Unbounded HIBE and Attribute-Based Encryption

Allison Lewko [*]
University of Texas at Austin
alewko@cs.utexas.edu

Brent Waters [†]
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 [39] 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.

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

[*] Supported by a 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.

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, 44, 33] 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 $\vec{v}^1 = (v_1^1, \ldots, v_d^1), \ldots, \vec{v}^k = (v_1^k, \ldots, v_d^k) \in \mathbb{Z}_p^d$, 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 $\vec{v}^1, \ldots, \vec{v}^k$ 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[1] from the same static, generically secure assumptions in composite order bilinear groups.

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

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 [44]. 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 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.

1.1 Related Work

Identity-Based Encryption was conceived by Shamir in [40] 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 [43]. 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 depending on the number of queries q made by an attacker). Using dual system encryption, Waters [44] 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 [44] provided a fully secure scheme from the decisional linear and decisional bilinear Diffie-Hellman assumptions. Lewko and Waters [33] 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 [39]. 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, 37, 38, 45]). Fully secure constructions were recently provided by Lewko, Okamoto, Sahai, Takashima, and Waters [31] and Okamoto and Takashima [36]. 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, 35, 36, 41]. Other works have considered related problems without addressing collusion resistance [3, 4, 5, 15, 34, 42].

The methodology of dual system encryption was introduced by Waters [44] and later used in [33, 31, 36, 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.

1.2 Organization

In Section 2, we provide the necessary background for HIBE schemes and define dual system encryption HIBE schemes as an abstraction. We also provide the necessary background on composite order bilinear groups and state our complexity assumptions. In Section 3, we present our HIBE construction. In section 4, we prove its security. In section 5, we present our KP-ABE construction. The relevant background and proof of security for the KP-ABE scheme appears in the Appendix.

2 Background

2.1 Hierarchical Identity-Based Encryption

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

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

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

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

Decrypt$(\mathrm{CT}, \mathrm{PP}, \mathrm{SK}_{\vec{\mathcal{I}}}) \rightarrow M$ The decryption algorithm takes a ciphertext CT, the public parameters PP, and a secret key $\mathrm{SK}_{\vec{\mathcal{I}}}$ as input. If the identity vector of the secret key, $\vec{\mathcal{I}}$, is a prefix of the identity vector used to encrypt the ciphertext, the decryption algorithm outputs the message M.

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

Security Definition We use the complete form of the security definition (given in [41]), which keeps track of how keys are generated and delegated. Security is defined through the following game between a challenger and an attacker. We call the Game HIBE.

Setup The challenger runs the Setup algorithm to generate the public parameters PP and master secret key MSK. It gives PP to the adversary. We let S denote the set of private keys that the challenger has created but not yet given to the adversary. We initialize $S = \emptyset$.

Phase 1 The adversary makes Create, Delegate, and Reveal key queries. To make a Create query, the attacker specifies an identity vector $\vec{\mathcal{I}}$. 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 $\mathrm{SK}_{\vec{\mathcal{I}}}$ in the set S and specifies an identity $\mathcal{I}'$. In response, the challenger appends $\mathcal{I}'$ to $\vec{\mathcal{I}}$ and makes a key for this new identity by running the delegation algorithm on $SK_{\vec{\mathcal{I}}}$ and $\mathcal{I}'$. 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 $\vec{\mathcal{I}}^*$. 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 $\vec{\mathcal{I}}^*$. 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 $\vec{\mathcal{I}}^*$.

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

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

Definition 1. *A Hierarchical Identity Based Encryption scheme is secure if all PPT adversaries achieve at most a negligible advantage (with respect to λ) in the above security game.*

2.2 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 defined above (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$(\lambda) \rightarrow \mathrm{PP}, \mathrm{MSK}$ The setup algorithm takes the security parameter λ as input and outputs the public parameters PP and the master secret key MSK.

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

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

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

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

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

2.3 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 secure (in the sense of Definition 1). To define these properties, we define the following variations of the Game HIBE described above.

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 $\Pi_D =$ (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 2. *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.*

Proof. We let $\mathcal{A}$ denote any PPT algorithm. We first note that in the real HIBE game, there are no calls to the semi-functional algorithms EncryptSF and KeyGenSF of Π_D. Hence, from $\mathcal{A}$'s perspective, playing the HIBE game with Π_D is the same as playing the HIBE game with Π. By delegation invariance, we have that:

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

By semi-functional ciphertext invariance, we have that:

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

By semi-functional key invariance, we have that:

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

Thus, by the triangle inequality, we may conclude that

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

By semi-functional security, we know that the quantity $Adv_{\mathcal{A}'}^{HIBE_{SF}}(\lambda)$ is negligible, hence $Adv_{\mathcal{A}}^{HIBE}(\lambda)$ must be negligible as well. Thus, the HIBE scheme Π is secure. □

2.4 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 3. *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.*

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ which achieves a non-negligible difference in advantage between Game HIBE$_C$ and Game HIBE$_{SF}$. Then we will show there must exist a PPT algorithm $\mathcal{B}$ which has a non-negligible difference in advantage between Game HIBE$_0$ and Game HIBE$_1$, contradicting one semi-functional key invariance. We let q denote the number of key queries that $\mathcal{A}$ makes. For k from 0 to q, we define Game HIBESF$_k$ as follows: the attacker receives a semi-functional ciphertext, semi-functional keys in response to the first k key requests, and normal keys in response to the remaining key requests. We note that Game HIBESF$_0$ is Game HIBE$_C$ and that Game HIBESF$_q$ is Game HIBE$_{SF}$.

Since $\mathcal{A}$ has a non-negligible difference in advantage between Game HIBE$_C$ and Game HIBE$_{SF}$ and q is polynomial, there must exist some value of k from 0 to $q-1$ such that

$$\left|Adv_{\mathcal{A}}^{HIBESF_k}(\lambda) - Adv_{\mathcal{A}}^{HIBESF_{k+1}}(\lambda)\right|$$

is non-negligible. Now, $\mathcal{B}$ works as follows. When it receives the public parameters from its challenger, it forwards these to $\mathcal{A}$. For the first k key requests that $\mathcal{A}$ makes, $\mathcal{B}$ forwards these to its challenger as requests for semi-functional keys, and returns the resulting semi-functional keys to $\mathcal{A}$. For the $k+1$ key request from $\mathcal{A}$, $\mathcal{B}$ forwards this to its challenger as the challenge key, and returns the resulting key to $\mathcal{A}$. For the remaining key requests from $\mathcal{A}$, $\mathcal{B}$ forwards these to its challenger as requests for normal keys, and returns the resulting keys to $\mathcal{A}$. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. Now, if $\mathcal{B}$ is playing Game HIBE$_0$, then $\mathcal{A}$ is playing Game HIBESF$_k$. If $\mathcal{B}$ is playing Game HIBE$_1$, then $\mathcal{A}$ is playing Game HIBESF$_{k+1}$. Hence, since $\mathcal{A}$ has a non-negligible difference in advantage between these two games, $\mathcal{B}$ has a non-negligible difference in advantage between Game HIBE$_0$ and Game HIBE$_1$. □

2.5 Composite Order Bilinear Groups

We will construct our system in composite order bilinear groups, first introduced in [14]. We let $\mathcal{G}$ denote a group generator, i.e. an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group G. In our case, we will define $\mathcal{G}$'s output as (N, G, G_T, e), where $N = p_1p_2p_3$ is a product of three distinct primes, G and G_T are cyclic groups of order N, and $e : G^2 \rightarrow G_T$ 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.

The group operations in G and G_T and the map e are computable in polynomial time with respect to λ, and the group descriptions of G and G_T include a generator of each group. 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 these subgroups are "orthogonal" to each other under the bilinear map e: i.e. if $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, then $e(h_i, h_j)$ is the identity element in G_T. If g_1 generates G_{p_1}, g_2 generates G_{p_2}, and g_3 generates G_{p_3}, then every element h of G can be expressed as $g_1^x g_2^y g_3^z$ for some values $x, y, z \in \mathbb{Z}_N$. We will refer to g_1^x as the "G_{p_1} part of h", for example.

2.6 Complexity Assumptions

In the assumptions below, we let $G_{p_1p_2}$ denote the subgroup of order p_1p_2 in G, for example. 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 $N = p_1p_2 \dots p_n$, there is a subgroup of order $\prod_{i \in S} p_i$ for each subset $S \subseteq \{1, \dots, n\}$. 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 [33], and Assumptions 2 and 4 here also appeared in [33]. 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:

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

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

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

$$T_1 \xleftarrow{R} G_{p_1p_2},\ T_2 \xleftarrow{R} G_{p_1}.$$

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:

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

$$g \xleftarrow{R} G_{p_1},\ g_2, X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ \alpha, s \xleftarrow{R} \mathbb{Z}_N$$

$$D = (\mathbb{G}, g, g_2, g_3, g^{\alpha}X_2, g^sY_2),$$

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

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:

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

$$g, X_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ X_3 \xleftarrow{R} G_{p_3}$$

$$D = (\mathbb{G}, g, g_2, X_1X_3),$$

$$T_1 \xleftarrow{R} G_{p_1},\ T_2 \xleftarrow{R} G_{p_1p_3}.$$

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:

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

$$g, X_1 \xleftarrow{R} G_{p_1},\ X_2, Y_2 \xleftarrow{R} G_{p_2},\ g_3, Y_3 \xleftarrow{R} G_{p_3}$$

$$D = (\mathbb{G}, g, g_3, X_1X_2, Y_2Y_3),$$

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

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 $\mathcal{A}$.

3 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 $\vec{\mathcal{I}}$ is not a prefix of identity vector $\vec{\mathcal{I}}^*$, then the *last* component of $\vec{\mathcal{I}}$ is not equal to *any* component of $\vec{I}^*$. In other words, when $\vec{\mathcal{I}} = (\mathcal{I}_1, \dots, \mathcal{I}_j)$ is not a prefix of $\vec{\mathcal{I}}^* = (\mathcal{I}^*_1, \dots, \mathcal{I}^*_\ell)$, we assume that $\mathcal{I}_j \neq \mathcal{I}^*_k$ for all $k \in \{1, \dots, \ell\}$. A simple scheme to achieve this encoding is to replace an arbitrary vector of component identities, $(\mathcal{I}_1, \dots, \mathcal{I}_j)$ by concatenating in each entry with all the previous entries: $(\mathcal{I}_1, \mathcal{I}_1||\mathcal{I}_2, \dots, \mathcal{I}_1||\mathcal{I}_2||\cdots||\mathcal{I}_j)$. 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.

3.1 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 $N = p_1p_2p_3$, 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$(M, (\mathcal{I}_1, \dots, \mathcal{I}_j), \mathrm{PP}), \rightarrow \mathrm{CT}$ The encryption algorithm chooses $s, t_1, \dots, t_j$ uniformly randomly from $\mathbb{Z}_N$. It creates the ciphertext as:

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

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

EncryptSF$(M, (\mathcal{I}_1, \dots, \mathcal{I}_j), \mathrm{PP}) \rightarrow \widetilde{\mathrm{CT}}$ 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$((\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}) \to \mathrm{SK}_{\vec{\mathcal{I}}}$ 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$((\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{MSK}, \mathrm{PP}) \to \widetilde{\mathrm{SK}}_{\vec{\mathcal{I}}}$ 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, $\mathrm{SK}_{\vec{\mathcal{I}}} = \{K'_{i,0}, K'_{i,1}, K'_{i,2}, K'_{i,3}\ \forall i\}$. 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$(\mathrm{PP}, \mathrm{SK}, \mathcal{I}_{j+1}) \to \mathrm{SK}'$ 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:

$$M = C/B.$$

3.2 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, $M = C/B$.

Relation to Attribute-Based Encryption We note that our HIBE construction can be reinterpreted as a Key-Policy Attribute-Based Encryption scheme, where the policies are restricted to be AND's of attributes. Identities now play the role of attributes, and the identity vector of the ciphertext can now be seen as a *set* of attributes. It is crucial to note that in the above construction, there is no mechanism of the construction requiring that the i^{th} level terms of the key be used with the i^{th} level terms of the ciphertext. This is instead imposed by the assumed properties of identity vectors. In our proof of security, we will leverage that the last level identity of a key which cannot decrypt will not match *any* component of the identity vector for the challenge ciphertext.

4 Security

We will 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 2 and 3, this implies that our HIBE system is secure. More formally, we will prove the following theorem:

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

Proving delegation invariance, semi-functional ciphertext invariance, and semi-functional security will be relatively straightforward. 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. We give the relatively simpler proofs first.

4.1 Delegation Invariance

We first prove that our dual system encryption HIBE scheme has delegation invariance.

Lemma 5. *Our system has delegation invariance.*

Proof. Since our delegation algorithm additively rerandomizes each of the random exponents $y_1, \ldots, y_j, r_1, \ldots, r_j, \lambda_1, \ldots, \lambda_j$ in a secret key, the distribution of a secret key obtained through any sequence of delegations is the same as the distribution of a secret key for the same identity vector generated by a direct call to KeyGen. Thus, for any PPT algorithm $\mathcal{A}$ in Game HIBE, we can define a PPT algorithm $\mathcal{A}'$ in Game HIBE$_{WD}$ that achieves exactly the same advantage. $\mathcal{A}'$ essentially runs the same as $\mathcal{A}$, except when $\mathcal{A}$ makes a Create or Delegate query, $\mathcal{A}'$ makes no query. When $\mathcal{A}$ makes a Reveal query, $\mathcal{A}'$ makes a KeyGen query for the same identity vector. Since the keys that $\mathcal{A}'$ receives come from the same distribution as the keys that $\mathcal{A}$ receives, their advantages are identical.

□

4.2 Semi-functional Ciphertext Invariance

We now prove that our dual system encryption HIBE scheme has semi-functional ciphertext invariance.

Lemma 6. *Under Assumption 1, our system has semi-functional ciphertext invariance.*

Proof. We assume there is a PPT attacker $\mathcal{A}$ such that $\mathcal{A}$ achieves a non-negligible difference in advantage between Game HIBE$_{WD}$ and Game HIBE$_C$. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 1 with non-negligible advantage. $\mathcal{B}$ is given $g \in G_{p_1}$ and T. $\mathcal{B}$ chooses a, b, c, d, α randomly from $\mathbb{Z}_N$. It gives the public parameters

$$\mathrm{PP} = \{N, G, g, u = g^a, h = g^b, v = g^c, w = g^d, e(g,g)^\alpha\}$$

to $\mathcal{A}$. Since $\mathcal{B}$ knows the master secret key α, it can respond to $\mathcal{A}$'s key requests by calling the key generation algorithm and giving $\mathcal{A}$ the resulting keys.

At some point, $\mathcal{A}$ provides two messages M_0, M_1 and requests the challenge ciphertext for some identity vector, denoted by $(\mathcal{I}_1^*, \ldots, \mathcal{I}_\ell^*)$. $\mathcal{B}$ forms the ciphertext as follows. It chooses $t_1, \ldots, t_\ell$ randomly from $\mathbb{Z}_N$ and β randomly from $\{0,1\}$ and sets:

$$C = M_\beta e(g,T)^\alpha, \ C_0 = T,$$

$$C_{i,1} = T^d v^{t_i}, \ C_{i,2} = g^{t_i}, \ C_{i,3} = (u^{\mathcal{I}_i^*} h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

This implicitly sets g^s equal to the G_{p_1} part of T. If $T \in G_{p_1}$, then this is a well-distributed normal ciphertext, and $\mathcal{B}$ has properly simulated Game HIBE$_{WD}$. If $T \in G_{p_1 p_2}$, then this is a well-distributed semi-functional ciphertext (since the value of d modulo p_2 is uncorrelated from its value modulo p_1 by the Chinese Remainder Theorem). Hence, $\mathcal{B}$ has properly simulated Game HIBE$_C$ in this case. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve an non-negligible advantage against Assumption 1. □

4.3 Semi-functional Security

We now prove that our dual system encryption HIBE scheme has semi-functional security.

Lemma 7. *Under Assumption 2, our system has semi-functional security.*

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ who achieves a non-negligible advantage in Game HIBE$_{SF}$. We will create a PPT algorithm $\mathcal{B}$ which has a non-negligible advantage against Assumption 2.

$\mathcal{B}$ receives $g, g_2, g_3, g^\alpha X_2, g^s Y_2, T$. It chooses a, b, c, d randomly from $\mathbb{Z}_N$ and sets the public parameters as:

$$\mathrm{PP} = \{N, G, g, u = g^a, h = g^b, v = g^c, w = g^d, e(g, g^\alpha X_2)\}.$$

It gives these to $\mathcal{A}$. We note that $\mathcal{B}$ does not know the master secret key α.

In response to a KeyGen query for $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, $\mathcal{B}$ will create a semi-functional key as follows. It chooses $r_1, \ldots, r_j \in \mathbb{Z}_N$ randomly, $y_1, \ldots, y_{j-1}, y_j' \in \mathbb{Z}_N$ randomly, and $\lambda_1, \ldots, \lambda_{j-1}, \lambda_j' \in \mathbb{Z}_N$ randomly up to the constraint that $\lambda_1 + \cdots + \lambda_{j-1} + \lambda_j' = 0$. It will implicitly set $\lambda_j = \alpha + \lambda_j'$ modulo p_1 and y_j equal to $\alpha + y_j'$ modulo p_1. It also chooses a random value $f \in \mathbb{Z}_N$. The semi-functional key is formed as:

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

$$K_{j,0} = (g^{\alpha} X_2)^{d+1} \cdot g^{\lambda'_j} \cdot w^{y'_j} \cdot (g_2 g_3)^{f(d+1)},\ K_{j,1} = (g^{\alpha} X_2) \cdot g^{y'_j} \cdot (g_2 g_3)^{f},$$

$$K_{j,2} = (g^{\alpha} X_2)^{c} \cdot v^{y'_j} \cdot (u^{\mathcal{I}_j} h)^{r_j} \cdot (g_2 g_3)^{fc},\ K_{j,3} = g^{r_j}.$$

This is a well-distributed semi-functional key with $\psi = d+1$ and $\sigma = c$ modulo p_2 and p_3, and $\tilde{y}_j$ is equal to f plus the discrete log of X_2 base g_2 modulo p_2 and equal to f modulo p_3. Notice that $\tilde{y}_j$ is freshly random modulo p_2 and p_3 for each key, while σ, ψ are the same for all keys, as specified by the KeyGenSF algorithm.

At some point, $\mathcal{A}$ provides $\mathcal{B}$ with two messages M_0, M_1 and a challenge identity vector, $(\mathcal{I}_1^*, \ldots, \mathcal{I}_\ell^*)$. $\mathcal{B}$ creates the challenge ciphertext as follows. It chooses $t_1, \ldots, t_\ell, \delta'$ randomly from $\mathbb{Z}_N$ and chooses β randomly from $\{0,1\}$. It sets:

$$C = M_\beta T,\ C_0 = g^s Y_2,$$

$$C_{i,1} = (g^s Y_2)^d \cdot v^{t_i} \cdot g_2^{\delta'},\ C_{i,2} = g^{t_i},\ C_{i,3} = (u^{\mathcal{I}_i} h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

If $T = e(g,g)^{\alpha s}$, this is a well-distributed semi-functional encryption of M_β with γ equal to the discrete log of Y_2 base g_2 and δ equal to d times this discrete log plus δ'. Notice that δ' randomizes this so that there is no correlation with d modulo p_2. Hence this is uncorrelated from the exponents modulo p_2 of the semi-functional keys. In this case, $\mathcal{B}$ has properly simulated Game HIBE$_{SF}$.

If T is a random element of G_T, then this is a semi-functional encryption of a random message, and hence the ciphertext contains no information about β. In this case, the advantage of $\mathcal{A}$ must be zero. Since we have assumed the advantage of $\mathcal{A}$ is non-negligible in Game HIBE$_{SF}$, $\mathcal{B}$ can use the output of $\mathcal{A}$ to obtain a non-negligible advantage against Assumption 2. □

4.4 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*[2]. 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 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$(M, (\mathcal{I}_1, \ldots, \mathcal{I}_j), \mathrm{PP}, \sigma) \rightarrow \widetilde{\mathrm{CT}}_E$ 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:

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

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

[2] 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.

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:

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

$$K_{j,0}:=K'_{j,0},\ K_{j,1}=K'_{j,1},\ K_{j,2}:=K'_{j,2}\cdot(g_2g_3)^{\tilde{r}_1},\ K_{j,3}:=K'_{j,3}\cdot(g_2g_3)^{\tilde{r}_2}.$$

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

4.4.1 Sequence of Games

We will 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.

We will 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. Our proofs for the transitions from Game HIBE'_0 to Game HIBE'_1 will rely on helpful lemmas about the computational indistinguishability of various oracles which dispense group elements, and these lemmas will be proven with additional inner hybrids over intermediate oracles. We present these lemmas in the next subsection. We organize our proof this way because it allows us to place most of the technical work in lemmas which can be directly reused in the ABE setting.

4.4.2 Oracle Lemmas

We now define four oracles which answer queries from an attacker by sampling various distributions of group elements from a composite order bilinear group. The definitions of these oracles are motivated by the following relationships with keys and ciphertexts in our HIBE scheme. The outputs of Oracle $\mathcal{O}_0$ will allow a simulator to produce semi-functional keys, a semi-functional ciphertext, and a normal challenge key. The outputs of Oracle $\mathcal{O}_1$ will allow a simulator to produce a semi-functional ciphertext, semi-functional keys, and an ephemeral semi-functional challenge key. The outputs of Oracle $\mathcal{O}_2$ will allow a simulator to produce semi-functional keys, an ephemeral semi-functional ciphertext, and an ephemeral semi-functional challenge key. Finally, the outputs of Oracle $\mathcal{O}_3$ will allow a simulator to produce semi-functional keys, a semi-functional ciphertext, and a semi-functional challenge key.

We will show that under our generalized subgroup decision assumptions, these oracles cannot be distinguished by an attacker. All oracles will be defined with respect to a bilinear group G of order $N = p_1p_2p_3$. All of our four oracles initially choose random elements $g, u, v, w \in G_{p_1}$, $g_2 \in G_{p_2}$, and $g_3 \in G_{p_3}$, as well as random exponents $\psi, \sigma, a', b', s, \delta, \gamma \in \mathbb{Z}_N$. They provide the attacker with a description of the group G, as well as the group elements

$$g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ w^y(g_2g_3)^{y\psi},\ g^y(g_2g_3)^y,\ v^y(g_2g_3)^{y\sigma}.$$

The attacker can then query the oracles in two ways. It can make one "challenge key-type" query for an identity $\mathcal{I} \in \mathbb{Z}_N$, which will return four group elements. It can make an arbitrary number of "ciphertext-type" queries for different identities $\mathcal{I}^* \in \mathbb{Z}_N$, and each will return three group elements. It can make these queries in any order - the only limitations being that there cannot be more than one challenge key-type query, and the identity of the challenge key-type query cannot equal the identity of any ciphertext-type query modulo p_3. (These are called challenge key-type and ciphertext-type queries because they essentially return the challenge key and ciphertext distributions of our HIBE system, respectively.)

Oracle $\mathcal{O}_0$ The first oracle, which we will denote by $\mathcal{O}_0$, responds to queries as follows. Upon receiving a challenge key-type query for $\mathcal{I} \in \mathbb{Z}_N$, it chooses $r, y' \in \mathbb{Z}_N$ randomly and returns the group elements

$$(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}}h)^r, g^r)$$

to the attacker. Upon receiving a ciphertext-type query for $\mathcal{I}^* \in \mathbb{Z}_N$, it chooses $t \in \mathbb{Z}_N$ randomly and returns the group elements

$$(w^s g_2^{\delta} v^t, g^t, (u^{\mathcal{I}^*}h)^t)$$

to the attacker.

Oracle $\mathcal{O}_1$ The next oracle, which we will denote by $\mathcal{O}_1$, responds to queries as follows. Upon receiving a challenge key-type query for $\mathcal{I} \in \mathbb{Z}_N$, it chooses $r, y' \in \mathbb{Z}_N$ randomly, and also chooses $X_2, Y_2 \in G_{p_2}$, $X_3, Y_3 \in G_{p_3}$ randomly. It returns the group elements

$$(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}}h)^r X_2X_3, g^r Y_2Y_3)$$

to the attacker. It responds to a ciphertext-type query in the same way as $\mathcal{O}_0$.

Oracle $\mathcal{O}_2$ The next oracle, which we will denote by $\mathcal{O}_2$, responds to queries as follows. Upon receiving a challenge key-type query, it responds in the same way as $\mathcal{O}_1$. Upon receiving a ciphertext-type query for $\mathcal{I}^* \in \mathbb{Z}_N$, it chooses $t \in \mathbb{Z}_N$ randomly and returns the group elements

$$(w^s g_2^{\delta} v^t g_2^{\sigma t}, g^t g_2^t, (u^{\mathcal{I}^*}h)^t g_2^{t(a'\mathcal{I}^*+b')})$$

to the attacker.

Oracle $\mathcal{O}_3$ The last oracle, which we will denote by $\mathcal{O}_3$, responds to ciphertext-type queries in the same way as $\mathcal{O}_0$, and responds to a challenge key-type query for $\mathcal{I} \in \mathbb{Z}_N$ by choosing a random $y', r \in \mathbb{Z}_N$ and returns the group elements

$$(w^{y'}(g_2g_3)^{y'\psi}, g^{y'}(g_2g_3)^{y'}, v^{y'}(g_2g_3)^{y'\sigma}(u^{\mathcal{I}}h)^r, g^r)$$

to the attacker.

We define the advantage of an attacker $\mathcal{A}$ in distinguishing between $\mathcal{O}_i$ and $\mathcal{O}_j$ to be:

$$|Pr[\mathcal{A}(\mathcal{O}_i) = 1] - Pr[\mathcal{A}(\mathcal{O}_j) = 1]|.$$

Here, we assume that $\mathcal{A}$ interacts with either $\mathcal{O}_i$ or $\mathcal{O}_j$, and then outputs a bit 0 or 1 encoding its guess of which oracle it interacted with. Relying on our generalized subgroup decision assumptions, we will show that the advantage of any PPT attacker in distinguishing between these oracles must be negligible. More precisely, we will prove the following lemmas:

Lemma 8. *Under Assumptions 3 and 4, no PPT attacker can distinguish between $\mathcal{O}_0$ and $\mathcal{O}_1$ with non-negligible advantage.*

Lemma 9. *Under Assumptions 3 and 4, no PPT attacker can distinguish between $\mathcal{O}_1$ and $\mathcal{O}_2$ with non-negligible advantage.*

Lemma 10. *Under Assumptions 3 and 4, no PPT attacker can distinguish between $\mathcal{O}_2$ and $\mathcal{O}_3$ with non-negligible advantage.*

We will prove these lemmas by going through several intermediary oracles. The main properties of our oracles are summarized in Table 1. The descriptions in the table are imprecise because they only specify what subgroups are present and do not specify the distributions. We will give complete definitions for each of our oracles as we encounter them, and we intend this table to be used only as a quick reference guide, not as a definition.

In the following proofs, we will often use the fact that if a value $c \in \mathbb{Z}_N$ is chosen uniformly at random, then values of c modulo p_1, p_2, and p_3 are also uniformly random in $\mathbb{Z}_{p_1}$, $\mathbb{Z}_{p_2}$, and $\mathbb{Z}_{p_3}$ respectively, and are independent from each other (this follows from the Chinese Remainder Theorem). We start by proving Lemma 8 in two steps. First, we define an intermediary oracle $\mathcal{O}_{1/2}$.

Oracle $\mathcal{O}_{1/2}$ This oracle initializes in the same way as $\mathcal{O}_0$, $\mathcal{O}_1$ and provides the attacker with initial group elements from the same distribution. $\mathcal{O}_{1/2}$ responds to ciphertext-type queries in the same way as oracles $\mathcal{O}_0, \mathcal{O}_1$. It responds to a challenge key-type query for $\mathcal{I} \in \mathbb{Z}_N$ by choosing a random $y', r \in \mathbb{Z}_N$, and also random $X_3, Y_3 \in G_{p_3}$. It returns the group elements

$$(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}}h)^r X_3, g^r Y_3)$$

to the attacker.

Lemma 11. *Under Assumption 3, no PPT attacker can distinguish between $\mathcal{O}_0$ and $\mathcal{O}_{1/2}$.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ such that $\mathcal{A}$ distinguishes $\mathcal{O}_0$ from $\mathcal{O}_{1/2}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which attains a non-negligible advantage against Assumption 3.

Table 1: Summary of Oracles

Oracle	**CT-Type Responses**	**CK-Type Response**
$\mathcal{O}_0$	G_{p_2} term on first element	all group elements in G_{p_1}
$\mathcal{O}_1$	same as $\mathcal{O}_0$	$G_{p_2p_3}$ terms on last 2 elements
$\mathcal{O}_2$	all elements in $G_{p_1p_2}$	same as $\mathcal{O}_1$
$\mathcal{O}_3$	same as $\mathcal{O}_0$	$G_{p_2p_3}$ terms on first 3 elements
$\mathcal{O}_{1/2}$	same as $\mathcal{O}_0$	G_{p_3} terms on last 2 elements
$\mathcal{O}_i^*$	first i like $\mathcal{O}_2$ rest like $\mathcal{O}_1$	same as $\mathcal{O}_1, \mathcal{O}_2$
$\mathcal{O}_i'$	like $\mathcal{O}_i^*$ except i^{th} has first term in G, rest in $G_{p_1p_3}$	same as $\mathcal{O}_1, \mathcal{O}_2$
$\mathcal{O}_i''$	like $\mathcal{O}^i$ except i^{th} has all terms in G	same as $\mathcal{O}_1, \mathcal{O}_2$
$\mathcal{O}_{2.1}$	same as $\mathcal{O}_2$	like $\mathcal{O}_2$ with extra G_{p_3} terms on first 2 elements
$\dagger\mathcal{O}_{2.2}$	all elements in G	same as $\mathcal{O}_{2.1}$
$\dagger\mathcal{O}_{2.3}$	same as $\mathcal{O}_{2.2}$	all terms in G
$\widetilde{\mathcal{O}}_i^*$	like $\mathcal{O}_i^*$	like $\mathcal{O}_{2.3}$
$\widetilde{\mathcal{O}}_i'$	like $\mathcal{O}_i'$	like $\mathcal{O}_{2.3}$
$\widetilde{\mathcal{O}}_i''$	like $\mathcal{O}_i''$	like $\mathcal{O}_{2.3}$
$\widetilde{\mathcal{O}}_{1/2}$	like $\mathcal{O}_3$	$G_{p_2p_3}$ terms on first 3 elements, G_{p_3} term on last element

Note: oracles marked with † initialize with an extra G_{p_3} term on $g^s g_2^{\gamma}$.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. $\mathcal{B}$ will simulate either $\mathcal{O}_0$ or $\mathcal{O}_{1/2}$ with $\mathcal{A}$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ picks values $a, b, c, d \in \mathbb{Z}_N$ uniformly at random and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It additionally chooses $s, \gamma, \delta, y_2 \in \mathbb{Z}_N$ randomly, and gives the attacker the group elements:

$$g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ (X_1X_3)^d g_2^{y_2 d},\ (X_1X_3) g_2^{y_2},\ (X_1X_3)^c g_2^{c y_2}.$$

We note that these are properly distributed, with y modulo p_1 implicitly set to the discrete logarithm of X_1 base g modulo p_1, ψ equal to d modulo p_2 and p_3, and σ equal to c modulo p_2 and p_3. Note that the values of c modulo p_1, p_2, p_3 are uncorrelated from each other by the Chinese Remainder Theorem, and $v = g^c$ only involves the value of c modulo p_1.

When $\mathcal{A}$ makes a ciphertext-type query for some identity $\mathcal{I}^*$, $\mathcal{B}$ responds by choosing a random $t \in \mathbb{Z}_N$ and returning $(w^s g_2^{\delta} v^t, g^t, (u^{\mathcal{I}^*} h)^t)$ to $\mathcal{A}$. When $\mathcal{A}$ makes its one challenge key-type query for some $\mathcal{I}$, $\mathcal{A}$ chooses a random $y' \in \mathbb{Z}_N$ and returns

$$(w^{y'}, g^{y'}, v^{y'} T^{a\mathcal{I}+b}, T)$$

to $\mathcal{A}$. This implicitly sets g^r to be the G_{p_1} part of T. If $T \in G_{p_1}$, then this matches the distribution of $\mathcal{O}_0$ (since there are no G_{p_3} terms here). If $T \in G_{p_1p_3}$, then this matches the distribution of $\mathcal{O}_{1/2}$ (note that a, b modulo p_2 are uniformly random and do not occur elsewhere

- so there are random G_{p_3} terms attached to the last two group elements). Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage against Assumption 3. □

Lemma 12. *Under Assumption 4, no PPT attacker can distinguish between $\mathcal{O}_{1/2}$ and $\mathcal{O}_1$.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ such that $\mathcal{A}$ distinguishes $\mathcal{O}_{1/2}$ from $\mathcal{O}_1$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which attains a non-negligible advantage against Assumption 4.

$\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. $\mathcal{B}$ will simulate either $\mathcal{O}_{1/2}$ or $\mathcal{O}_1$ with $\mathcal{A}$, depending on the value of T (which is either in $G_{p_1p_3}$ or G). $\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It chooses $y, \psi, \sigma \in \mathbb{Z}_N$ randomly and gives the attacker the group elements:

$$g,\ u,\ h,\ v,\ w,\ X_1X_2,\ w^y(Y_2Y_3)^{y\psi},\ g^y(Y_2Y_3)^y,\ v^y(Y_2Y_3)^{y\sigma}.$$

These are properly distributed with g^s implicitly set to be X_1.

When $\mathcal{A}$ makes a ciphertext-type query for some identity $\mathcal{I}^*$, $\mathcal{B}$ responds by choosing a random $t \in \mathbb{Z}_N$ and returning $((X_1X_2)^d v^t, g^t, (u^{\mathcal{I}^*}h)^t)$ to $\mathcal{A}$. (Note that this implicitly sets $g_2^\delta = X_2^d$, which is uniformly random because the value of d modulo p_2 does not occur elsewhere.) When $\mathcal{A}$ makes its challenge key-type query for some $\mathcal{I}$, $\mathcal{A}$ chooses a random $y' \in \mathbb{Z}_N$ and returns

$$(w^{y'}, g^{y'}, v^{y'}T^{a\mathcal{I}+b}, T)$$

to $\mathcal{A}$. As in the previous lemma, this implicitly sets g^r to be the G_{p_1} part of T. We note that a, b modulo p_2, p_3 are uniformly random and do not appear elsewhere. Thus, when $T \in G_{p_1p_3}$, these last two terms will have random elements of G_{p_3} attached (matching the distribution of $\mathcal{O}_{1/2}$), and when $T \in G$, these last two terms will have random elements in both G_{p_3} and G_{p_2} attached (matching the distribution of $\mathcal{O}_1$). Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage against Assumption 4. □

If some PPT attacker could distinguish between $\mathcal{O}_0$ and $\mathcal{O}_1$ with non-negligible probability, than it would also distinguish either between $\mathcal{O}_0$ and $\mathcal{O}_{1/2}$ or between $\mathcal{O}_{1/2}$ and $\mathcal{O}_1$ with non-negligible probability. This shows that Lemmas 11 and 12 imply Lemma 8.

We now prove Lemma 9 in a hybrid argument using polynomially many steps. We let q denote the number of ciphertext-type queries made by a PPT attacker $\mathcal{A}$. We will define additional oracles $\mathcal{O}_i^*$ for each i from 0 to q, $\mathcal{O}_i'$ for each i from 1 to q, and $\mathcal{O}_i''$ for each i from 1 to q.

Oracle $\mathcal{O}_i^*$ This oracle initializes in the same way as $\mathcal{O}_1$, $\mathcal{O}_2$ and provides the attacker with initial group elements from the same distribution. It also responds to challenge key-type queries in the same way as $\mathcal{O}_1$, $\mathcal{O}_2$. It keeps a counter of ciphertext-type queries which is initially equal to one. It increments this counter after each response to a ciphertext-type query. In response to the j^{th} ciphertext-type query for some $\mathcal{I}_j^*$, if $j \leq i$, it responds exactly like $\mathcal{O}_2$. If $j > i$, it responds exactly like $\mathcal{O}_1$. In particular, $\mathcal{O}_0^*$ is identical to $\mathcal{O}_1$ and $\mathcal{O}_q^*$ is identical to $\mathcal{O}_2$.

Oracle $\mathcal{O}_i'$ This oracle acts the same as $\mathcal{O}_i^*$ except in its response to the i^{th} ciphertext-type query. For the i^{th} ciphertext-type query, it chooses a random $t \in \mathbb{Z}_N$ and random elements $X_3, Y_3 \in G_{p_3}$ and responds with:

$$(w^s g_2^\delta v^t X_3^\sigma, g^t X_3, (u^{\mathcal{I}_j^*}h)^t Y_3).$$

Oracle $\mathcal{O}''_i$ This oracle acts the same as $\mathcal{O}'_i, \mathcal{O}^*_i$ except in its response to the i^{th} ciphertext-type query. For the i^{th} ciphertext-type query, it chooses a random $t \in \mathbb{Z}_N$ and random elements $X_3, Y_3 \in G_{p_3}$ and responds with:

$$\left(w^s g_2^\delta v^t g_2^{\sigma t} X_3^\sigma,\ g^t g_2^t X_3,\ (u^{\mathcal{I}^*_j} h)^t g_2^{t(a'\mathcal{I}^*_j + b')} Y_3\right).$$

We will transition from $\mathcal{O}_1 = \mathcal{O}^*_0$ to $\mathcal{O}'_1$, then to $\mathcal{O}''_1$, then to $\mathcal{O}^*_1$, $\mathcal{O}'_2$, and so on, until we arrive at $\mathcal{O}^*_q = \mathcal{O}_2$. We prove computational indistinguishability for each transition along the way in the following lemmas.

Lemma 13. *Under Assumption 3, no PPT attacker can distinguish between $\mathcal{O}^*_i$ and $\mathcal{O}'_{i+1}$ with non-negligible advantage for all $i = 0, \ldots, q-1$.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}^*_i$ and $\mathcal{O}'_{i+1}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which achieves non-negligible advantage against Assumption 3.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. $\mathcal{B}$ will simulate either $\mathcal{O}^*_i$ or $\mathcal{O}'_{i+1}$ with $\mathcal{A}$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $s, \gamma, \delta, y_2, a', b' \in \mathbb{Z}_N$ randomly and gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s g_2^\gamma,\ (X_1X_3)^d (g_2)^{y_2 d},\ (X_1X_3) g_2^{y_2},\ (X_1X_3)^c (g_2)^{y_2 c}\right).$$

These are properly distributed with y implicitly set to the discrete logarithm of X_1 base g modulo p_1, ψ equal to d modulo p_2, p_3, and $\sigma = c$ modulo p_2, p_3.

When $\mathcal{A}$ makes its challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ responds as follows. It chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$(w^{y'},\ g^{y'},\ v^{y'}(X_1X_3)^{r(a\mathcal{I}+b)} g_2^{r_1},\ (X_1X_3)^r g_2^{r_2}).$$

When $\mathcal{A}$ makes its j^{th} ciphertext-type query for $\mathcal{I}^*_j$ where $j \leq i$, $\mathcal{B}$ chooses $t \in Z_N$ randomly and responds with:

$$\left(w^s g_2^\delta v^t g_2^{ct},\ g^t g_2^t,\ (u^{\mathcal{I}^*_j} h)^t g_2^{t(a'\mathcal{I}^*_j + b')}\right).$$

This is identically distributed to a response from $\mathcal{O}_2$ (recall the $\sigma = c$ modulo p_2).

When $\mathcal{A}$ makes ciphertext type query $i+1$ for $\mathcal{I}^*_{i+1}$, $\mathcal{B}$ responds with:

$$\left(w^s g_2^\delta T^c,\ T,\ T^{a\mathcal{I}^*_{i+1} + b}\right).$$

When $\mathcal{A}$ makes its j^{th} ciphertext-type query for $\mathcal{I}^*_j$ where $j > i+1$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^\delta v^t,\ g^t,\ (u^{\mathcal{I}^*_j} h)^t\right).$$

This identically distributed to a response from $\mathcal{O}_1$.

We must now argue that the challenge key-type query and $i+1$ ciphertext-type query responses are properly distributed. If $T \in G_{p_1}$, then the response to the $i+1$ ciphertext-type query is identically distributed to a response from $\mathcal{O}_1$, and the values a, b modulo p_3 only appear in the response to the challenge key-type query, hence the G_{p_3} parts on the last two group elements here appear random in G_{p_3}. This means that the responses of $\mathcal{B}$ properly simulate the responses of $\mathcal{O}^*_i$. If $T \in G_{p_1p_3}$, then we must argue that $a\mathcal{I} + b$ and $a\mathcal{I}^*_{i+1} + b$ both appear to be uniformly random modulo p_3: this follows from pairwise independence of the function $a\mathcal{I} + b$ modulo p_3, since we have restricted the adversary to choose $\mathcal{I}$ and $\mathcal{I}^*_{i+1}$ so that $\mathcal{I} \neq \mathcal{I}^*_{i+1}$ modulo p_3. This means that the G_{p_3} components on the last two group elements of the challenge key-type query response and on the $i+1$ ciphertext-type query response are uniformly random in the attacker's view. Thus, $\mathcal{B}$ has properly simulated the responses of $\mathcal{O}'_{i+1}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve a non-negligible advantage against Assumption 3. □

Lemma 14. *Under Assumption 4, no PPT attacker can distinguish between $\mathcal{O}'{}_i$ and $\mathcal{O}''_i$ with non-negligible advantage for all $i = 1, \ldots, q$.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}'_i$ and $\mathcal{O}''_i$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which achieves non-negligible advantage against Assumption 4.

$\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. $\mathcal{B}$ will simulate either $\mathcal{O}'_i$ or $\mathcal{O}''_i$ with $\mathcal{A}$, depending on the value of T (which is either in $G_{p_1p_3}$ or G). $\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $\psi, y \in \mathbb{Z}_N$ randomly gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ X_1X_2,\ w^y(Y_2Y_3)^{\psi},\ g^y(Y_2Y_3),\ v^y(Y_2Y_3)^c\right).$$

These are properly distributed, with $g^s = X_1$ and $g_2^{\gamma} = X_2$. Note that this sets σ equal to c modulo p_2 and p_3.

When $\mathcal{A}$ makes it challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly, and responds with:

$$(w^{y'},\ g^{y'},\ v^{y'}(u^{\mathcal{I}}h)^r(Y_2Y_3)^{r_1},\ g^r(Y_2Y_3)^{r_2}).$$

This has uniformly random terms in G_{p_2} and G_{p_3} on the last two elements, since r_1, r_2 are both uniformly random modulo p_2 and p_3.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}^*_j$ where $j < i$, $\mathcal{B}$ chooses $t' \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_2)^d(X_1X_2)^{ct'},\ (X_1X_2)^{t'},\ (X_1X_2)^{'t(a\mathcal{I}^*_j+b)}\right).$$

This sets $X_2^d = g_2^{\delta}$, which is uniformly random because the value of d modulo p_2 will not appear elsewhere. It implicitly sets $g^t = X_1^{t'}$. This is identically distributed to a response from $\mathcal{O}_2$, with a', b' equal to a, b modulo p_2, and $\sigma = c$ modulo p_2. We note that this is in the only context in which the values of a, b modulo p_2 appear, so this is equivalent to choosing a', b' independently at random.

When $\mathcal{A}$ makes ciphertext-type query i for $\mathcal{I}^*_i$, $\mathcal{B}$ responds with:

$$\left((X_1X_2)^dT^c,\ T,\ T^{(a\mathcal{I}^*_i+b)}\right).$$

This implicitly sets $g^t = T$. We note that the G_{p_3} terms here are properly distributed, since the G_{p_3} part of T is random, and the values of a, b modulo p_3 do not appear elsewhere.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}^*_j$ where $j > i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_2)^dv^t,\ g^t,\ (u^{\mathcal{I}^*_j}h)^t\right).$$

This is identically distributed to a response from $\mathcal{O}_1$.

If $T \in G_{p_1p_3}$, then the response for ciphertext-type query i is identically distributed to a response from $\mathcal{O}'_i$. If $T \in G$, then this response additionally has terms in G_{p_2} which are appropriately distributed with $c = \sigma$, $a = a'$, $b = b'$ modulo p_2. Thus, the response is identically distributed to a response from $\mathcal{O}''_i$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve non-negligible advantage against Assumption 4. □

Lemma 15. *Under Assumption 3, no PPT attacker can distinguish between $\mathcal{O}''{}_i$ and $\mathcal{O}_i^*$ with non-negligible advantage for all $i = 1, \dots, q$.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}_i''$ and $\mathcal{O}_i^*$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which achieves non-negligible advantage against Assumption 3.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. $\mathcal{B}$ will simulate either $\mathcal{O}_i''$ or $\mathcal{O}_i^*$ with $\mathcal{A}$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $s, \gamma, \delta, y_2, a', b' \in \mathbb{Z}_N$ randomly and gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ (X_1X_3)^d (g_2)^{y_2 d},\ (X_1X_3) g_2^{y_2},\ (X_1X_3)^c (g_2)^{y_2 c}\right).$$

These are properly distributed with y implicitly set to the discrete logarithm of X_1 base g modulo p_1, ψ equal to d modulo p_2, p_3, and $\sigma = c$ modulo p_2, p_3.

When $\mathcal{A}$ makes a challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$(w^{y'},\ g^{y'},\ v^{y'}(X_1X_3)^{r(a\mathcal{I}+b)} g_2^{r_1},\ (X_1X_3)^r g_2^{r_2}).$$

We note that the G_{p_2} parts here are uniformly random.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j < i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t g_2^{ct},\ g^t g_2^t,\ (u^{\mathcal{I}_j^*} h)^t g_2^{t(a'\mathcal{I}_j^* + b')}\right).$$

This is identically distributed to a response from $\mathcal{O}_2$.

When $\mathcal{A}$ makes ciphertext-type query i for $\mathcal{I}_i^*$, $\mathcal{B}$ chooses $t_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} T^c \cdot g_2^{ct_2},\ T g_2^{t_2},\ T^{a\mathcal{I}_i^* + b} g_2^{t_2(a'\mathcal{I}_i^* + b')}\right).$$

We note that the G_{p_2} parts here are properly distributed, since $\sigma = c$ modulo p_2.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j > i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t,\ g^t,\ (u^{\mathcal{I}_j^*} h)^t\right).$$

This is identically distributed to a response from $\mathcal{O}_1$.

When $T \in G_{p_1}$, the values of a, b modulo p_3 only appear in the response to the challenge key-type query, which means that the G_{p_3} terms on the last two group elements there are uniformly random. Also, the response to the i^{th} ciphertext-type query is distributed exactly like a response from $\mathcal{O}_2$. In this case, $\mathcal{B}$ has properly simulated the responses of $\mathcal{O}_i^*$.

When $T \in G_{p_1p_3}$, we must argue that the values $a\mathcal{I} + b$ and $a\mathcal{I}_i^* + b$ appear uniformly random modulo p_3: this follows by pairwise independence of $a\mathcal{I} + b$ as a function of $\mathcal{I}$ modulo p_3, since $\mathcal{I} \neq \mathcal{I}_i^*$ modulo p_3 and a, b modulo p_3 only appear in these two values. Hence, $\mathcal{B}$ has properly simulated the response of $\mathcal{O}_i''$ in this case. We have thus shown that $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve non-negligible advantage against Assumption 3. □

Since $\mathcal{O}_0^* = \mathcal{O}_1$ and $\mathcal{O}_q^* = \mathcal{O}_2$, the existence of an attacker who can distinguish $\mathcal{O}_1$ from $\mathcal{O}_2$ with non-negligible probability would violate one of Lemmas 13, 14, 15 (since q must be polynomial). Thus, Lemmas 13, 14, and 15 imply Lemma 9.

Finally, we will prove Lemma 10, again using a hybrid argument over a polynomial number of steps. We must first define several new intermediary oracles.

Oracle $\mathcal{O}_{2.1}$ This oracle initializes in the same way as $\mathcal{O}_2$ and provides the attacker with initial group elements from the same distribution. It answers ciphertext-type queries in the same as $\mathcal{O}_2$. To answer a challenge key-type query for $\mathcal{I}$, it chooses $y', r \in \mathbb{Z}_N$ randomly, $X_2, Y_2 \in G_{p_2}$ randomly, and $X_3, Y_3 \in G_{p_3}$ randomly. It responds with:

$$\left(w^{y'} g_3^{y'\psi},\ g^{y'} g_3^{y'},\ v^{y'} (u^{\mathcal{I}} h)^r X_2 X_3,\ g^r Y_2 Y_3\right).$$

Notice that the G_{p_2} and G_{p_3} components on the last two elements are still random like for $\mathcal{O}_2$, but there are now additional G_{p_3} components on the first two terms sharing the value ψ modulo p_3 with the initial group elements given to the attacker.

Oracle $\mathcal{O}_{2.2}$ This oracle initializes a bit differently from the other oracles. It fixes random elements $g, u, h, v, w \in G_{p_1}$, $g_2 \in G_{p_2}$, and $g_3 \in G_{p_3}$. It chooses random exponents $s, \gamma, \delta, y, \psi, \sigma, a', b', t_3, t_3', t_3'' \in \mathbb{Z}_N$. It initially provides the attacker with the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s (g_2 g_3)^{\gamma},\ w^y (g_2 g_3)^{y\psi},\ g^y (g_2 g_3)^y, v^y (g_2 g_3)^{y\sigma}\right).$$

What differs from the previous oracles here is the added g_3^{γ} and term: notice that this is uniformly random in G_{p_3}, since γ is random modulo p_3 (and uncorrelated from its value modulo p_2). This oracle answers the challenge-key type query in the same way as $\mathcal{O}_{2.1}$. To answer a ciphertext-type query for $\mathcal{I}^*$, it chooses random values $t \in \mathbb{Z}_N$ and responds with:

$$\left(w^s g_2^{\delta} v^t g_2^{\sigma t} g_3^{t_3},\ g^t g_2^t g_3^{t_3'},\ (u^{\mathcal{I}^*} h)^t g_2^{t(a'\mathcal{I}^* + b')} g_3^{t_3''}\right).$$

It is crucial to note that these G_{p_3} terms are *the same* for each ciphertext-type query response.

Oracle $\mathcal{O}_{2.3}$ This oracle initializes in the same way as $\mathcal{O}_{2.2}$ and provides the attacker with the same initial elements as $\mathcal{O}_{2.2}$. It responds to ciphertext-type queries in the same was as $\mathcal{O}_{2.2}$. To answer a challenge key-type query for $\mathcal{I}$, it chooses $y, r \in \mathbb{Z}_N$ randomly, $X_2, Y_2 \in G_{p_2}$ randomly, and $X_3, Y_3 \in G_{p_3}$ randomly. It responds with:

$$\left(w^{y'} (g_2 g_3)^{y'\psi},\ g^{y'} (g_2 g_3)^{y'},\ v^{y'} (u^{\mathcal{I}} h)^r X_2 X_3,\ g^r Y_2 Y_3\right).$$

Once we have gone from $\mathcal{O}_2$ to $\mathcal{O}_{2.1}$, to $\mathcal{O}_{2.2}$, and then to $\mathcal{O}_{2.3}$, we have arrived at an oracle which is very similar to $\mathcal{O}_3$, except that it produces additional random terms in G_{p_2} and G_{p_3} for the challenge key-type query and ciphertext-type queries. Essentially, we can think of the challenge key-type response of $\mathcal{O}_{2.3}$ as having the same underlying structure as the response of $\mathcal{O}_3$, but with additional random components in the G_{p_2} and G_{p_3} subgroups on the last two group elements (terms from G_{p_2} and G_{p_3} do appear on the second to last group element in $\mathcal{O}_3$'s response, but these are not fully randomized as they are for $\mathcal{O}_{2.3}$). To get rid of the unwanted components here and on the ciphertext-type responses, we "unwind" the oracle transitions above, stripping off the excess terms and finally arriving at $\mathcal{O}_3$. This unwinding is accomplished by traversing through the following intermediary oracles.

Oracles $\widetilde{\mathcal{O}}_i^*$, $\widetilde{\mathcal{O}}_i'$, $\widetilde{\mathcal{O}}_i''$ These oracles are like $\mathcal{O}_i^*$, $\mathcal{O}_i'$ and $\mathcal{O}_i''$ respectively, except that the challenge key-type response for all of them is identically distributed to the response of $\mathcal{O}_{2.3}$ (meaning that there are G_{p_2}, G_{p_3} terms on the first two group elements which are distributed as $(g_2 g_3)^{y'\psi}, (g_2 g_3)^{y'}$).

Oracle $\widetilde{\mathcal{O}}_{1/2}$ This oracle is like $\mathcal{O}_3$, except that the challenge key-type response has additional random terms in G_{p_3} on its last two group elements.

We will transition from $\mathcal{O}_{2.3}$ to $\widetilde{\mathcal{O}}^*_q$ (this strips off the unwanted G_{p_3} terms in the ciphertext-type responses). We then move to $\widetilde{\mathcal{O}}''_q$, then to $\widetilde{\mathcal{O}}'_q$, then to Game $\widetilde{\mathcal{O}}^*_{q-1}$, and so on, until we arrive at $\widetilde{\mathcal{O}}^*_0$. Now all of the excess terms on the ciphertext-type responses are gone, and we are left with ciphertext-type responses that are identical to $\mathcal{O}_3$. Next, we transition to $\widetilde{\mathcal{O}}_{1/2}$ and finally to $\mathcal{O}_3$.

We now show that under our assumptions, no PPT attacker can detect any of the transitions between $\mathcal{O}_2$ and $\mathcal{O}_3$ with non-negligible advantage. This will prove Lemma 10.

Lemma 16. *Under Assumption 3, no PPT attacker can distinguish between $\mathcal{O}_2$ and $\mathcal{O}_{2.1}$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}_2$ and $\mathcal{O}_{2.1}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage.

$\mathcal{B}$ receives $\mathcal{B}$ receives g, g_2, X_1X_3, T. It will simulate either $\mathcal{O}_2$ or $\mathcal{O}_{2.1}$ with $\mathcal{A}$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ chooses $a, b, c, d \in \mathbb{Z}_N$ randomly, and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses random values $s, \gamma, \delta, y_2, a', b' \in \mathbb{Z}_N$ and gives the attacker the following group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s g_2^\gamma, (X_1X_3)^d g_2^{y_2 d},\ (X_1X_3)g_2^{y_2},\ (X_1X_3)^c g_2^{y_2 c}\right).$$

These are properly distributed, with ψ equal to d modulo p_2 and p_3 and σ equal to c modulo p_2 and p_3. This implicitly sets $g^y = X_1$.

When $\mathcal{A}$ makes its challenge key-type request for $\mathcal{I}$, $\mathcal{B}$ chooses $r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left(T^d,\ T,\ T^c(X_1X_3)^{r(a\mathcal{I}+b)} g_2^{r_1},\ (X_1X_3)^r g_2^{r_2}\right).$$

We note that the G_{p_3} terms on the last two elements are uniformly random, since the values of a and b modulo p_3 do not appear elsewhere.

When $\mathcal{A}$ makes a ciphertext-type request for $\mathcal{I}^*$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^\delta v^t g_2^{ct},\ g^t g_2^t,\ (u^{\mathcal{I}^*} h)^t g_2^{a'\mathcal{I}^*+b'}\right).$$

This is identically distributed to a response from $\mathcal{O}_2$.

Now, if $T \in G_{p_1}$, then the first two elements of the challenge key-type response are in G_{p_1}, and the last two elements have uniformly random components in G_{p_2} and G_{p_3}. In this case, $\mathcal{B}$ has properly simulated $\mathcal{O}_2$. If $T \in G_{p_1p_3}$, then the distribution of the G_{p_3} components on the first two group elements of the challenge key response match the value of $\psi = d$ modulo p_3, and $\mathcal{B}$ has properly simulated $\mathcal{O}_{2.1}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to attain non-negligible advantage against Assumption 3. □

Lemma 17. *Under Assumption 4, no PPT attacker can distinguish between $\mathcal{O}_{2.1}$ and $\mathcal{O}_{2.2}$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}_{2.1}$ and $\mathcal{O}_{2.2}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage. We will invoke the assumption with the roles of p_2 and p_3 reversed.

$\mathcal{B}$ receives $g, g_2, X_1X_3, Y_2Y_3, T$. It will simulate either $\mathcal{O}_{2.1}$ or $\mathcal{O}_{2.2}$, depending on the value of T (which is either in $G_{p_1p_2}$ or G). It chooses random values $a, b, c, d \in \mathbb{Z}_N$ and sets $u =$

$g^a, h = g^b, v = g^c, w = g^d$. It chooses random values $\sigma, y, t_3 \in \mathbb{Z}_N$ and gives the attacker the following group elements:

$$\Big(g,\ u,\ h,\ v,\ w,\ T,\ w^y(Y_2Y_3)^d,\ g^y(Y_2Y_3),\ v^{y'}(Y_2Y_3)^\sigma\Big).$$

We note that this sets $\psi = d$ modulo p_2 and p_3. It implicitly sets g^s to be the G_{p_1} part of T. If $T \in G_{p_1p_2}$, this is distributed identically to the initial elements provided by $\mathcal{O}_{2.1}$. If $T \in G$, this is distributed identically to the initial elements provided by $\mathcal{O}_{2.2}$.

When $\mathcal{A}$ makes its challenge key-type request for $\mathcal{I}$, $\mathcal{B}$ chooses random values $y', r, r_1, r_2 \in \mathbb{Z}_N$ and responds with:

$$\Big((X_1X_3)^{dy'},\ (X_1X_3)^{y'}, (X_1X_3)^{cy'}(u^{\mathcal{I}}h)^r(Y_2Y_3)^{r_1},\ g^r(Y_2Y_3)^{r_2}\Big).$$

This is identically distributed to a response from $\mathcal{O}_{2.1}$ or $\mathcal{O}_{2.2}$. We note that the random term $Y_3^{r_1}$ means that the value of c modulo p_3 is *not revealed* here - it remains completely hidden from the attacker's view.

When $\mathcal{A}$ makes a ciphertext-type request for $\mathcal{I}^*$, $\mathcal{B}$ chooses $t' \in \mathbb{Z}_N$ and responds with:

$$\Big(T^dT^{ct_3}v^{t'}g_2^{\sigma t'},\ T^{t_3}g^{t'}g_2^{t'},\ T^{t_3(a\mathcal{I}^*+b)}g^{t'(a\mathcal{I}^*+b)}g_2^{t'(a\mathcal{I}^*+b)}\Big).$$

This implicitly sets g^t equal to $g^{t'}$ times the G_{p_1} part of T^{t_3}. We let g_2^τ denote the G_{p_2} part of T. We argue that the G_{p_2} terms here are properly distributed as $g_2^\delta g_2^{\sigma t''}, g_2^{t''}, g_2^{t''(a'\mathcal{I}^*+b')}$, where $a = a'$, $b = b'$, $t'' = t' + \tau t_3$, and $\delta = \tau(d + ct_3 - \sigma t_3)$. Since the values of a, b modulo p_2 do not appear in any other context, a' and b' are random modulo p_2. Since t' is random modulo p_2, t'' is also random modulo p_2, and since the value of c modulo p_2 does not appear elsewhere, δ is also random modulo p_2. Hence, the G_{p_2} terms here are properly distributed.

If $T \in G_{p_1p_2}$, this $\mathcal{B}$ has properly simulated $\mathcal{O}_{2.1}$. If $T \in G$, then the G_{p_3} terms attached to the ciphertext-type response are uniformly random, since t_3 is uniformly random modulo p_3, the value of c modulo p_3 is uniformly random and is not revealed elsewhere, and the values of a, b modulo p_3 are also uniformly random and do not appear elsewhere. In this case, $\mathcal{B}$ has properly simulated $\mathcal{O}_{2.2}$. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to attain non-negligible advantage against Assumption 4. □

Lemma 18. *Under Assumption 4, no PPT attacker can distinguish between $\mathcal{O}_{2.2}$ and $\mathcal{O}_{2.3}$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}_{2.2}$ and $\mathcal{O}_{2.3}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage.

$\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. It will simulate either $\mathcal{O}_{2.2}$ or $\mathcal{O}_{2.3}$, depending on the value of T (which is either in $G_{p_1p_3}$ or G). It chooses random values $a, b, c, d \in \mathbb{Z}_N$ and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses random values $s, \gamma, \delta, y, t_3, t_3' \in \mathbb{Z}_N$ and gives the attacker the group elements:

$$\Big(g,\ u,\ h,\ v,\ w,\ g^s(Y_2Y_3)^\gamma,\ w^y(Y_2Y_3)^d,\ g^y(Y_2Y_3),\ v^y(Y_2Y_3)^c\Big).$$

We note that this is properly distributed and sets $\psi = d$ modulo p_2 and p_3 and $\sigma = c$ modulo p_2 and p_3.

When $\mathcal{A}$ makes its challenge key-type request for $\mathcal{I}$, $\mathcal{B}$ chooses random values $r, r_1, r_2 \in \mathbb{Z}_N$ and responds with:

$$\Big(T^d,\ T,\ T^c(u^{\mathcal{I}}h)^r(Y_2Y_3)^{r_1},\ g^r(Y_2Y_3)^{r_2}\Big).$$

This implicitly sets $g^{y'}$ equal to the G_{p_1} part of T. We note that this matches the value $d = \psi$ modulo p_2, and also modulo p_3 if T has a G_{p_3} component.

When $\mathcal{A}$ makes a ciphertext-type request for $\mathcal{I}^*$, $\mathcal{B}$ chooses a random value $t' \in \mathbb{Z}_N$ and responds with:

$$\left(w^s (Y_2Y_3)^{\delta} (X_1X_2)^{ct'},\ (X_1X_2)^{t'} g_3^{t_3},\ (X_1X_2)^{a\mathcal{I}^*+b} g_3^{t_3'}\right).$$

This implicitly sets $g^t = X_1^{t'}$. It also sets $a' = a$ and $b' = b$ modulo p_2, which are properly distributed because a, b modulo p_2 do not appear elsewhere.

If $T \in G_{p_1p_3}$, the challenge key-type response is identically distributed to a response from $\mathcal{O}_{2.2}$. If $T \in G$, then the challenge key-type response is identically distributed to a response from $\mathcal{O}_{2.3}$. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to obtain a non-negligible advantage against Assumption 4. □

We now go through our "unwinding steps" to get to $\mathcal{O}_3$. We let q denote the number of queries the attacker makes. We note that q must be polynomial.

Lemma 19. *Under Assumption 4, no PPT attacker can distinguish between $\mathcal{O}_{2.3}$ and $\widetilde{\mathcal{O}}_q^*$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\mathcal{O}_{2.3}$ and $\widetilde{\mathcal{O}}_q^*$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage. We will invoke the assumption with the roles of p_2 and p_3 reversed.

$\mathcal{B}$ receives $g, g_2, X_1X_3, Y_2Y_3, T$. It will simulate either $\mathcal{O}_{2.3}$ or $\widetilde{\mathcal{O}}_q^*$, depending on the value of T (which is either in $G_{p_1p_2}$ or G). It chooses random values $a, b, c, d \in \mathbb{Z}_N$ and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It chooses random values $\sigma, y, t_3 \in \mathbb{Z}_N$ and gives the attacker the following group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ T,\ w^y (Y_2Y_3)^d,\ g^y (Y_2Y_3),\ v^y (Y_2Y_3)^{\sigma}\right).$$

We note that this sets $\psi = d$ modulo p_2 and p_3. If $T \in G_{p_1p_2}$, then this matches the initial elements provided by $\widetilde{\mathcal{O}}_q^*$. If $T \in G$, then this matches the initial elements provided by $\mathcal{O}_{2.3}$.

When $\mathcal{A}$ makes its challenge key-type request for $\mathcal{I}$, $\mathcal{B}$ chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_3g_2)^{dy'},\ (X_1X_3g_2)^{y'},\ (X_1X_3g_2)^{cy'} (u^{\mathcal{I}}h)^r (Y_2Y_3)^{r_1},\ g^r (Y_2Y_3)^{r_2}\right).$$

When $\mathcal{A}$ makes a ciphertext-type request for $\mathcal{I}^*$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ and responds with:

$$\left(T^d T^{ct_3} v^t g_2^{\sigma t},\ T^{t_3} g^t g_2^t,\ T^{t_3(a\mathcal{I}^*+b)} g^{t(a\mathcal{I}^*+b)} g_2^{t(a\mathcal{I}^*+b)}\right).$$

We note that this is very similar to the way $\mathcal{B}$ behaves in the proof of Lemma 17. The only difference is the $g_2^{dy'}, g_2^{y'}, g_2^{cy'}$ terms which have been added to the challenge key. As in the proof of Lemma 17, we have that if $T \in G$, the G_{p_3} components of the challenge ciphertext are properly distributed as in a response from $\mathcal{O}_{2.3}$, since the value of c modulo p_3 is *not revealed* by the challenge key-type response (it is hidden by the random term $Y_3^{r_1}$). Also as in the proof of Lemma 17, we have that the G_{p_2} components of the ciphertext-type responses are properly distributed. Thus, if $T \in G_{p_1p_2}$, $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_q^*$, and when $T \in G$, $\mathcal{B}$ has properly simulated the responses of $\mathcal{O}_{2.3}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve non-negligible advantage against Assumption 4. □

Lemma 20. *Under Assumption 3, no PPT attacker can distinguish between $\widetilde{\mathcal{O}}_i^*$ and $\widetilde{\mathcal{O}}_i''$ with non-negligible advantage, for each i from 1 to q.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\widetilde{\mathcal{O}}_i^*$ and $\widetilde{\mathcal{O}}_i''$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. It will simulate either $\widetilde{\mathcal{O}}_i^*$ or $\widetilde{\mathcal{O}}_i''$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). We note that the responses of $\widetilde{\mathcal{O}}_i^*$ are exactly like the responses of $\mathcal{O}_i^*$, except for the challenge key-type response, and the responses of $\widetilde{\mathcal{O}}_i''$ are exactly like the responses of $\mathcal{O}_i''$, except for the challenge key-type response. So $\mathcal{B}$ will behave like the simulator in the proof of Lemma 15, except for the key-type response. We describe the full behavior of $\mathcal{B}$ here for completeness.

$\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $s, \gamma, \delta, y_2, a', b' \in \mathbb{Z}_N$ randomly and gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ (X_1X_3)^d (g_2)^{y_2 d},\ (X_1X_3) g_2^{y_2},\ (X_1X_3)^c (g_2)^{y_2 c}\right).$$

These are properly distributed with $\psi = d$ modulo p_2, p_3, and $\sigma = c$ modulo p_2, p_3.

When $\mathcal{A}$ makes a challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_3g_2)^{dy'},\ (X_1X_3g_2)^{y'},\ (X_1X_3g_2)^{cy'}(X_1X_3)^{r(a\mathcal{I}+b)} g_2^{r_1},\ (X_1X_3)^r g_2^{r_2}\right).$$

We note that the G_{p_2} parts here are uniformly random.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j < i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t g_2^{ct},\ g^t g_2^t,\ (u^{\mathcal{I}_j^*} h)^t g_2^{t(a'\mathcal{I}_j^*+b')}\right).$$

This is identically distributed to a response from $\mathcal{O}_2$.

When $\mathcal{A}$ makes ciphertext-type query i for $\mathcal{I}_i^*$, $\mathcal{B}$ chooses $t_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} T^c \cdot g_2^{ct_2},\ T g_2^{t_2},\ T^{a\mathcal{I}_i^*+b} g_2^{t_2(a'\mathcal{I}_i^*+b')}\right).$$

We note that the G_{p_2} parts here are properly distributed, since $\sigma = c$ modulo p_2.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j > i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t,\ g^t,\ (u^{\mathcal{I}_j^*} h)^t\right).$$

This is identically distributed to a response from $\mathcal{O}_1$.

Now, we note that $a\mathcal{I}_i^* + b$ and $a\mathcal{I} + b$ appear uniformly random modulo p_3 by pairwise independence, since $\mathcal{I} \neq \mathcal{I}_i^*$ modulo p_3. Thus, when $T \in G_{p_1p_3}$, $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_i''$. When $T \in G_{p_1}$, the i^{th} response to a ciphertext-type query does not contain any G_{p_3} components, and so $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_i^*$. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve non-negligible advantage against Assumption 3. □

Lemma 21. *Under Assumption 4, no PPT attacker can distinguish between $\widetilde{\mathcal{O}}_i''$ and $\widetilde{\mathcal{O}}_i'$ with non-negligible advantage, for each i from 1 to q.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ who distinguishes between $\widetilde{\mathcal{O}}_i''$ and $\widetilde{\mathcal{O}}_i'$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage.

$\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. $\mathcal{B}$ will simulate either $\widetilde{\mathcal{O}}_i''$ or $\widetilde{\mathcal{O}}_i'$ with $\mathcal{A}$, depending on the value of T (which is either in $G_{p_1p_3}$ or G). $\mathcal{B}$ operates very similarly to the simulator in the proof of Lemma 14, except for its challenge key-type response. We describe its full behavior below for completeness.

$\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $\psi, y \in \mathbb{Z}_N$ randomly gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ X_1X_2,\ w^y(Y_2Y_3)^{\psi},\ g^y(Y_2Y_3),\ v^y(Y_2Y_3)^c\right).$$

These are properly distributed, with $g^s = X_1$ and $g_2^{\gamma} = X_2$. Note that this sets σ equal to c modulo p_2 and p_3.

When $\mathcal{A}$ makes it challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly, and responds with:

$$(w^{y'}(Y_2Y_3)^{y'\psi},\ g^{y'}(Y_2Y_3)^{y'},\ v^{y'}(u^{\mathcal{I}}h)^r(Y_2Y_3)^{r_1},\ g^r(Y_2Y_3)^{r_2}).$$

This has uniformly random terms in G_{p_2} and G_{p_3} on the last two elements, and the G_{p_2}, G_{p_3} terms on the first two elements match the proper distribution in terms of ψ.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j < i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_2)^d(X_1X_2)^{ct},\ (X_1X_2)^t,\ (X_1X_2)^{t(a\mathcal{I}_j^*+b)}\right).$$

This sets $X_2^d = g_2^{\delta}$, which is uniformly random because the value of d modulo p_2 does not appear elsewhere. This is identically distributed to a response from $\mathcal{O}_2$, with a', b' equal to a, b modulo p_2, and $\sigma = c$ modulo p_2. We note that this is in the only context in which the values of a, b modulo p_2 appear, so this is equivalent to choosing a', b' independently at random.

When $\mathcal{A}$ makes ciphertext-type query i for $\mathcal{I}_i^*$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_2)^dT^c,\ T,\ T^{t(a\mathcal{I}_i^*+b)}\right).$$

We note that the G_{p_3} terms here are properly distributed, since the G_{p_3} part of T is random, and the values of a, b modulo p_3 do not appear elsewhere.

When $\mathcal{A}$ makes ciphertext-type query j for $\mathcal{I}_j^*$ where $j > i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_2)^dv^t,\ g^t,\ (u^{\mathcal{I}_j^*}h)^t\right).$$

This is identically distributed to a response from $\mathcal{O}_1$.

If $T \in G_{p_1p_3}$, then the response for ciphertext-type query i is identically distributed to a response from $\widetilde{\mathcal{O}}'_i$. If $T \in G$, then this response additionally has terms in G_{p_2} which are appropriately distributed with $c = \sigma$, $a = a'$, $b = b'$ modulo p_2. Thus, the response is identically distributed to a response from $\widetilde{\mathcal{O}}''_i$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve non-negligible advantage against Assumption 4. □

Lemma 22. *Under Assumption 3, no PPT attacker can distinguish between $\widetilde{\mathcal{O}}'_i$ and $\widetilde{\mathcal{O}}^*_{i-1}$ with non-negligible advantage, for each i from 1 to q.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which distinguishes between $\widetilde{\mathcal{O}}'_i$ and $\widetilde{\mathcal{O}}^*_{i-1}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. $\mathcal{B}$ will simulate either $\widetilde{\mathcal{O}}'_i$ or $\widetilde{\mathcal{O}}^*_i$ with $\mathcal{A}$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ acts very similarly to the simulator in the proof of Lemma 13, except for its challenge key-type response. We describe its full behavior below for completeness.

$\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It also chooses $s, \gamma, \delta, y_2, a', b' \in \mathbb{Z}_N$ randomly and gives $\mathcal{A}$ the group elements:

$$\left(g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ (X_1X_3)^d (g_2)^{y_2 d},\ (X_1X_3) g_2^{y_2},\ (X_1X_3)^c (g_2)^{y_2 c}\right).$$

These are properly distributed with ψ equal to d modulo p_2, p_3, and $\sigma = c$ modulo p_2, p_3.

When $\mathcal{A}$ makes its challenge key-type query for $\mathcal{I}$, $\mathcal{B}$ responds as follows. It chooses $y', r, r_1, r_2 \in \mathbb{Z}_N$ randomly and responds with:

$$\left((X_1X_3g_2)^{dy'},\ (X_1X_3g_2)^{y'},\ (X_1X_3g_2)^{cy'}(X_1X_3)^{r(a\mathcal{I}+b)} g_2^{r_1},\ (X_1X_3)^r g_2^{r_2}\right).$$

When $\mathcal{A}$ makes its j^{th} ciphertext-type query for $\mathcal{I}_j^*$ where $j < i$, $\mathcal{B}$ chooses $t \in Z_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t g_2^{ct},\ g_t g_2^t,\ (u^{\mathcal{I}_j^*} h)^t g_2^{t(a'\mathcal{I}_j^* + b')}\right).$$

This is identically distributed to a response from $\mathcal{O}_2$ (recall the $\sigma = c$ modulo p_2).

When $\mathcal{A}$ makes ciphertext type query i for $\mathcal{I}_i^*$, $\mathcal{B}$ responds with:

$$\left(w^s g_2^{\delta} T^c,\ T,\ T^{a\mathcal{I}_i^* + b}\right).$$

When $\mathcal{A}$ makes its j^{th} ciphertext-type query for $\mathcal{I}_j^*$ where $j > i$, $\mathcal{B}$ chooses $t \in \mathbb{Z}_N$ randomly and responds with:

$$\left(w^s g_2^{\delta} v^t,\ g^t,\ (u^{\mathcal{I}_j^*} h)^t\right).$$

This identically distributed to a response from $\mathcal{O}_1$.

We note that $a\mathcal{I} + b$ and $a\mathcal{I}_i^* + b$ both appear to be uniformly random modulo p_3: this follows from pairwise independence and that $\mathcal{I} \neq \mathcal{I}_i^*$ modulo p_3. If $T \in G_{p_1}$, then $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_{i-1}^*$. If $T \in G_{p_1p_3}$, then $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_i'$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to obtain non-negligible advantage against Assumption 3. □

Lemma 23. *Under Assumption 4, no PPT attacker can distinguish between $\widetilde{\mathcal{O}}_0^*$ and $\widetilde{\mathcal{O}}_{1/2}$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which distinguishes between $\widetilde{\mathcal{O}}_0^*$ and $\widetilde{\mathcal{O}}_{1/2}$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage.

$\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. It will simulate either $\widetilde{\mathcal{O}}_0^*$ or $\widetilde{\mathcal{O}}_{1/2}$, depending on the value of T (which is either in $G_{p_1p_3}$ or G). $\mathcal{B}$'s behavior will be the same as the simulator in the proof in Lemma 12, except for its challenge key-type response. We describe its full behavior below for completeness.

$\mathcal{B}$ chooses $a, b, c, d, \alpha \in \mathbb{Z}_N$ randomly and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It chooses $y, \psi, \sigma \in \mathbb{Z}_N$ randomly and gives the attacker the group elements:

$$g,\ u,\ h,\ v,\ w,\ X_1X_2,\ w^y (Y_2Y_3)^{y\psi},\ g^y (Y_2Y_3)^y,\ v^y (Y_2Y_3)^{y\sigma}.$$

These are properly distributed with g^s implicitly set to be X_1.

When $\mathcal{A}$ makes a ciphertext-type query for some identity $\mathcal{I}^*$, $\mathcal{B}$ responds by choosing a random $t \in \mathbb{Z}_N$ and returning $((X_1X_2)^d v^t, g^t, (u^{\mathcal{I}^*} h)^t)$ to $\mathcal{A}$. (Note that this implicitly sets $g_2^{\delta} = X_2^d$, which is uniformly random because the value of d modulo p_2 does not occur elsewhere.)

When $\mathcal{A}$ makes its challenge key-type query for some $\mathcal{I}$, $\mathcal{A}$ chooses a random $y' \in \mathbb{Z}_N$ and responds with:

$$\left(w^{y'}(Y_2Y_3)^{\psi y'}, g^{y'}(Y_2Y_3)^{y'}, v^{y'}(Y_2Y_3)^{\sigma y'}T^{a\mathcal{I}+b}, T\right).$$

This implicitly sets g^r to be the G_{p_1} part of T. We note that a, b modulo p_2, p_3 are uniformly random and do not appear elsewhere. Thus, when $T \in G_{p_1p_3}$, these last two terms will have random elements of G_{p_3} attached (matching the distribution of $\widetilde{\mathcal{O}}_{1/2}$, and when $T \in G$, these last two terms will have random elements in both G_{p_3} and G_{p_2} attached (matching the distribution of $\widetilde{\mathcal{O}}_0^*$). Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage against Assumption 4. □

Lemma 24. *Under Assumption 3, no PPT attacker can distinguish between $\widetilde{\mathcal{O}}_{1/2}$ and $\mathcal{O}_3$ with non-negligible advantage.*

Proof. We assume there exists a PPT attacker $\mathcal{A}$ which distinguishes between $\widetilde{\mathcal{O}}_{1/2}$ and $\mathcal{O}_3$ with non-negligible advantage. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage.

$\mathcal{B}$ receives g, g_2, X_1X_3, T. It will simulate either $\widetilde{\mathcal{O}}_{1/2}$ or $\mathcal{O}_3$, depending on the value of T (which is either in G_{p_1} or $G_{p_1p_3}$). $\mathcal{B}$ will behave much like the simulator in the proof of Lemma 11, except for its challenge key-type response. We describe its full behavior below for completeness.

$\mathcal{B}$ selects values $a, b, c, d \in \mathbb{Z}_N$ uniformly at random and sets $u = g^a, h = g^b, v = g^c, w = g^d$. It additionally chooses $s, \gamma, \delta, y_2 \in \mathbb{Z}_N$ randomly, and gives the attacker the group elements:

$$g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ (X_1X_3)^d g_2^{y_2 d},\ (X_1X_3)g_2^{y_2},\ (X_1X_3)^c g_2^{c y_2}.$$

We note that these are properly distributed, with ψ equal to d modulo p_2 and p_3, and σ equal to c modulo p_2 and p_3.

When $\mathcal{A}$ makes its challenge key-type query for $\mathcal{I}$, $\mathcal{A}$ chooses a random $y' \in \mathbb{Z}_N$ and responds with:

$$\left((X_1X_3g_2)^{dy'}, (X_1X_3g_2)^{y'}, (X_1X_3g_2)^{cy'}T^{a\mathcal{I}+b}, T\right).$$

This implicitly sets g^r to be the G_{p_1} part of T.

When $\mathcal{A}$ makes a ciphertext-type query for some identity $\mathcal{I}^*$, $\mathcal{B}$ responds by choosing a random $t \in \mathbb{Z}_N$ and returning $(w^s g_2^{\delta} v^t, g^t, (u^{\mathcal{I}^*}h)^t)$ to $\mathcal{A}$.

If $T \in G_{p_1p_3}$, the $\mathcal{B}$ has properly simulated the responses of $\widetilde{\mathcal{O}}_{1/2}$. If $T \in G_{p_1}$, then $\mathcal{B}$ has properly simulated the responses of $\mathcal{O}_3$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to gain a non-negligible advantage against Assumption 3. □

When we put this all together, we see that Lemmas 16-24 imply Lemma 10, since we successfully transitioned from $\mathcal{O}_2$ to $\mathcal{O}_3$ with a polynomial number of intermediary steps, each of which is computationally indistinguishable to the attacker. This concludes our proof that oracles $\mathcal{O}_0$, $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$ are computationally indistinguishable.

4.4.3 Proof of One Semi-Functional Key Invariance

With the aid of Lemmas 8, 9, 10, we now prove our dual system encryption HIBE scheme has one semi-functional key invariance. This completes our proof of its security. We employ the games outlined in Section 4.4.1.

Lemma 25. *Under Assumptions 3 and 4, for any PPT attacker $\mathcal{A}$, the difference in $\mathcal{A}$'s advantage between Game HIBE$_\beta$ and Game HIBE$'_\beta$ for $\beta = 0, 1$ is negligible.*

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ and a value of $\beta \in \{0,1\}$ such that $\mathcal{A}$'s advantage changes non-negligibly between Game HIBE$_\beta$ and Game HIBE$'_\beta$. We will either create a PPT algorithm $\mathcal{B}$ that breaks Assumption 3 with non-negligible advantage or a PPT algorithm $\mathcal{B}$ that breaks Assumption 4 with non-negligible advantage.

Because $\mathcal{A}$'s advantage differs non-negligibly, it must be that while playing Game HIBE$_\beta$, $\mathcal{A}$ produces two values $\mathcal{I}, \mathcal{I}' \in \mathbb{Z}_N$ which are unequal modulo N but are equal modulo p_3, with non-negligible probability. We let A denote the g.c.d. of $\mathcal{I} - \mathcal{I}'$ and N, and we let B denote N/A. We then have that p_3 divides A, and $B \neq 1$. We consider two possible cases: 1) p_1 divides B and 2) $A = p_1 p_3$, $B = p_2$. At least one of these cases must occur with non-negligible probability.

If case 1) occurs with non-negligible probability, we can create a $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage. $\mathcal{B}$ receives g, g_2, X_1X_3, T. It can use these terms to simulate Game HIBE$_\beta$ with $\mathcal{A}$ as follows. It chooses $\alpha, a, b, c, d \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

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

We note that $\mathcal{B}$ knows the master secret key α, so it can easily make normal keys. Since $\mathcal{B}$ also knows g_2, it can easily make semi-functional ciphertexts.

To make semi-functional keys, $\mathcal{B}$ uses X_1X_3 and g_2. More precisely, to make a semi-functional key for $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, $\mathcal{B}$ chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ up to the constraint that $\lambda_1 + \cdots + \lambda_j = \alpha$. It also chooses random values $y_1, \ldots, y_{j-1}, y'_j, r_1, \ldots, r_j \in \mathbb{Z}_N$. It forms the key as:

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

$$K_{j,0} = g^{\lambda_j}(X_1X_3g_2)^{dy'_j},\ K_{j,1} = (X_1X_3g_2)^{y'_j},\ K_{j,2} = (X_1X_3g_2)^{cy'_j}(u^{\mathcal{I}_j}h)^{r_j}, K_{j,3} = g^{r_j}.$$

This makes properly distributed semi-functional keys with $\psi = d$ modulo p_2, p_3 and $\sigma = c$ modulo p_2, p_3.

Now, if $\mathcal{A}$ fails to produce $\mathcal{I}, \mathcal{I}'$ such that $\gcd(\mathcal{I} - \mathcal{I}', N) = A$ is divisible by p_3 and p_1 divides $B = N/A$, then $\mathcal{B}$ guesses randomly. However, with non-negligible probability, $\mathcal{A}$ will produce such an $\mathcal{I}, \mathcal{I}'$. $\mathcal{B}$ can detect this by computing $A = \gcd(\mathcal{I} - \mathcal{I}', N)$ and $B = N/A$, checking that g^B is the identity element (this will occur only if p_1 divides B since g has order p_1 in G) and checking that $(X_1X_3)^B \neq 1$ (this confirms that p_3 does not divide B, hence it must divide A). When $\mathcal{B}$ detects this situation, it can test whether $T \in G_{p_1}$ or $T \in G_{p_1p_3}$ by testing if T^B is 1. If $T^B = 1$ holds, then $T \in G_{p_1}$. If $T^B \neq 1$, then $T \in G_{p_1p_3}$. Thus, $\mathcal{B}$ achieves non-negligible advantage in breaking Assumption 3.

If case 2) occurs with non-negligible probability, we can create a $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage. $\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. It can use these terms to simulate Game HIBE$_\beta$ with $\mathcal{A}$ as follows. It chooses $\alpha, a, b, c, d \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

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

We note that $\mathcal{B}$ knows the master secret key α, so it can easily make normal keys.

To make a semi-functional ciphertext for $(\mathcal{I}^*_1, \ldots, \mathcal{I}^*_\ell)$ and message M, $\mathcal{B}$ chooses random values $t_1, \ldots, t_\ell \in \mathbb{Z}_N$ and forms the ciphertext as:

$$C = Me(X_1X_2, g)^\alpha,\ C_0 = X_1X_2,$$

$$C_{i,1} = (X_1X_2)^d (X_1X_2)^{ct_i},\ C_{i,2} = (X_1X_2)^{t_i},\ C_{i,3} = (X_1X_2)^{a\mathcal{I}^*_i + b} \quad \forall i \in \{1, \ldots, \ell\}.$$

We note that this will set $\sigma = c$ modulo p_2.

$\mathcal{B}$ chooses a random value $\psi \in \mathbb{Z}_N$. To make a semi-functional key for $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, $\mathcal{B}$ chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ up to the constraint that $\lambda_1 + \cdots + \lambda_j = \alpha$. It also chooses random values $y_1, \ldots, y_j, r_1, \ldots, r_j \in \mathbb{Z}_N$. It forms the key as:

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

$$K_{j,0} = g^{\lambda_j} w^{y_j}(Y_2Y_3)^{\psi y_j},\ K_{j,1} = g^{y_j}(Y_2Y_3)^{y_j},\ K_{j,2} = v^{y_j}(Y_2Y_3)^{cy_j}(u^{\mathcal{I}_j}h)^{r_j}, K_{j,3} = g^{r_j}.$$

We note that the semi-functional ciphertext and keys are well-distributed, and share the common value of $\sigma = c$ modulo p_2 as required. We note that the G_{p_2} terms on the ciphertext are random because the value of d modulo p_2 does not appear elsewhere.

Now, if $\mathcal{A}$ fails to produce $\mathcal{I}, \mathcal{I}'$ such that $\gcd(\mathcal{I} - \mathcal{I}', N) = A$, where $A = p_1p_3$ and $B = p_2$, then $\mathcal{B}$ guesses randomly. However, with non-negligible probability, $\mathcal{A}$ will produce such an $\mathcal{I}, \mathcal{I}'$. $\mathcal{B}$ can detect this by computing A, B and testing that g^B and g_3^B are not the identity element (this confirms that $B = p_2$, since it demonstrates the p_1 and p_3 do not divide B). Now, $\mathcal{B}$ can learn whether T has a G_{p_2} component or not by testing if T^A is the identity element or not. If it is not, then T has a G_{p_2} component. Thus, $\mathcal{B}$ achieves non-negligible advantage in breaking Assumption 4. □

Lemma 26. *Under Assumptions 3 and 4, our dual system encryption HIBE scheme has one semi-functional key invariance.*

Proof. Relying on the above lemma, if this is false we may assume there exists a PPT attacker $\mathcal{A}$ which achieves a non-negligible difference advantage between Game HIBE$'_0$ and Game HIBE$'_1$. This means that $\mathcal{A}$ must also achieve a non-negligible difference in advantage between at least one of the following pairs of games: Game HIBE$'_0$ and Game EK, Game EK and Game EC, and Game EC and Game HIBE$'_1$. Given such an $\mathcal{A}$, we will create a PPT algorithm $\mathcal{B}$ which distinguishes one of the following pairs of oracles with non-negligible advantage: $\mathcal{O}_0$ and $\mathcal{O}_1$, $\mathcal{O}_1$ and $\mathcal{O}_2$, and $\mathcal{O}_2$ and $\mathcal{O}_3$. This violates one of Lemmas 8, 9, 10.

We assume $\mathcal{B}$ interacts with one of $\mathcal{O}_0$, $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$. $\mathcal{B}$ initially obtains the group elements

$$g,\ u,\ h,\ v,\ w,\ g^s g_2^\gamma,\ w^y(g_2g_3)^{y\psi},\ g^y(g_2g_3)^y,\ v^y(g_2g_3)^{y\sigma}$$

from its oracle. It chooses $\alpha \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

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

We note that $\mathcal{B}$ knows the master secret key α.

When $\mathcal{A}$ requests a normal key, $\mathcal{B}$ can responds by using the usual key generation algorithm, since it knows α. When $\mathcal{A}$ requests a semi-functional key for some identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, $\mathcal{B}$ creates one as follows. It chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ up to the constraint that $\lambda_1 + \cdots + \lambda_j = \alpha$. It also chooses random values $y_1, \ldots, y_{j-1}, y'_j, r_1, \ldots, r_j \in \mathbb{Z}_N$. It forms the key as:

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

$$K_{j,0} = g^{\lambda_j}\left(w^y(g_2g_3)^{y\psi}\right)^{y'_j},\ K_{j,1} = \left(g^y(g_2g_3)^y\right)^{y'_j},\ K_{j,2} = \left(v^y(g_2g_3)^{y\sigma}\right)^{y'_j}(u^{\mathcal{I}_j}h)^{r_j},\ K_{j,3} = g^{r_j}.$$

We note that this creates properly distributed semi-functional secret key with the values of ψ, σ fixed modulo p_2 and p_3, and the value of y_j implicitly set to yy'_j, which is rerandomized for each key.

When $\mathcal{A}$ requests the challenge key for some identity vector $(\mathcal{I}_1, \ldots, \mathcal{I}_j)$, $\mathcal{B}$ makes a challenge key-type query to the oracle with input value $\mathcal{I}_j$. It receives from its oracle four group elements, which will denote by (T_0, T_1, T_2, T_3). $\mathcal{B}$ chooses random values $\lambda_1, \ldots, \lambda_j \in \mathbb{Z}_N$ up to the constraint that $\lambda_1 + \cdots + \lambda_j = \alpha$, and also chooses random values $y_1, \ldots, y_{j-1}, r_1, \ldots, r_{j-1} \in \mathbb{Z}_N$. $\mathcal{B}$ forms the challenge key for $\mathcal{A}$ as:

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

$$K_{j,0} = g^{\lambda_j} T_0,\ K_{j,1} = T_1,\ K_{j,2} = T_2,\ K_{j,3} = T_3.$$

If $\mathcal{B}$ is interacting with $\mathcal{O}_0$, then (T_0, T_1, T_2, T_3) will be distributed as $(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}_j} h)^r, g^r)$ for $y', r \in \mathbb{Z}_N$ randomly chosen, and so this will be a properly distributed normal key. If $\mathcal{B}$ is interacting with $\mathcal{O}_1$ or $\mathcal{O}_2$, (T_0, T_1, T_2, T_3) will be distributed as $(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}_j} h)^r X_2 X_3, g^r Y_2 Y_3)$, where $y', r \in \mathbb{Z}_N$, $X_2, Y_2 \in G_{p_2}$, and $Y_2, Y_3 \in G_{p_3}$ are chosen randomly, and so this will be a properly distributed ephemeral semi-functional key. If $\mathcal{B}$ is interacting with $\mathcal{O}_3$, (T_0, T_1, T_2, T_3) will be distributed as $(w^{y'}(g_2 g_3)^{\psi y'}, g^{y'}(g_2 g_3)^{y'}, v^{y'}(g_2 g_3)^{\sigma y'}(u^{\mathcal{I}_j} h)^r, g^r)$, where $y', r \in \mathbb{Z}_N$ are randomly chosen, and so this will be a properly distributed semi-functional key.

When $\mathcal{A}$ requests the challenge ciphertext for messages M_0, M_1 and identity vector $(\mathcal{I}_1^*, \ldots, \mathcal{I}_\ell^*)$, $\mathcal{B}$ makes a ciphertext-type query to the oracle for each $\mathcal{I}_i^*$ (We recall the value $\mathcal{I}_j$ from the challenge key cannot be equal to any of these values $\mathcal{I}^*$ modulo p_3.) In response to each query for $\mathcal{I}_i^*$, $\mathcal{B}$ receives three group elements, which we denote by (T_1^i, T_2^i, T_3^i). $\mathcal{B}$ chooses $\beta \in \{0, 1\}$ randomly and forms the ciphertext as:

$$C = M_\beta e(g^s g_2^\gamma, g)^\alpha, C_0 = g^s g_2^\gamma,$$

$$C_{i,1} = T_1^i,\ C_{i,2} = T_2^i,\ C_{i,3} = T_3^i \quad \forall i \in \{1, \ldots, \ell\}.$$

If $\mathcal{B}$ is interacting with $\mathcal{O}_0$, $\mathcal{O}_1$, or $\mathcal{O}_3$, then each (T_1^i, T_2^i, T_3^i) will be distributed as

$$(w^s g_2^\delta v^{t_i}, g^{t_i}, (u^{\mathcal{I}_i^*} h)^{t_i}),$$

where $t_i \in \mathbb{Z}_N$ is randomly chosen, so this will be a properly distributed semi-functional ciphertext. If $\mathcal{B}$ is interacting with $\mathcal{O}_2$, then each (T_1^i, T_2^i, T_3^i) will be distributed as

$$(w^s g_2^\delta v^{t_i} g_2^{\sigma t_i}, g^{t_i} g_2^{t_i}, (u^{\mathcal{I}_i^*} h)^{t_i} g_2^{t_i(a'\mathcal{I}_i^* + b')}),$$

where $t_i \in \mathbb{Z}_N$ is randomly chosen, and $a', b' \in \mathbb{Z}_N$ are randomly chosen and do not vary with i. In this case, $\mathcal{B}$ has produced a properly distributed ephemeral semi-functional ciphertext.

Thus, if $\mathcal{B}$ is interacting with $\mathcal{O}_0$, then it has properly simulated Game HIBE_0'. If $\mathcal{B}$ is interacting with $\mathcal{O}_1$, then it has properly simulated Game EK. If $\mathcal{B}$ is interacting with $\mathcal{O}_2$, then it has properly simulated Game EC. And finally, if $\mathcal{B}$ is interacting with $\mathcal{O}_3$, then it has properly simulated Game HIBE_1'. Thus, since $\mathcal{A}$ must achieve a non-negligible difference of advantage between at least one of these pairs of games, $\mathcal{B}$ will be able to distinguish the corresponding pair of oracles with non-negligible advantage. This violates one of Lemmas 8, 9, 10. Hence, under Assumptions 3 and 4, our dual system encryption HIBE scheme has one semi-functional key invariance. □

This completes the proof of Theorem 4.

5 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 $\vec{\alpha}$ with first coordinate equal to α, and the shares are obtained by multiplying the rows of the LSSS matrix by the sharing vector $\vec{\alpha}$. A subset of rows is capable of reconstructing the shared secret if and only if their span includes the vector $(1, 0, \ldots, 0)$.

5.1 Construction

Setup$(\lambda) \rightarrow \mathrm{PP}, \mathrm{MSK}$ 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:

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

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

Encrypt$(M, S \subseteq U, \mathrm{PP}) \rightarrow \mathrm{CT}$ 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$(\mathrm{MSK}, \mathrm{PP}, (A, \rho)) \rightarrow \mathrm{SK}$ 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 $\vec{\alpha} \in \mathbb{Z}_N^m$ 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 $\lambda_x := A_x \cdot \vec{\alpha}$ denote the share associated with the row A_x of A. The secret key is formed as:

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

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

Decrypt$(\mathrm{SK}, \mathrm{CT}) \rightarrow M$ 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:

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

$$M = C/B.$$

5.2 Correctness

We observe that:

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

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

This shows that $M = C/B$.

5.3 Security

We will 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 Appendix A). 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 can be found in Appendix B.

Theorem 27. *Under Assumptions 1-4, our KP-ABE system is selectively secure.*

5.4 Delegation for KP-ABE

In general, we say that a KP-ABE system provides delegation capabilities if it has an additional algorithm, Delegate, which takes in the public parameters, a secret key for an access structure $\mathbb{A}$, and a target access structure $\mathbb{A}'$. It outputs a new secret key for $\mathbb{A}'$ when the access structures $\mathbb{A}$ and $\mathbb{A}'$ satisfy certain conditions. A necessary condition is for $\mathbb{A}$ to represent an access policy which is equivalent to or more restrictive than the policy represented by $\mathbb{A}'$. However, we may not allow delegation to every such $\mathbb{A}'$. A KP-ABE system with delegation for tree access structures was presented in [28]. We will enhance our KP-ABE construction above to include delegation capabilities for LSSS matrices. We first formally define security for KP-ABE systems with delegation.

5.4.1 Security Definition

Selective security for KP-ABE systems with delegation is defined in terms of the following game between a challenger and an attacker. We let U denote the universe of attributes. We will later refer to this as Game KP-ABE with delegation. We assume that the universe of attributes is known by the attacker in the initialization phase. (In our particular system, this corresponds to giving the attacker the value N before the other public parameters.)

Initialization The attacker chooses a set $S^* \subseteq U$ of attributes which it will attack, and gives this to the challenger.

Setup The challenger runs the Setup algorithm and gives the public parameters PP to the attacker. It also initializes a set $Z = \emptyset$.

Phase 1 In this phase, the attacker can make many queries of the following types:

- Create: In a Create query, the attacker gives the challenger an access structure $\mathbb{A}$. The challenger creates a key for this by calling the KeyGen algorithm, and adds the key to the set Z. It gives the attacker a reference to the key only (it does not give the challenger the key itself).
- Delegate: In a Delegate query, the attacker gives the challenger a reference to a key in Z and a target access structure $\mathbb{A}'$. If allowed by the delegation algorithm, the challenger runs Delegate to produce a key for $\mathbb{A}'$ from the referenced key in Z. It adds this new key to Z and gives the attacker a reference to it (again, it does not give the attacker the key).
- Reveal: In a Reveal query, the attacker gives the challenger a reference to a key in Z, and the challenger gives the referenced key to the attacker. The attacker is not allowed to make reveal queries on keys for access policies which are satisfied by S^*.

Challenge The attacker declares two equal length messages M_0 and M_1. The challenger flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β under S^*, producing CT. It gives CT to the attacker.

Phase 2 The attacker again makes Create, Delegate, and Reveal queries, subject to the same constraints as in Phase 1.

Guess The attacker outputs a guess β' for β.

The advantage of an attacker $\mathcal{A}$ in this game is defined to be $Adv_{\mathcal{A}}^{KP-ABE}(\lambda) = Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 28. *A key-policy attribute-based encryption system with delegation is selectively secure if all polynomial time attackers have at most a negligible advantage in the above security game.*

We note that adaptive security for KP-ABE systems with delegation is defined similarly, except that there is no initialization phase. In other words, the attacker does not have to provide the set S^* until the challenge phase when it requests the ciphertext. The attacker is still constrained to ask only to reveal keys with access policies not satisfied by S^* in both Phase 1 and Phase 2. So when the attacker declares S^*, it must be the case that all the keys revealed to the attacker in Phase 1 have access policies not satisfied by S^*.

In Appendix A, we define the selective security game for KP-ABE without delegation, which we call Game KP-ABE. As in the HIBE case, it is not necessary to prove security in the more elaborate definition if the delegation algorithm produces keys which are appropriately re-randomized. More formally, we will prove that our KP-ABE system with delegation has the following property:

Delegation Invariance We say a KP-ABE scheme with delegation Π_D = (Setup, Encrypt, KeyGen, Decrypt, Delegate) has *delegation invariance* if for any PPT algorithm $\mathcal{A}$, there exists a PPT algorithm $\mathcal{A}'$ such that the advantage of $\mathcal{A}$ in Game KP-ABE with delegation is negligibly close to the advantage of $\mathcal{A}'$ in Game KP-ABE against Π = (Setup, Encrypt, KeyGen, Decrypt).

It follows that if a KP-ABE system Π_D = (Setup, Encrypt, KeyGen, Decrypt, Delegate) has delegation invariance and the system Π = (Setup, Encrypt, KeyGen, Decrypt) without the delegation algorithm is proven selectively secure, then Π_D is also selectively secure in the sense of Definition 28. (The same holds for the adaptive versions of the security definitions.)

5.4.2 The Delegation Algorithm for our KP-ABE System

We now state the delegation algorithm for our KP-ABE construction. It allows delegation from an $n \times m$ LSSS matrix (A, ρ) to an $n' \times m'$ LSSS matrix (A', ρ') when the following two conditions are met. We let $e_j \in \mathbb{Z}_N^{m'}$ denote the vector of length m' with a 1 in the j^{th} coordinate and 0's elsewhere. For each row A_x of A, we let $A_x^{m'}$ denote the vector of length m' formed by taking the length m vector A_x and concatenating $m' - m$ 0's on the end.

1. $m \leq m'$
2. For each attribute $s_i \in U$ in the image of ρ', we have:

$$Span_{\mathbb{Z}_N}\{A'_x \ s.t. \ \rho'(x) = s_i\} \subseteq Span_{\mathbb{Z}_N}\{A_x^{m'} \ s.t. \ \rho(x) = s_i, e_{m+1}, \ldots, e_{m'}\}.$$

In other words, this requires the A' has more columns than A, and for each attribute, the span of the rows associated with that attribute in A includes the projection onto $\mathbb{Z}_N^m$ of the rows of A' associated with that attribute. It is easy to verify that if (A, ρ) and (A', ρ') satisfy these conditions, then the access policy represented by (A', ρ') is either equivalent to or more restrictive than the policy represented by (A, ρ). However, there are matrix pairs (A, ρ), (A', ρ') where (A', ρ') is equivalent or more restrictive which fail to satisfy these conditions. Nonetheless, we will later show "completeness" of our delegation mechanism in the sense that if (A', ρ') represents an access policy which is equivalent to or more restrictive than (A, ρ), then there exists an LSSS matrix (A'', ρ'') which represents the *same policy* as (A', ρ'), and the pair (A, ρ) and (A'', ρ'') do satisfy conditions 1 and 2 above.

Delegate(PP, SK, $(A', \rho')) \rightarrow$ SK$'$ If (A, ρ) and (A', ρ') satisfy conditions 1 and 2, then the delegation algorithm proceeds as follows. It chooses a random vector $\vec{\alpha}' \in \mathbb{Z}_N^{m'}$ with first coordinate equal to 0. We let $\vec{\alpha} \in \mathbb{Z}_N^m$ denote the vector sharing α in the key SK for (A, ρ). The vector sharing α for the delegated key will be implicitly set to $\vec{\alpha} + \vec{\alpha}'$ (where we think of $\vec{\alpha}$ as being concatenated with $m' - m$ 0's on the end). Note that this is distributed as a freshly chosen random vector of length m' with first coordinate equal to α. We let $\lambda'_x = A'_x \cdot \vec{\alpha}'$. The algorithm also chooses random values $r_x, y_x \in \mathbb{Z}_N$ for x from 1 to m'. For each row A'_x of x, we compute coefficients $\omega_1^x, \dots, \omega_n^x \in \mathbb{Z}_N$ such that ω_i^x is 0 whenever $\rho(i) \neq \rho(x)$, and $\omega_1^x A_1 + \cdots + \omega_n^x A_n$ is equal to the projection of A'_x onto $\mathbb{Z}_N^m$. We let SK $= \{K_{x,0}, K_{x,1}, K_{x,2}, K_{x,3} \; \forall x \in \{1, \dots, m\}\}$ denote the secret key for (A, ρ).

The key SK$'$ is formed as:

$$K'_{x,0} = g^{\lambda'_x} w^{y_x} \prod_{i=1}^{n} K_{i,0}^{\omega_i^x}, \; K'_{x,1} = g^{y_x} \prod_{i=1}^{n} K_{i,1}^{\omega_i^x},$$

$$K'_{x,2} = v^{y_x} (u^{\rho(x)} h)^{r_x} \prod_{i=1}^{n} K_{i,2}^{\omega_i^x}, \; K'_{x,3} = g^{r_x} \prod_{i=1}^{n} K_{i,3}^{\omega_i^x} \quad \forall x \in \{1, \dots, n'\}.$$

We note that this re-randomizes the sharing vector as well as all the random exponents r_x, y_x. Hence, this key has the *same distribution* as a key for (A', ρ') generated by calling KeyGen instead.

Lemma 29. *Our KP-ABE system with delegation has delegation invariance.*

Proof. Since the keys produced by the delegation algorithm have the same distribution as fresh keys generated from KeyGen, we can define $\mathcal{A}'$ to run like $\mathcal{A}$, except replacing all of $\mathcal{A}$'s queries with only queries to the key generation algorithm for the keys that $\mathcal{A}$ requests to be revealed. Since this produces the same distribution on the keys, $\mathcal{A}$'s advantage will be precisely the same as $\mathcal{A}$'s advantage. Hence, our system has delegation invariance. □

5.4.3 Completeness of our Delegation Conditions

We now show that if the LSSS matrix (A', ρ') represents an equivalent or more restrictive policy than (A, ρ), then there exists some LSSS matrix (A'', ρ') which represents the same access policy as (A', ρ'), and (A, ρ) can delegate to (A'', ρ'') under our conditions 1 and 2. The size of A'' will be polynomially related to the sizes of A, A', and (A'', ρ'') will be efficiently computable from (A, ρ) and (A', ρ'). Essentially, we can adapt the completeness argument for trees given in [28] to the LSSS matrix setting. In the tree or boolean formula setting, we note that if $\mathcal{T}$ is a boolean formula and $\mathcal{T}'$ is an equivalent or more restrictive boolean formula, then the formula $\mathcal{T} AND \mathcal{T}'$ is equivalent to $\mathcal{T}'$. In [28], they note that a key for $\mathcal{T}$ can always be used to form a key for $\mathcal{T} AND \mathcal{T}'$.

Similarly, we will form (A'', ρ'') so that it represents an AND of the policies of (A, ρ) and (A', ρ'), and hence it is equivalent to (A', ρ). We will start with a copy of A, and for each row, we will add a 1 in a new column and $m' - 1$ 0's following it. We will then place a copy of A' in these columns in new rows. Essentially, this will force a user to satisfy A' in order to use each row of A in reconstructing the secret.

As usual, we let A be $n \times m$ and A' be $n' \times m'$. We define A'' to be $n + nn' \times m + nm'$.[3] We define the rows of A'' as follows. The first n rows of A'' are equal to the rows of A in the first m coordinates, and then row i of A'' for i from 1 to n additionally has a 1 in entry $m + 1 + (i - 1)m'$ (it has 0's in all other coordinates). The next n' rows have a copy of A' in columns $m + 1$ through $m + m'$, the next n' rows after that have a copy of A' in columns $m + m' + 1$ through $m + 2m'$, and so on. Letting J_i denote an $n \times m'$ matrix with a 1 in the i^{th} row, first column entry and 0's elsewhere, we can illustrate this pictorially as:

$$A'' = \begin{pmatrix} A & J_1 & J_2 & J_3 & \dots & J_n \\ 0 & A' & 0 & 0 & \dots & 0 \\ 0 & 0 & A' & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & \dots & A' \end{pmatrix}$$

In the block matrix above, a 0 represents a block of 0's of the appropriate size. We define ρ'' on A'' in the natural way - i.e each row of A'' is associated with the attribute assigned to the row of A or A' which is embedded in it.

To see that (A'', ρ'') represents the same policy as A', we note that if a set of attributes satisfies (A', ρ'), it also satisfies (A, ρ). Thus, to reconstruct the secret from such a set of attributes using shares for (A'', ρ''), the user can reconstruct the secret in the copy of A, yielding a vector which is 1 in the first coordinate, and non-zero only in the coordinates $m+1+(i-1)m'$ for i from 1 to n. These non-zero coordinates can be canceled out by reconstructing the vector $e_{m+1+(i-1)m'}$ using the i^{th} copy of A (we let e_j denote the vector with a 1 in the j^{th} coordinate and 0's elsewhere.

To see that (A, ρ) can delegate to (A'', ρ'') under our conditions 1 and 2, we note that A'' clearly has more columns than A, and for each attribute, the span of the rows associated with that attribute in A includes the projection onto $\mathbb{Z}_N^m$ of the rows of A'' associated with that attribute, so condition 2 holds. (Note that if we consider the first mn coordinates of rows of A'' associated with some attribute, we get exactly the rows of A associated with that attribute.)

This shows that our KP-ABE system allows delegation between from any policy represented by a LSSS matrix to any other equivalent or weaker policy represented by a LSSS matrix.

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.

[3] One can probably find much smaller matrices A'' meeting our requirements in many cases, but we are only concerned with completeness here and we do not attempt to minimize the size of A''.

[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. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–479, 2010.

[34] G. Miklau and D. Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, 2003.

[35] T. Okamoto and K. Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214–231, 2009.

[36] T. Okamoto and K. Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.

[37] 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.

[38] 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.

[39] A. Sahai and B. Waters. Fuzzy identity based encryption. In *EUROCRYPT*, pages 457–473, 2005.

[40] A. Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[41] E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *Automata, Languages and Programming*, pages 560–578, 2008.

[42] N. Smart. Access control using pairing based cryptography. In *CT-RSA*, pages 111–121, 2003.

[43] B. Waters. Efficient identity-based ecnryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.

[44] B. Waters. Dual system encryption: realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.

[45] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, 2011.

A Background for ABE

We now formally define access structures, linear secret-sharing schemes, KP-ABE systems, and selective security for KP-ABE systems.

A.1 Access Structures

Definition 30. *(Access Structure [6]) Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An* access structure *(respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \backslash \{\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets*.*

In our setting, attributes will play the role of parties and we will use monotone access structures. We note that it is possible to (inefficiently) realize general access structures with our techniques by having the negation of an attribute represented as a separate attribute (so the total number of attributes will be doubled).

A.2 Linear Secret-Sharing Schemes

Our construction will employ linear secret-sharing schemes (LSSS). We use the following definition, adapted from [6].

Definition 31. *(Linear Secret-Sharing Schemes (LSSS)) A secret sharing scheme* Π *over a set of parties $\mathcal{P}$ is called* linear *(over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix A called the share-generating matrix for* Π*. The matrix A has ℓ rows and n columns. For all $i = 1, \ldots, \ell$, the i^{th} row of A is labeled by a party $\rho(i)$ (ρ is a function from $\{1, \ldots, \ell\}$ to $\mathcal{P}$). When we consider the column vector $v = (s, r_2, \ldots, r_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared and $r_2, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Av is the vector of ℓ shares of the secret s according to* Π*. The share $(Av)_i$ belongs to party $\rho(i)$.*

We note the *linear reconstruction* property: we let Π denote an LSSS for access structure $\mathbb{A}$. We let S denote an authorized set, and define $I \subseteq \{1, \ldots, \ell\}$ as $I = \{i | \rho(i) \in S\}$. Then there exist constants $\{\omega_i \in \mathbb{Z}_p\}_{i \in I}$ such that, for any valid shares $\{\lambda\}_i$ of a secret s according to Π, we have: $\sum_{i \in I} \omega_i \lambda_i = s$. These constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix A [6].

Boolean Formulas We note that access policies could alternatively be described in terms of monotonic boolean formulas. LSSS access structures are more general and can be derived from such representations: one can use standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix.

A.3 KP-ABE Definition

A Key-Policy Attribute-Based Encryption Scheme consists of the following algorithms:

Setup$(\lambda, U) \rightarrow \mathrm{PP}, \mathrm{MSK}$ The setup algorithm takes in the security parameter λ and the attribute universe description U. It outputs the public parameters PP and a master secret key MSK.

Encrypt$(M, \mathrm{PP}, S) \rightarrow \mathrm{CT}$ The encryption algorithm takes in a message M, the public parameters, and a set of attributes S. It outputs a ciphertext CT.

KeyGen$(\mathbb{A}, \mathrm{MSK}, \mathrm{PP}) \rightarrow \mathrm{SK}$ The key generation algorithm takes in an access structure $\mathbb{A}$, the master secret key MSK, and the public parameters PP. It outputs a secret key SK.

Decrypt$(\mathrm{CT}, \mathrm{SK}) \rightarrow M$ The decryption algorithm takes in a ciphertext encrypted under a set of attributes S and a secret key for an access structure $\mathbb{A}$. It will output the message M if S satisfies $\mathbb{A}$.

A.4 Selective Security for KP-ABE Systems

We define selective security for KP-ABE systems in terms of the following game between an attacker and a challenger. We let U denote the universe of attributes. We will later refer to this selective security game as Game KP-ABE. We assume that the universe of attributes is known by the attacker in the initialization phase. (In our particular system, this corresponds to giving the attacker the value N before the other public parameters.)

Initialization The attacker chooses a set $S^* \subseteq U$ of attributes which it will attack, and gives this to the challenger.

Setup The challenger runs the Setup algorithm and gives the public parameters PP to the attacker.

Phase 1 The attacker queries the challenger for private keys corresponding to access structures, under the restriction that the access structure of each key must not be satisfied by S^*.

Challenge The attacker declares two equal length messages M_0 and M_1. The challenger flips a random coin $\beta \in \{0,1\}$, and encrypts M_β under S^*, producing CT. It gives CT to the attacker.

Phase 2 The attacker queries the challenger for private keys corresponding to access structures, again with the restriction that the access structure of each key must not be satisfied by S^*.

Guess The attacker outputs a guess β' for β.

The advantage of an attacker $\mathcal{A}$ in this game is defined to be $Adv_{\mathcal{A}}^{KP-ABE}(\lambda) = Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 32. *A key-policy attribute-based encryption system is selectively secure if all polynomial time attackers have at most a negligible advantage in the above security game.*

We note that adaptive security for KP-ABE systems is defined similarly, except that there is no initialization phase. In other words, the attacker does not have to provide the set S^* until the challenge phase when it requests the ciphertext. The attacker is still constrained to ask only for keys with access policies not satisfied by S^* in both Phase 1 and Phase 2. So when the attacker declares S^*, it must be the case that all the keys requested by the attacker in Phase 1 have access policies not satisfied by S^*.

B Security Proof for Our KP-ABE System

We now prove that our KP-ABE system is selectively secure. As for our HIBE system, we begin by defining an abstraction of dual system encryption KP-ABE schemes and three security properties which will imply selective security. Since we are working in the selective security setting, the challenger knows the set S^* throughout the security game, and may refer to it in creating semi-functional keys. To incorporate this into our abstraction, we define the semi-functional key generation algorithm below to take in S^* as an additional input.

B.1 Dual System Encryption KP-ABE

A dual system encryption KP-ABE scheme consists of the following algorithms. We note that the algorithms EncryptSF and KeyGenSF need not run in polynomial time, since they will not be used in the normal operation of the system. They are only needed for the security proof.

Setup$(\lambda, U) \rightarrow \mathrm{PP}, \mathrm{MSK}$ The setup algorithm takes in the security parameter λ and the attribute universe description U. It outputs the public parameters PP and a master secret key MSK.

Encrypt$(M, \mathrm{PP}, S) \rightarrow \mathrm{CT}$ The encryption algorithm takes in a message M, the public parameters, and a set of attributes S. It outputs a ciphertext CT.

EncryptSF$(M, \mathrm{PP}, S) \rightarrow \widetilde{\mathrm{CT}}$ The semi-functional encryption algorithm takes in a message M, the public parameters, and a set of attributes S. It outputs a semi-functional ciphertext $\widetilde{\mathrm{CT}}$.

KeyGen$(\mathbb{A}, \mathrm{MSK}, \mathrm{PP}) \rightarrow \mathrm{SK}$ The key generation algorithm takes in an access structure $\mathbb{A}$, the master secret key MSK, and the public parameters PP. It outputs a secret key SK.

KeyGenSF$(\mathbb{A}, \mathrm{MSK}, \mathrm{PP}, S^*) \rightarrow \widetilde{\mathrm{SK}}$ The semi-functional key generation algorithm takes in an access structure $\mathbb{A}$, the master secret key MSK, the public parameters PP, and a set of attributes S^*. It outputs a semi-functional secret key $\widetilde{\mathrm{SK}}$.

Decrypt($\mathrm{CT}, \mathrm{SK}) \rightarrow M$ The decryption algorithm takes in a ciphertext encrypted under a set of attributes S and a secret key for an access structure $\mathbb{A}$. It will output the message M if S satisfies $\mathbb{A}$ and the key and ciphertext are *not both semi-functional.*

B.2 Selective Security Properties for Dual System Encryption KP-ABE

We now define three security properties for a dual system encryption KP-ABE scheme. We will show that a system which has these properties is selectively secure. To define these properties, we first define the following variations of the selective security game above.

We define Game KP-ABE$_C$ to be the same as Game KP-ABE above, except that the challenger will create a semi-functional ciphertext by calling EncryptSF in the challenge phase instead of calling Encrypt. We define Game KP-ABE$_{SF}$ to be the same as Game KP-ABE$_C$, except that the challenger responds to all key requests by calling KeyGenSF and inputting the set S^* initially provided by the attacker. In other words, in this game the attacker receives all semi-functional keys as well as a semi-functional ciphertext.

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

$$\left|Adv_{\mathcal{A}}^{KP-ABE}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_C}(\lambda)\right| = negl(\lambda).$$

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

$$\left|Adv_{\mathcal{A}}^{KP-ABE_C}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_{SF}}(\lambda)\right| = negl(\lambda).$$

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

$$Adv_{\mathcal{A}}^{KP-ABE_{SF}} = negl(\lambda).$$

Theorem 33. *If a dual system encryption KP-ABE scheme* Π_D *=(Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt) has semi-functional ciphertext invariance, semi-functional key invariance, and semi-functional security, then* Π *=(Setup, Encrypt, KeyGen, Decrypt) is a selectively secure KP-ABE scheme.*

Proof. This proof is analogous to the HIBE case, but we include it for completeness. We let $\mathcal{A}$ denote any PPT algorithm. We will show that $\mathcal{A}$'s advantage in the selective security game against Π is negligible under the assumption that Π_D =(Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt) has semi-functional ciphertext invariance, semi-functional key invariance, and semi-functional security. First, we note that in the original Game KP-ABE, there are no calls to the semi-functional algorithms. Hence, the advantage of $\mathcal{A}$ against Π in this game is the same as its advantage against Π_D. Therefore, it suffices to consider $\mathcal{A}$'s advantage against Π_D.

By semi-functional ciphertext invariance, we have that:

$$\left|Adv_{\mathcal{A}}^{KP-ABE}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_C}(\lambda)\right| = negl(\lambda).$$

By semi-functional key invariance, we have that:

$$\left|Adv_{\mathcal{A}}^{KP-ABE_C}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_{SF}}(\lambda)\right| = negl(\lambda).$$

By the triangle inequality, we conclude that:

$$\left| Adv_{\mathcal{A}}^{KP-ABE}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_{SF}}(\lambda) \right| = negl(\lambda).$$

By semi-functional security, we know that $Adv_{\mathcal{A}}^{KP-ABE_{SF}}(\lambda)$ is negligible, thus $Adv_{\mathcal{A}}^{KP-ABE}(\lambda)$ is negligible as well. This proves that Π is selectively secure. □

B.3 An Alternative Security Property

As in the HIBE case, semi-functional key invariance is typically the most challenging aspect of the security proof and is difficult to prove directly. This motivates the definition of *one semi-functional key invariance*, which is easier to prove and implies semi-functional key invariance through a hybrid argument. To define this alternative property, we first need to define an additional security game, Game KP-ABE$_b$, where b can take values 0 and 1. In this game, when the attacker requests a key, it specifies whether it wants a normal or semi-functional key. The challenger then calls KeyGen or KeyGenSF respectively to produce the key (note that KeyGenSF is always called with the set S^* as input, where the attacker declares S^* at the beginning of the game). At some point, the attacker specifies a *challenge key*. If the value of b is 0, then the challenger responds with a normal key. If the value of b is 1, then the challenger responds with a semi-functional key. The ciphertext provided to the attacker in this game is semi-functional.

One Semi-functional Key Invariance We say a dual system encryption KP-ABE scheme Π_D =(Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt) has *one semi-functional key invariance* if, for any PPT attacker $\mathcal{A}$, the advantage of $\mathcal{A}$ in Game KP-ABE$_0$ is negligibly close to its advantage in Game KP-ABE$_1$. We denote this by:

$$\left| Adv_{\mathcal{A}}^{KP-ABE_0}(\lambda) - Adv_{\mathcal{A}}^{KP-ABE_1}(\lambda) \right| = negl(\lambda).$$

Theorem 34. *If a dual system encryption KP-ABE scheme Π_D =(Setup, Encrypt, EncryptSF, KeyGen, KeyGenSF, Decrypt) has one semi-functional key invariance, then it also has semi-functional key invariance.*

Proof. This proof is also analogous to the HIBE case, but we include it for completeness. We suppose there exists a PPT attacker $\mathcal{A}$ which achieves a non-negligible difference in advantage between Game KP-ABE$_C$ and Game KP-ABE$_{SF}$. Then we will show there must exist a PPT algorithm $\mathcal{B}$ which has a non-negligible difference in advantage between Game KP-ABE$_0$ and Game KP-ABE$_1$, contradicting one semi-functional key invariance. We let q denote the number of key queries that $\mathcal{A}$ makes. For k from 0 to q, we define Game KP-ABESF$_k$ as follows: the attacker receives a semi-functional ciphertext, semi-functional keys in response to the first k key requests, and normal keys in response to the remaining key requests. We note that Game KP-ABESF$_0$ is Game KP-ABE$_C$ and that Game KP-ABESF$_q$ is Game KP-ABE$_{SF}$.

Since $\mathcal{A}$ has a non-negligible difference in advantage between Game KP-ABE$_C$ and Game KP-ABE$_{SF}$ and q is polynomial, there must exist some value of k from 0 to $q-1$ such that

$$\left| Adv_{\mathcal{A}}^{KP-ABESF_k}(\lambda) - Adv_{\mathcal{A}}^{KP-ABESF_{k+1}}(\lambda) \right|$$

is non-negligible. Now, $\mathcal{B}$ works as follows. When it receives the public parameters from its challenger, it forwards these to $\mathcal{A}$. For the first k key requests that $\mathcal{A}$ makes, $\mathcal{B}$ forwards these to its challenger as requests for semi-functional keys, and returns the resulting semi-functional keys to $\mathcal{A}$. For the $k+1$ key request from $\mathcal{A}$, $\mathcal{B}$ forwards this to its challenger as the challenge key, and returns the resulting key to $\mathcal{A}$. For the remaining key requests from $\mathcal{A}$, $\mathcal{B}$ forwards these to its challenger as requests for normal keys, and returns the resulting keys to $\mathcal{A}$. $\mathcal{B}$ outputs whatever $\mathcal{A}$ outputs. Now, if $\mathcal{B}$ is playing Game KP-ABE$_0$, then $\mathcal{A}$ is playing Game KP-ABESF$_k$. If $\mathcal{B}$ is playing Game KP-ABE$_1$, then $\mathcal{A}$ is playing Game KP-ABESF$_{k+1}$. Hence, since $\mathcal{A}$ has a non-negligible difference in advantage between these two games, $\mathcal{B}$ has a non-negligible difference in advantage between Game KP-ABE$_0$ and Game KP-ABE$_1$. □

B.4 Our Semi-functional Algorithms

We now define the semi-functional algorithms for our KP-ABE scheme, which make it into a dual system encryption KP-ABE scheme. The other algorithms are defined above in Subsection 5.1. We let g_2 denote a generator of the subgroup G_{p_2} and g_3 denote a generator of the subgroup G_{p_3}.

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

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

$$C_{s_i,1} = C'_{s_i,1} \cdot g_2^{\delta},\ C_{s_i,2} = C'_{s_i,2},\ C_{s_i,3} = C'_{s_i,3} \quad \forall s_i \in S.$$

We note that the additional term g_2^{δ} in each $C_{s_i,1}$ is the same for each value of i.

KeyGenSF$(\mathrm{MSK}, \mathrm{PP}, (A, \rho), S^*) \rightarrow \widetilde{\mathrm{SK}}$ The first time the semi-functional key generation algorithm is called, it chooses two random values $\psi, \sigma \in \mathbb{Z}_N$ which it stores and uses on all subsequent calls. Each time it is called, the semi-functional key generation algorithm first calls the normal key generation algorithm KeyGen to obtain a normal secret key, $\mathrm{SK} = \{K'_{x,0}, K'_{x,1}, K'_{x,2}, K'_{x,3}\ \forall x \in \{1, \ldots n\}\}$. For each x such that $\rho(x) \notin S$, it chooses a random value $\tilde{y}_x \in \mathbb{Z}_N$.

It forms the semi-functional key as:

$$K_{x,0} = K'_{x,0},\ K_{x,1} = K'_{x,1},\ K_{x,2} = K'_{x,2},\ K_{x,3} = K'_{x,3} \quad \forall x\ s.t.\ \rho(x) \in S^*,$$

$$K_{x,0} = K'_{x,0} \cdot (g_2 g_3)^{\psi \tilde{y}_x},\ K_{x,1} = K'_{x,1} \cdot (g_2 g_3)^{\tilde{y}_x},$$
$$K_{x,2} = K'_{x,2} \cdot (g_2 g_3)^{\sigma \tilde{y}_x},\ K_{x,3} = K'_{x,3} \quad \forall x\ s.t.\ \rho(x) \notin S^*.$$

B.5 Semi-functional Ciphertext Invariance

We now show that our dual system encryption KP-ABE scheme has semi-functional ciphertext invariance.

Lemma 35. *Under Assumption 1, our dual system encryption KP-ABE scheme has semi-functional ciphertext invariance.*

Proof. We assume there is a PPT attacker $\mathcal{A}$ such that $\mathcal{A}$ achieves a non-negligible difference in advantage between Game KP-ABE and Game KP-ABE$_c$. We will create a PPT algorithm $\mathcal{B}$ which breaks Assumption 1 with non-negligible advantage. $\mathcal{B}$ is given $g \in G_{p_1}$ and T. $\mathcal{B}$

receives the set S^* from $\mathcal{A}$, and then $\mathcal{B}$ chooses a, b, c, d, α randomly from $\mathbb{Z}_N$. It gives the public parameters

$$\mathrm{PP} = \{N, G, g, u = g^a, h = g^b, v = g^c, w = g^d, e(g, g)^\alpha\}$$

to $\mathcal{A}$. Since $\mathcal{B}$ knows the master secret key α, it can respond to $\mathcal{A}$'s key requests by calling the key generation algorithm and giving $\mathcal{A}$ the resulting keys.

At some point, $\mathcal{A}$ provides two messages M_0, M_1 and requests the challenge ciphertext for S^*. We let ℓ denote the size of S^*, and we let $s_1, \ldots, s_\ell$ denote the elements of S^*. $\mathcal{B}$ forms the ciphertext as follows. It chooses $t_1, \ldots, t_\ell$ randomly from $\mathbb{Z}_N$ and β randomly from $\{0, 1\}$ and sets:

$$C = M_\beta e(g, T)^\alpha, \; C_0 = T,$$
$$C_{i,1} = T^d v^{t_i}, \; C_{i,2} = g^{t_i}, \; C_{i,3} = (u^{s_i} h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

This implicitly sets g^s equal to the G_{p_1} part of T. If $T \in G_{p_1}$, then this is a well-distributed normal ciphertext, and $\mathcal{B}$ has properly simulated Game KP-ABE. If $T \in G_{p_1 p_2}$, then this is a well-distributed semi-functional ciphertext (since the value of d modulo p_2 is random), and $\mathcal{B}$ has properly simulated Game KP-ABE$_C$. Thus, $\mathcal{B}$ can use the output of $\mathcal{A}$ to achieve an non-negligible advantage against Assumption 1. □

B.6 Semi-functional Security

We now show that our dual system encryption KP-ABE scheme has semi-functional security.

Lemma 36. *Under Assumption 2, our dual system encryption KP-ABE scheme has semi-functional security.*

Proof. We suppose there exists a PPT attacker $\mathcal{A}$ who achieves a non-negligible advantage in Game KP-ABE$_{SF}$. We will create a PPT algorithm $\mathcal{B}$ which has a non-negligible advantage against Assumption 2.

$\mathcal{B}$ receives $g, g_2, g_3, g^\alpha X_2, g^s Y_2, T$. It also receives S^* from $\mathcal{A}$. It chooses a, b, c, d randomly from $\mathbb{Z}_N$ and sets the public parameters as:

$$\mathrm{PP} = \{N, G, g, u = g^a, h = g^b, v = g^c, w = g^d, e(g, g^\alpha X_2)\}.$$

It gives these to $\mathcal{A}$. We note that $\mathcal{B}$ does not know the master secret key α.

In response to a KeyGen query for an $n \times m$ LSSS matrix (A, ρ), $\mathcal{B}$ will create a semi-functional key as follows. It chooses random values $y'_1, \ldots, y'_n, r_1, \ldots, r_n \in \mathbb{Z}_N$, a random vector $\vec{\mu} \in \mathbb{Z}_N^m$ up to the constraint that the first coordinate is zero, and a vector $\vec{\nu}$ which is chosen uniformly at random from the set of vectors in $\mathbb{Z}_N^m$ which are orthogonal to all rows A_x of A where $\rho(x) \in S^*$ and have first entry equal to 1 (note that this set is non-empty because S^* does not satisfy the access matrix (A, ρ)). $\mathcal{B}$ will implicitly set $\vec{\alpha} = \alpha\vec{\nu} + \vec{\mu}$ (note that this is distributed as a uniformly random vector with first entry equal to α). It also chooses random values $f_x \in \mathbb{Z}_N$ for each x such that $\rho(x) \notin S^*$. It forms the semi-functional key as:

$$K_{x,0} = g^{A_x \cdot \vec{\mu}} w^{y'_x}, \; K_{x,1} = g^{y'_x}, \; K_{x,2} = v^{y'_x}(u^{\rho(x)} h)^{r_x}, \; K_{x,3} = g^{r_x} \quad \forall x \text{ } s.t. \text{ } \rho(x) \in S^*,$$

$$K_{x,0} = g^{A_x \cdot \vec{\mu}} (g^\alpha X_2)^{(d+1) A_x \cdot \vec{\nu}} \cdot w^{y'_x} \cdot (g_2 g_3)^{(d+1) f_x}, \; K_{x,1} = g^{y'_x} (g^\alpha X_2)^{A_x \cdot \vec{\nu}} (g_2 g_3)^{f_x},$$

$$K_{x,2} = v^{y'_x} (g^\alpha X_2)^{c A_x \cdot \vec{\nu}} (g_2 g_3)^{c f_x} (u^{\rho(x)} h)^{r_x}, \; K_{x,3} = g^{r_x} \quad \forall x \text{ } s.t. \text{ } \rho(x) \notin S^*.$$

This is a properly distributed semi-functional key with $\psi = d + 1$ modulo p_2, p_3, $\sigma = c$ modulo p_2, p_3, $y_x = y'_x$ modulo p_1 for all x s.t. $\rho(x) \in S^*$, $y_x = \alpha A_x \cdot \nu + y'_x$ modulo p_1 for all x s.t.

$\rho(x) \notin S^*$, $\tilde{y}_x$ equal to f_x modulo p_3, and $\tilde{y}_x$ modulo p_2 equal to f_x plus the discrete logarithm of X_2 base g_2 times $A_x \cdot \nu$.

At some point, $\mathcal{A}$ provides $\mathcal{B}$ with two messages M_0, M_1. We let ℓ denote the size of S^* and we let $s_1, \ldots, s_\ell$ denote the elements of S^*. $\mathcal{B}$ creates the challenge ciphertext as follows. It chooses $t_1, \ldots, t_\ell, \delta'$ randomly from $\mathbb{Z}_N$ and chooses β randomly from $\{0,1\}$. It sets:

$$C = M_\beta T, \ C_0 = g^s Y_2,$$

$$C_{i,1} = (g^s Y_2)^d \cdot v^{t_i} \cdot g_2^{\delta'}, \ C_{i,2} = g^{t_i}, \ C_{i,3} = (u^{s_i} h)^{t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

If $T = e(g,g)^{\alpha s}$, this is a well-distributed semi-functional encryption of M_β with γ equal to the discrete log of Y_2 base g_2 and $\delta =$ equal to d times this discrete log plus δ'. Notice that δ' randomizes this so that there is no correlation with d modulo p_2. Hence this is uncorrelated from the exponents modulo p_2 of the semi-functional keys. In this case, $\mathcal{B}$ has properly simulated Game KP-ABE$_{SF}$.

If T is a random element of G_T, then this is a semi-functional encryption of a random message, and hence the ciphertext contains no information about β. In this case, the advantage of $\mathcal{A}$ must be zero. Since the advantage of $\mathcal{A}$ is non-negligible in Game KP-ABE$_{SF}$, $\mathcal{B}$ can use the output of $\mathcal{A}$ to obtain a non-negligible advantage against Assumption 2. □

B.7 One Semi-functional Key Invariance

To prove one semi-functional key invariance for our dual system encryption KP-ABE scheme, we will proceed similarly to our proof of one semi-functional key invariance in our HIBE scheme. We begin by defining ephemeral semi-functional keys and ciphertexts, which will be similar to our HIBE proof, except that we will now need an additional hybrid over the rows A_x of the key, and hence ephemeral semi-functionality for keys will be defined more "locally" as occurring only on a certain row of the key at a time. We will let I denote the number of rows A_x in an access matrix (A, ρ) such that $\rho(x) \notin S^*$. The ephemeral semi-functional key generation algorithm will take in an additional input i, which is an integer allowed to take values from 0 to $I+1$. These algorithms will also take the values σ, ψ in as input - since they will share these values with regular semi-functional keys and ciphertexts.

EncryptESF$(M, \text{PP}, S, \sigma) \rightarrow \widetilde{\text{CT}}_E$ The ephemeral semi-functional encryption algorithm first calls the normal encryption algorithm, Encrypt, to obtain a normal ciphertext
CT $= \{C', C'_0, C'_{s_i,1}, C'_{s_i,2}, C'_{s_i,3} \ \forall s_i \in S\}$. It then chooses random values $\gamma, \delta, a', b', t_1, \ldots, t_\ell \in \mathbb{Z}_N$, where ℓ is the size of S. It forms the ephemeral semi-functional ciphertext $\widetilde{\text{CT}}_E$ as:

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

$$C_{i,1} := C'_{i,1} \cdot g_2^\gamma \cdot g_2^{\sigma t_i}, \ C_{i,2} := C'_{i,2} \cdot g_2^{t_i}, \ C_{i,3} := C'_{i,3} \cdot g_2^{(a' s_i + b') t_i} \quad \forall i \in \{1, \ldots, \ell\}.$$

KeyGenESF$(\text{MSK}, \text{PP}, (A, \rho), S^*, i, \sigma, \psi) \rightarrow \widetilde{\text{SK}}_E$ The ephemeral semi-functional key generation algorithm first calls the normal key generation algorithm KeyGen to obtain a normal secret key, SK $= \{K'_{x,0}, K'_{x,1}, K'_{x,2}, K'_{x,3} \ \forall x \in \{1, \ldots n\}\}$. It chooses additional random values $r_1, r_2 \in Z_N$. We let x_j be the index of the j^{th} row A_x of A such that $\rho(x) \notin S^*$. $\mathcal{B}$ chooses additional random values $\tilde{y}_j \in \mathbb{Z}_N$ for each $j < i$. The ephemeral semi-functional key $\widetilde{\text{SK}}_E$ is formed as:

$$K_{x,0} = K'_{x,0}, \ K_{x,1} = K'_{x,1}, \ K_{x,2} = K'_{x,2}. \ K_{x,3} = K'_{x,3} \quad \forall x \ s.t. \ \rho(x) \in S^*,$$

$$K_{x_j,0} = K'_{x_j,0}(g_2g_3)^{\psi\tilde{y}_j},\ K_{x_j,1} = K'_{x_j,1}(g_2g_3)^{\tilde{y}_j},$$

$$K_{x_j,2} = K'_{x_j,2}(g_2g_3)^{\sigma\tilde{y}_j}.\ K_{x_j,3} = K'_{x_j,3} \forall x_j\ s.t\ j < i$$

$$K_{x_i,0} = K'_{x_i,0},\ K_{x_i,1} = K'_{x_i,1},\ K_{x_i,2} = K'_{x_i,2}(g_2g_3)^{r_1},\ K_{x_i,3} = K'_{x_i,3}(g_2g_3)^{r_2}.$$

$$K_{x_j,0} = K'_{x_j,0},\ K_{x_j,1} = K'_{x_j,1},\ K_{x_j,2} = K'_{x_j,2}.\ K_{x_j,3} = K'_{x_j,3} \quad \forall x_j\ s.t.\ j > i.$$

We again note that the values ψ, σ are shared with the semi-functional keys and ciphertexts. The KeyGenESF algorithm with index i essentially produces a key where the first $i - 1$ rows of the matrix with attributes not in S^* are distributed as they would be in a semi-functional key, the i^{th} such row is distributed as ephemeral semi-functional, and the remaining rows are distributed as they would be in a normal key. Note that setting $i = 0$ will yield a normal key, and setting $i = I + 1$ will yield a semi-functional key.

We also extend our definition of the semi-functional key generation algorithm to take in an index i from 0 to I. It acts the same as the non-indexed semi-functional key generation algorithm for the first i rows with attributes not in S^*, and then leaves the rest of the rows normal. Note that if we set our index to I, we reproduce our non-indexed definition. More formally:

KeyGenSF($\text{MSK}, \text{PP}, (A, \rho), S^*, i) \rightarrow \widetilde{\text{SK}}$ The first time the semi-functional key generation algorithm is called, it chooses two random values $\psi, \sigma \in \mathbb{Z}_N$ which it stores and uses on all subsequent calls. Each time it is called, the semi-functional key generation algorithm first calls the normal key generation algorithm KeyGen to obtain a normal secret key, $\text{SK} = \{K'_{x,0}, K'_{x,1}, K'_{x,2}, K'_{x,3}\ \forall x \in \{1, \dots n\}\}$. We let x_j be the index of the j^{th} row A_x of A such that $\rho(x) \notin S^*$. $\mathcal{B}$ chooses additional random values $\tilde{y}_j \in \mathbb{Z}_N$ for each $j \leq i$.

It forms the semi-functional key as:

$$K_{x,0} = K'_{x,0},\ K_{x,1} = K'_{x,1},\ K_{x,2} = K'_{x,2},\ K_{x,3} = K'_{x,3} \quad \forall x\ s.t.\ \rho(x) \in S^*,$$

$$K_{x_j,0} = K'_{x_j,0} \cdot (g_2g_3)^{\psi\tilde{y}_j},\ K_{x_j,1} = K'_{x_j,1} \cdot (g_2g_3)^{\tilde{y}_j},$$

$$K_{x_j,2} = K'_{x_j,2} \cdot (g_2g_3)^{\sigma\tilde{y}_j},\ K_{x_j,3} = K'_{x_j,3} \quad \forall x_j\ s.t.\ j \leq i,$$

$$K_{x_j,0} = K'_{x_j,0},\ K_{x_j,1} = K'_{x_j,1},\ K_{x_j,2} = K'_{x_j,2}.\ K_{x_j,3} = K'_{x_j,3} \quad \forall x_j\ s.t.\ j > i.$$

In order to get from Game KP-ABE$_0$ to Game KP-ABE$_1$, we will step through the following intermediary games. In these games, the distributions of the challenge and ciphertext vary, while the distributions of the requested normal and semi-functional keys remain the same as in Games KP-ABE$_0$ and KP-ABE$_1$.

Game KP-ABE$'_0$ This game is exactly like Game KP-ABE$_0$, except with the added restriction that the attacker cannot produce an access matrix (A, ρ) for the challenge key such that $\rho(x) \notin S^*$ for some x, but $\rho(x)$ is equal to some element of S^* when both are reduced modulo p_3.

Game EK$_i$ In this game, the ciphertext is semi-functional, and the challenge key is now ephemeral semi-functional with index i. We retain the added modular restriction from the previous game.

Game EC$_i$ In this game, the ciphertext is ephemeral semi-functional and the challenge key is ephemeral semi-functional with index i. We retain the added modular restriction.

Game SF$_i$ In this game, the ciphertext is semi-functional, and the challenge key semi-functional with index i. We retain the added modular restriction.

Game KP-ABE$'_1$ This game is exactly like Game KP-ABE$_1$, except with the added modular restriction.

We will transition between these games in the following order. We begin with Game KP-ABE$_0$ and move to Game KP-ABE$'_0$. We then move to Game EK$_1$, then Game EC$_1$, then Game SF$_1$, then Game EK$_2$, Game EC$_2$, Game SF$_2$, and so on, until we arrive at Game SF$_I$, which is the same as Game KP-ABE$'_1$. Finally, we transition to Game KP-ABE$_1$. We will prove that through each of these (polynomially many) transitions, the advantage of an attacker $\mathcal{A}$ can only change negligibly. This will show that our system has one semi-functional key invariance.

Lemma 37. *Under Assumptions 3 and 4, for any PPT attacker $\mathcal{A}$, the difference in $\mathcal{A}$'s advantage between Game KP-ABE$_\beta$ and Game KP-ABE$'_\beta$ for $\beta = 0, 1$ is negligible.*

Proof. We proceed similarly as in the proof of Lemma 25. We suppose there exists a PPT attacker $\mathcal{A}$ and a value of $\beta \in \{0, 1\}$ such that $\mathcal{A}$'s advantage changes non-negligibly between Game KP-ABE$_\beta$ and Game KP-ABE$'_\beta$. We will either create a PPT algorithm $\mathcal{B}$ that breaks Assumption 3 with non-negligible advantage or a PPT algorithm $\mathcal{B}$ that breaks Assumption 4 with non-negligible advantage.

Because $\mathcal{A}$'s advantage differs non-negligibly, it must be that while playing Game KP-ABE$_\beta$, $\mathcal{A}$ produces two values $s^*, s' \in \mathbb{Z}_N$ which are unequal modulo N but are equal modulo p_3, with non-negligible probability. We let A denote the g.c.d. of $s^* - s'$ and N, and we let B denote N/A. We then have that p_3 divides A, and $B \neq 1$. We consider two possible cases: 1) p_1 divides B and 2) $A = p_1 p_3$, $B = p_2$. At least one of these cases must occur with non-negligible probability.

If case 1) occurs with non-negligible probability, we can create a $\mathcal{B}$ which breaks Assumption 3 with non-negligible advantage. $\mathcal{B}$ receives g, g_2, X_1X_3, T. It can use these terms to simulate Game KP-ABE$_\beta$ with $\mathcal{A}$ as follows. It chooses $\alpha, a, b, c, d \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

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

We note that $\mathcal{B}$ knows the master secret key α, so it can easily make normal keys. Since $\mathcal{B}$ also knows g_2, it can easily make semi-functional ciphertexts.

To make semi-functional keys, $\mathcal{B}$ uses X_1X_3 and g_2. More precisely, to make a semi-functional key for (A, ρ), $\mathcal{B}$ chooses random values $r_1, \ldots, r_n, y_1, \ldots, y_n \in \mathbb{Z}_N$, and a random vector $\vec{\alpha} \in \mathbb{Z}_N^m$ with first entry equal to α. For each row A_x of A, we let $\lambda_x = A_x \cdot \vec{\alpha}$. It forms the semi-functional key as:

$$K_{x,0} = g^{\lambda_x} w^{y_x},\ K_{x,1} = g^{y_x},\ K_{x,2} = v^{y_x} (u^{\rho(x)} h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x \ s.t.\ \rho(x) \in S^*,$$

$$K_{x,0} = g^{\lambda_x} (X_1 X_3 g_2)^{d y_x},\ K_{x,1} = (X_1 X_3 g_2)^{y_x},$$

$$K_{x,2} = (X_1 X_3 g_2)^{c y_x} (u^{\rho(x)} h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x \ s.t.\ \rho(x) \notin S^*.$$

This produces properly distributed semi-functional keys with $\psi = d$ modulo p_2, p_3 and $\sigma = c$ modulo p_2, p_3.

Now, if $\mathcal{A}$ fails to produce s^*, s' such that $\gcd(\mathcal{I}-\mathcal{I}', N) = A$ is divisible by p_3 and p_1 divides $B = N/A$, then $\mathcal{B}$ guesses randomly. However, with non-negligible probability, $\mathcal{A}$ will produce such an s^*, s'. $\mathcal{B}$ can detect this and use it to determine the nature of T in the same way as in the proof of Lemma 25. Thus, $\mathcal{B}$ achieves non-negligible advantage in breaking Assumption 3.

If case 2) occurs with non-negligible probability, we can create a $\mathcal{B}$ which breaks Assumption 4 with non-negligible advantage. $\mathcal{B}$ receives $g, g_3, X_1X_2, Y_2Y_3, T$. It can use these terms to simulate Game KP-ABE$_\beta$ with $\mathcal{A}$ as follows. It chooses $\alpha, a, b, c, d \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

$$\mathrm{PP} = \{N, G, g, u = g^a, h = g^b, v = g^c, w = g^d, e(g,g)^\alpha\}.$$

We note that $\mathcal{B}$ knows the master secret key α, so it can easily make normal keys.

To make a semi-functional ciphertext for $S^* = \{s_1, \ldots, s_\ell\}$ and message M, $\mathcal{B}$ chooses random values $t_1, \ldots, t_\ell \in \mathbb{Z}_N$ and forms the ciphertext as:

$$C = Me(X_1X_2, g)^\alpha,\ C_0 = X_1X_2,$$

$$C_{i,1} = (X_1X_2)^d(X_1X_2)^{ct_i},\ C_{i,2} = (X_1X_2)^{t_i},\ C_{i,3} = (X_1X_2)^{as_i+b} \quad \forall i \in \{1, \ldots, \ell\}.$$

We note that this will set $\sigma = c$ modulo p_2.

$\mathcal{B}$ chooses a random value $\psi \in \mathbb{Z}_N$. To make a semi-functional key for (A, ρ), $\mathcal{B}$ chooses random values $r_1, \ldots, r_n, y_1, \ldots, y_n \in \mathbb{Z}_N$ and a random vector $\vec{\alpha} \in \mathbb{Z}_N^m$ with first entry equal to α. It forms the key as:

$$K_{x,0} = g^{\lambda_x} w^{y_x},\ K_{x,1} = g^{y_x},\ K_{x,2} = v^{y_x}(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \in S^*,$$

$$K_{x,0} = g^{\lambda_x} w^{y_x}(Y_2Y_3)^{\psi y_x},\ K_{x,1} = g^{y_x}(Y_2Y_3)^{y_x},$$

$$K_{x,2} = v^{y_x}(Y_2Y_3)^{cy_x}(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \notin S^*.$$

We note that the semi-functional ciphertext and keys are well-distributed, and share the common value of $\sigma = c$ modulo p_2 as required (notice that the G_{p_2} terms on the ciphertext are random because the value of d modulo p_2 does not appear elsewhere).

Now, if $\mathcal{A}$ fails to produce s^*, s' such that $\gcd(s^* - s', N) = A$, where $A = p_1p_3$ and $B = p_2$, then $\mathcal{B}$ guesses randomly. However, with non-negligible probability, $\mathcal{A}$ will produce such an s^*, s'. $\mathcal{B}$ can detect this and exploit this in the same was as in the proof of Lemma 25. Thus, $\mathcal{B}$ achieves non-negligible advantage in breaking Assumption 4. □

Lemma 38. *Under Assumptions 3 and 4, our dual system encryption KP-ABE scheme has one semi-functional key invariance.*

Proof. We proceed similarly to the proof of Lemma 26. We suppose there exists a PPT attacker $\mathcal{A}$ who achieves a non-negligible difference in advantage between Game KP-ABE$_0$ and Game KP-ABE$_1$. By the above lemma, we may assume that $\mathcal{A}$ also achieve a non-negligible difference in advantage between Game KP-ABE$'_0$ (i.e. Game SF$_0$) and Game KP-ABE$'_1$ (i.e. Game SF$_I$, where I denote the number of rows in the challenge key's access matrix whose attributes are not in the attribute set S^* of the ciphertext).

Since our hybrid sequence of games between Game SF$_0$ and Game SF$_1$ has at most a polynomial number of steps, there must exist a value of $i \in \{1, \ldots, I\}$ such that $\mathcal{A}$ achieves a non-negligible advantage between one of the following pairs of games: Game SF$_{i-1}$ and Game EK$_i$, Game EK$_i$ and Game EC$_i$, Game EC$_i$ and Game SF$_i$. We will use $\mathcal{A}$ to create a PPT algorithm $\mathcal{B}$ which will be able to distinguish with non-negligible advantage between one of the following pairs of oracles from Section 4.4.2: $\mathcal{O}_0$ and $\mathcal{O}_1$, $\mathcal{O}_1$ and $\mathcal{O}_2$, $\mathcal{O}_2$ and $\mathcal{O}_3$. This will contradict one of Lemmas 8, 9, 10.

We assume $\mathcal{B}$ interacts with one of $\mathcal{O}_0$, $\mathcal{O}_1$, $\mathcal{O}_2$, and $\mathcal{O}_3$. $\mathcal{B}$ initially obtains the group elements

$$g,\ u,\ h,\ v,\ w,\ g^s g_2^{\gamma},\ w^y(g_2g_3)^{y\psi},\ g^y(g_2g_3)^y,\ v^y(g_2g_3)^{y\sigma}$$

from its oracle. It chooses $\alpha \in \mathbb{Z}_N$ randomly, and gives $\mathcal{A}$ the following public parameters:

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

We note that $\mathcal{B}$ knows the master secret key α.

When $\mathcal{A}$ requests a normal key, $\mathcal{B}$ can responds by using the usual key generation algorithm, since it knows α. When $\mathcal{A}$ requests a semi-functional key for some access matrix (A,ρ), $\mathcal{B}$ creates one as follows. It chooses random values $y'_1,\dots,y'_n,r_1,\dots,r_n \in \mathbb{Z}_N$ and a random vector $\vec{\alpha} \in \mathbb{Z}_N^m$ with first entry equal to α. For each row A_x of A, we let $\lambda_x = A_x \cdot \vec{\alpha}$. $\mathcal{B}$ forms the key as:

$$K_{x,0} = g^{\lambda_x} w^{y'_x},\ K_{x,1} = g^{y'_x},\ K_{x,2} = v^{y'_x}(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \in S^*,$$

$$K_{x,0} = g^{\lambda_x}\left(w^y(g_2g_3)^{y\psi}\right)^{y'_x},\ K_{x,1} = (g^y(g_2g_3)^y)^{y'_x},$$

$$K_{x,2} = (v^y(g_2g_3)^{y\sigma})^{y'_x}\,(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \notin S^*.$$

We note that this creates properly distributed semi-functional secret keys with the values of ψ, σ fixed modulo p_2 and p_3, and the value of y_x implicitly set to yy'_x for the rows such that $\rho(x) \notin S^*$.

When $\mathcal{A}$ requests the challenge key for some access matrix (A,ρ), $\mathcal{B}$ makes a challenge key-type query to the oracle with input value $\rho(x_i) \in \mathbb{Z}_N$, where $x_i \in \{1,\dots,n\}$ is the index of the i^{th} row A_x in A such that $\rho(x) \notin S^*$. $\mathcal{B}$ receives from its oracle four group elements in response, which we will denote by (T_0,T_1,T_2,T_3). $\mathcal{B}$ chooses random values $r_j, y'_j \in \mathbb{Z}_N$, for all $j \in \{1,\dots,n\}$ such that $j \neq x_i$. It also chooses a random vector $\vec{\alpha} \in \mathbb{Z}_N^m$ with first entry equal to α, and we set $\lambda_x = A_x \cdot \vec{\alpha}$. $\mathcal{B}$ forms the challenge key as:

$$K_{x,0} = g^{\lambda_x} w^{y'_x},\ K_{x,1} = g^{y'_x},\ K_{x,2} = v^{y'_x}(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \in S^*,$$

$$K_{x_i,0} = g^{\lambda_{x_i}} T_0,\ K_{x_i,1} = T_1,\ K_{x_i,2} = T_2,\ K_{x_i,3} = T_3,$$

$$K_{x,0} = g^{\lambda_x}\left(w^y(g_2g_3)^{y\psi}\right)^{y'_x},\ K_{x,1} = (g^y(g_2g_3)^y)^{y'_x},$$

$$K_{x,2} = (v^y(g_2g_3)^{y\sigma})^{y'_x}\,(u^{\rho(x)}h)^{r_x},\ K_{x,3} = g^{r_x} \quad \forall x\ s.t.\ \rho(x) \notin S^* \wedge x \neq x_i.$$

If $\mathcal{B}$ is interacting with $\mathcal{O}_0$, then (T_0,T_1,T_2,T_3) will be distributed as $(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}_j}h)^r, g^r)$ for $y', r \in \mathbb{Z}_N$ randomly chosen, and so this will be a properly distributed normal key. If $\mathcal{B}$ is interacting with $\mathcal{O}_1$ or $\mathcal{O}_2$, (T_0,T_1,T_2,T_3) will be distributed as $(w^{y'}, g^{y'}, v^{y'}(u^{\mathcal{I}_j}h)^r X_2X_3, g^r Y_2Y_3)$, where $y', r \in \mathbb{Z}_N$, $X_2, Y_2 \in G_{p_2}$, and $Y_2, Y_3 \in G_{p_3}$ are chosen randomly, and so this will be a properly distributed ephemeral semi-functional key. If $\mathcal{B}$ is interacting with $\mathcal{O}_3$, (T_0,T_1,T_2,T_3) will be distributed as $(w^{y'}(g_2g_3)^{\psi y'}, g^{y'}(g_2g_3)^{y'}, v^{y'}(g_2g_3)^{\sigma y'}(u^{\mathcal{I}_j}h)^r, g^r)$, where $y', r \in \mathbb{Z}_N$ are randomly chosen, and so this will be a properly distributed semi-functional key.

When $\mathcal{A}$ requests the challenge ciphertext for messages M_0, M_1 and $S^* = \{s_1,\dots,s_\ell\}$, $\mathcal{B}$ makes a ciphertext-type query to the oracle for each s_j (We recall the value $\rho(x_i)$ from the challenge key cannot be equal to any of these values s_j modulo p_3.) In response to each query for s_j, $\mathcal{B}$ receives three group elements, which we denote by (T_1^j, T_2^j, T_3^j). $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly and forms the ciphertext as:

$$C = M_\beta e(g^s g_2^{\gamma}, g)^{\alpha}, C_0 = g^s g_2^{\gamma},$$

$$C_{j,1} = T_1^j,\ C_{j,2} = T_2^j,\ C_{j,3} = T_3^j \quad \forall j \in \{1, \ldots, \ell\}.$$

If $\mathcal{B}$ is interacting with $\mathcal{O}_0$, $\mathcal{O}_1$, or $\mathcal{O}_3$, then each (T_1^j, T_2^j, T_3^j) will be distributed as

$$(w^s g_2^\delta v^{t_j}, g^{t_j}, (u^{s_j} h)^{t_j}),$$

where $t_j \in \mathbb{Z}_N$ is randomly chosen, so this will be a properly distributed semi-functional ciphertext. If $\mathcal{B}$ is interacting with $\mathcal{O}_2$, then each (T_1^j, T_2^j, T_3^j) will be distributed as

$$(w^s g_2^\delta v^{t_j} g_2^{\sigma t_j}, g^{t_j} g_2^{t_j}, (u^{s_j} h)^{t_j} g_2^{t_j(a' s_j + b')}),$$

where $t_j \in \mathbb{Z}_N$ is randomly chosen, and $a', b' \in \mathbb{Z}_N$ are randomly chosen and do not vary with j. In this case, $\mathcal{B}$ has produced a properly distributed ephemeral semi-functional ciphertext.

Thus, if $\mathcal{B}$ is interacting with $\mathcal{O}_0$, then it has properly simulated Game SF$_{i-1}$. If $\mathcal{B}$ is interacting with $\mathcal{O}_1$, then it has properly simulated Game EK$_i$. If $\mathcal{B}$ is interacting with $\mathcal{O}_2$, then it has properly simulated Game EC$_i$. And finally, if $\mathcal{B}$ is interacting with $\mathcal{O}_3$, then it has properly simulated Game SF$_i$. Thus, since $\mathcal{A}$ must achieve a non-negligible difference of advantage between at least one of these pairs of games, $\mathcal{B}$ will be able to distinguish the corresponding pair of oracles with non-negligible advantage. This violates one of Lemmas 8, 9, 10. Hence, under Assumptions 3 and 4, our dual system encryption KP-ABE scheme has one semi-functional key invariance. □

This concludes our proof of Theorem 27.