key, then it will compute the message M as follows. First, it computes constants ω_x such that $\sum_{\rho(x) \in S} \omega_x A_x = (1, 0, \ldots, 0)$. It then computes:

$$B = \prod_{\rho(x) \in S} \left(\frac{e(C_0, K_{x,0}) e(C_{\rho(x),2}, K_{x,2})}{e(C_{\rho(x),1}, K_{x,1}) e(C_{\rho(x),3}, K_{x,3})} \right)^{\omega_x},\ M = C/B.$$

Correctness We observe that:

$$B = \prod_{\rho(x) \in S} \left(\frac{e(g,g)^{s\lambda_x} e(g,w)^{s y_x} e(g,v)^{t_{\rho(x)} y_x} e(g, u^{\rho(x)} h)^{t_{\rho(x)} r_x}}{e(w,g)^{s y_x} e(v,g)^{t_{\rho(x)} y_x} e(u^{\rho(x)} h, g)^{t_\rho(x) r_x}} \right)^{\omega_x},$$

$$= \prod_{\rho(x) \in S} \left(e(g,g)^{s\lambda_x} \right)^{\omega_x} = e(g,g)^{s\alpha}.$$

This shows that $M = C/B$.

[5] We also assume the access matrix (A, ρ) is given as part of the key.

6.2 Security

We prove that our system is selectively secure using a similar strategy to our proof of adaptive security for our HIBE system (the formal definition for selective security in the KP-ABE setting can be found in the full version). We were able to achieve adaptive security in the HIBE setting because we could assume that, regardless of what identity vector was chosen for the challenge ciphertext, each requested key would have a final identity component which would not match any components of the challenge ciphertext. This allowed us to put our semi-functional components only on the last level of the key, which was crucial to preserving the appearance of randomness via our pairwise independence argument in the middle stages of the proof. In the adaptive KP-ABE setting, we only know that the policy of a requested key will fail to be satisfied by the attribute set of the ciphertext, but we do not know *how* it will fail to be satisfied. In other words, for keys which are requested by the attacker before the challenge ciphertext, we do not know which rows of the keys will correspond to attributes which are not in the challenge ciphertext. This leaves us in a bind - we do not know where to put the semi-functional terms. If we try to put semi-functional terms on each row, we will not be able to make the semi-functional terms appear suitably random in the attacker's view. If we put the semi-functional terms on too few rows, we will not achieve a meaningful kind of semi-functionality.

This problem is solved by moving to the selective security model, which forces the attacker to reveal the attribute set of the challenge ciphertext at the very start of the game. This means that when the simulator is faced with a key request, it already knows which rows of the key correspond to attributes which are absent from the ciphertext, and it can place the semi-functional terms exactly on these rows. We must add an additional hybrid to our proof strategy here so that we can change the rows of a key from normal to semi-functional one at a time. The proof of the following theorem, as well as discussion of delegation capabilities for our ABE scheme, can be found in the full version.

Theorem 4. *Under Assumptions 1-4, our KP-ABE system is selectively secure.*

References

1. S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.
2. S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *CRYPTO*, pages 98–115, 2010.
3. S. Al-Riyami, J. Malone-Lee, and N. Smart. Escrow-free encryption supporting cryptographic workflow. In *Int. J. Inf. Sec.*, volume 5, pages 217–229, 2006.
4. W. Bagga, R. Molva, and S. Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS*, page 368, 2006.
5. M. Barbosa and P. Farshim. Secure cryptographic workflow in the standarad model. In *INDOCRYPT*, pages 379–393, 2006.
6. A. Beimel. Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

7. M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In *TCC*, 2011.
8. John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 321–334.
9. D. Boneh and X. Boyen. Efficient selective-id secure identity based encryption without random oracles. In *EUROCRYPT*, pages 223 – 238, 2004.
10. D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.
11. D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.
12. D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.
13. D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.
14. D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–342, 2005.
15. R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *ACM conference on Computer and Communications Security*, pages 146–157, 2004.
16. R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.
17. D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.
18. M. Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.
19. M. Chase and S. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.
20. S. Chatterjee and P. Sarkar. Generalization of the selective-id security model for hibe protocols. In *Public Key Cryptography*, pages 241–256, 2006.
21. L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *ACM conference on Computer and Communications Security*, pages 456–465, 2007.
22. S. Chow, Y. Dodis, Y. Rouselakis, and B. Waters. Practical leakage-resilient identity-based encryption from simple assumptions.
23. C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proceedings of the 8th IMA International Conference on Cryptography and Coding*, pages 26–28, 2001.
24. C. Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.
25. C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *Proceedings of the 40th annual ACM Symposium on Theory of Computing*, pages 197–206, 2008.
26. C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.
27. V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP*, pages 579–591, 2008.
28. V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute based encryption for fine-grained access control of encrypted data. In *ACM conference on Computer and Communications Security*, pages 89–98, 2006.

29. J. Horwitz and B. Lynn. Toward hierarchical identity-based encryption. In *EUROCRYPT*, pages 466–481, 2002.
30. J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
31. A. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
32. A. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. In *TCC*, 2011.
33. A. Lewko and B. Waters. Unbounded HIBE and attribute-based encryption. Cryptology ePrint Archive, Report 2011/049, 2011. `http://eprint.iacr.org/`.
34. A. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.
35. G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, 2003.
36. T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214–231, 2009.
37. T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
38. R. Ostrovksy, A. Sahai, and B. Waters. Attribute based encryption with non-monotonic access structures. In *ACM conference on Computer and Communications Security*, pages 195–203, 2007.
39. M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure attribute-based systems. In *ACM conference on Computer and Communications Security*, pages 99–112, 2006.
40. A. Sahai and B. Waters. Fuzzy identity based encryption. In *EUROCRYPT*, pages 457–473, 2005.
41. A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.
42. E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *Automata, Languages and Programming*, pages 560–578, 2008.
43. N. Smart. Access control using pairing based cryptography. In *CT-RSA*, pages 111–121, 2003.
44. B. Waters. Efficient identity-based ecnryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.
45. B. Waters. Dual system encryption: realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.
46. B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, 2011.