Tools for Simulating Features of Composite Order Bilinear Groups in the Prime Order Setting

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

Abstract

In this paper, we explore a general methodology for converting composite order pairing-based cryptosystems into the prime order setting. We employ the dual pairing vector space approach initiated by Okamoto and Takashima and formulate versatile tools in this framework that can be used to translate composite order schemes for which the prior techniques of Freeman were insufficient. Our techniques are typically applicable for composite order schemes relying on the canceling property and proven secure from variants of the subgroup decision assumption, and will result in prime order schemes that are proven secure from the decisional linear assumption. As an instructive example, we obtain a translation of the Lewko-Waters composite order IBE scheme. This provides a close analog of the Boneh-Boyen IBE scheme that is proven fully secure from the decisional linear assumption. We also provide a translation of the Lewko-Waters unbounded HIBE scheme.

1 Introduction

Recently, several cryptosystems have been constructed in composite order bilinear groups and proven secure from instances (and close variants) of the general subgroup decision assumption defined in [3]. For example, the systems presented in [27, 25, 29, 28, 26] provide diverse and advanced functionalities like identity-based encryption (IBE), hierarchical identity-based encryption (HIBE), and attribute-based encryption with strong security guarantees (e.g. full security, leakage-resilience) proven from static assumptions. These works leverage convenient features of composite order bilinear groups that are not shared by prime order bilinear groups, most notably the presence of orthogonal subgroups of coprime orders. Up to isomorphism, a composite order bilinear group has the structure of a direct product of prime order subgroups, so every group element can be decomposed as the product of components in the separate subgroups. However, when the group order is hard to factor, such a decomposition is hard to compute. The orthogonality of these subgroups means that they can function as independent spaces, allowing a system designer to use them in different ways without any cross interactions between them destroying correctness. Security relies on the assumption that these subgroups are essentially inseparable: given a random group element, it should be hard to decide which subgroups contribute non-trivial components to it.

Though composite order bilinear groups have appealing features, it is desirable to obtain the same functionalities and strong guarantees achieved in composite order groups from other assumptions, particularly from the decisional linear assumption (DLIN) in prime order bilinear groups. The ability to work with prime order bilinear groups instead of composite order ones

*Supported by a Microsoft Research PhD Fellowship.

offers several advantages. First, we can obtain security under the more standard decisional linear assumption. Second, we can achieve much more efficient systems for the same security levels. This is because in composite order groups, security typically relies on the hardness of factoring the group order. This requires the use of large group orders, which results in considerably slower pairing operations.

There have been many previous examples of cryptosystems that were first built in composite order groups while later analogs were obtained in prime order groups. These include Groth-Ostrovsky-Sahai proofs [22, 21], the Boneh-Sahai-Waters traitor tracing scheme [10, 15], and the functional encryption schemes of Lewko-Okamoto-Sahai-Takashima-Waters [25, 33]. Waters also notes that the dual system encryption techniques in [39] used to obtain prime order systems were first instantiated in composite order groups. These results already suggest that there are strong parallels between the composite order and prime order settings, but the translation techniques are developed in system-specific ways.

Beyond improving the assumptions and efficiency for particular schemes, our goal in this paper is to expand our general understanding of how tools that are conveniently inherent in the composite order setting can be simulated in the prime order setting. We begin by asking: what are the basic features of composite order bilinear groups that are typically exploited by cryptographic constructions and security proofs? Freeman considers this question in [14] and identifies two such features, called *projecting* and *canceling*. These properties are defined in Section 2.3 below. Freeman then provides examples of how to construct either of these properties using pairings of vectors of group elements in prime order groups. Notably, Freeman does not provide a way of simultaneously achieving *both* projecting and canceling. There may be good reason for this, since Meiklejohn, Shacham, and Freeman [30] have shown that both properties cannot be simultaneously achieved in prime order groups when one relies on the decisional linear assumption in a "natural way".

By instantiating either projecting or canceling in prime order groups, Freeman [14] successfully translates several composite order schemes into prime order schemes: the Boneh-Goh-Nissim encryption scheme [9], the Boneh-Sahai-Waters traitor tracing system [10], and the Katz-Sahai-Waters predicate encryption scheme [24]. These translations are accomplished using a three step process. The first step is to write the scheme in an abstract framework (replacing subgroups by subspaces of vectors in the exponent), the second step is to translate the assumptions into prime order analogs, and the third step is to transfer the security proof.

There are two aspects of Freeman's approach that can render the results unsatisfying in certain cases. First, the step of translating the assumptions often does not result in standard assumptions like DLIN. A reduction to DLIN is only provided for the most basic variant of the subgroup decision assumption, and does not extend (for example) to the general subgroup decision assumption from [3]. Second, the step of translating the proof fails for many schemes, including all of the recent composite order schemes employing the dual system encryption proof methodology [27, 25, 29, 28, 26]. These schemes use only canceling and not projecting, and so this is unrelated to the limitations discussed in [30].

The reason for this failure is instructive to examine. As Freeman points out, "the recent identity-based encryption scheme of Lewko and Waters [27] uses explicitly in its security proof the fact that the group G has two subgroups of relatively prime order". The major obstacle here is not translating the description of the scheme or its assumptions - instead the problem lies in translating a trick in the security proof. The trick works as follows. Suppose we have a group G of order $N = p_1 p_2 \dots p_m$, where $p_1, \dots, p_m$ are distinct primes. Then if we take an element $g_1 \in G$ of order p_1 (i.e. an element of the subgroup of G with order p_1) and a random exponent $a \in \mathbb{Z}_N$, the group element g_1^a *reveals no information* about the value of a modulo the other primes. Only $a \bmod p_1$ is revealed. The fact that $a \bmod p_2$, for instance, is uniformly random

even conditioned on $a \bmod p_1$ follows from the Chinese Remainder Theorem. In the security proof of the Lewko-Waters scheme, there are elements of the form g_1^a in the public parameters, and the fact that $a \bmod p_2$ remains information-theoretically hidden is later used to argue that all the keys and ciphertext received by the attacker are properly distributed in the midst of a hybrid argument.

Clearly, in a prime order group, we cannot hope to construct subgroups with coprime orders. There are a few possible paths for resolving this difficulty. We could start by reworking proofs in the composite order setting to avoid using this trick and then hope to apply the techniques of [14] without modification. This approach is likely to result in more complicated (though still static) assumptions in the composite order setting, which will translate into more complicated assumptions in the prime order setting. Since we prefer to rely only on the decisional linear assumption, we follow an alternate strategy: finding a version of this trick in prime order groups that does not rely on coprimeness. This is possible because coprimeness here is used a mechanism for achieving "parameter hiding," meaning that some useful information is information-theoretically hidden from the attacker, even after the public parameters are revealed. We can construct an alternate mechanism in prime order groups that similarly enables a form of parameter hiding.

Our Contribution We present versatile tools that can be used to translate composite order bilinear systems relying on canceling to prime order bilinear systems, particularly those whose security proofs rely on general subgroup decision assumptions and employ the coprime mechanism discussed above. This includes schemes like [27], which could not be handled by Freeman's methods. Our tools are based in the dual pairing vector space framework initiated by Okamoto and Takashima [31, 32]. We observe that dual pairing vector spaces provide a mechanism for parameter hiding that can be used in place of coprimeness. We then formulate an assumption in prime order groups that can be used to mimic the effect of the general subgroup decision assumption in composite order groups. We prove that this assumption is implied by DLIN. Putting these ingredients together, we obtain a flexible toolkit for turning a new class of composite order constructions into prime order constructions that can be proven secure from DLIN.

We demonstrate the use of our toolkit by providing a translation of the composite order Lewko-Waters IBE construction [27]. This yields a prime order IBE construction that is proven fully secure from DLIN and also inherits the intuitive structure of the Boneh-Boyen IBE [5]. Compared to the fully secure prime order IBE construction in [39], our scheme achieves comparable efficiency and security with a simpler structure. As a second application, we provide a translation of the Lewko-Waters unbounded HIBE scheme [29]. This additionally demonstrates how to handle delegation of secret keys with our tools.

We note that some composite order systems employing dual system encryption, such as the attribute-based encryption scheme in [25], already have analogs in prime order groups proven secure from DLIN using dual pairing vector spaces. In [33], Okamoto and Takashima provide a functional encryption scheme in prime order bilinear groups that is proven fully secure under DLIN. Their construction encompasses both attribute-based and inner product encryption, and their proof relies on dual system encryption techniques, similarly to [25]. While they focus on providing a particular construction and proof, our goal is to formulate a more general strategy for translating composite order schemes into prime order schemes with analogous proofs.

1.1 Related Work

The concept of identity-based encryption was first proposed by Shamir [36] and later constructed by Boneh and Franklin [8] and Cocks [13]. In an identity-based encryption scheme, users are associated with identities and obtain secret keys from a master authority. Encryption to any identity can be done knowing only the identity and some global public parameters. Both of the initial constructions of IBE were proven secure in the random oracle model. The first standard model constructions, by Canetti, Halevi, and Katz [11] and Boneh and Boyen [5] relied on selective security, which is a more restrictive security model requiring the attacker to announce the identity to be attacker prior to viewing the public parameters. Subsequently, Boneh and Boyen [6], Gentry [16], and Waters [38, 39] provided constructions proven fully secure in the standard model from various assumptions. Except for the scheme of [13], which relied on the quadratic residuousity assumption, all of the schemes we have cited above rely on bilinear groups. A lattice-based IBE construction was first provided by Gentry, Peikert, and Vaikuntanathan in [18].

Hierarchical identity-based encryption was proposed by Horwitz and Lynn [23] and then constructed by Gentry and Silverberg [19] in the random oracle model. In a HIBE scheme, users are associated with identity vectors that indicate their places in a hierarchy (a user Alice is a superior of the user Bob if her identity vector is a prefix of his). Any user can obtain a secret key for his identity vector either from the master authority or from one of his superiors (i.e. a mechanism for key delegation to subordinates is provided). Selectively secure standard model constructions of HIBE were provided by Boneh and Boyen [5] and Boneh, Boyen, and Goh [7] in the bilinear setting and by Cash, Hofheinz, Kiltz, and Peikert [12] and Agrawal, Boneh, and Boyen [1, 2] in the lattice-based setting. Fully secure constructions allowing polynomial depth were given by Gentry and Halevi [17], Waters [39], and Lewko and Waters [27]. The first unbounded construction (meaning that the maximal depth is not bounded by the public parameters) was given by Lewko and Waters in [29].

Attribute-based encryption (ABE) is a more flexible functionality than (H)IBE, first introduced by Sahai and Waters in [35]. In an ABE scheme, keys and ciphertexts are associated with attributes and access policies instead of identities. In a ciphertext-policy ABE scheme, keys are associated with attributes and ciphertexts are associated with access policies. In a key-policy ABE scheme, keys are associated with access policies and ciphertexts are associated with attributes. In both cases, a key can decrypt a ciphertext if and only if the attributes satisfy the formula. There are several constructions of both kinds of ABE schemes, e.g. [35, 20, 34, 4, 25, 33, 40].

The dual system encryption methodology was introduced by Waters in [39] as a tool for proving full security of advanced functionalities such as (H)IBE and ABE. It was further developed in several subsequent works [27, 25, 33, 26, 29, 28]. Most of these works have used composite order groups as a convenient setting for instantiating the dual system methodology, with the exception of [33]. Here, we extend and generalize the techniques of [33] to demonstrate that this use of composite order groups can be viewed as an intermediary step in the development of prime order systems whose security relies on the DLIN assumption.

1.2 Organization

In Section 2, we provide the necessary background on composite order bilinear groups, prime order bilinear groups, and dual pairing vector spaces. In Section 3, we construct the main tools that we will employ to simulate relevant features of composite order groups in the prime order setting. In Section 4, we demonstrate the use of our tools by providing a prime order translation of the Lewko-Waters composite order IBE scheme from [27]. This provides a close

variant of the original Boneh-Boyen IBE scheme [5] that is proven fully secure from the decisional linear assumption. In Section 5, we briefly discuss how our techniques can be easily extended to schemes providing delegation capabilities, specifically the Lewko-Waters unbounded HIBE scheme in [29]. In Section 6, we discuss further extensions of our techniques. In Appendix A, we provide the formal definitions for IBE and HIBE and their standard IND-CPA security definitions. Finally, in Appendix B, we give the complete details of our prime order translation of the Lewko-Waters unbounded HIBE scheme and the proof of its security. We advise the reader that Appendix B is included for completeness, but is not necessary to understanding the main ideas of our work.

2 Background

2.1 Composite Order Bilinear Groups

When G is a bilinear group of composite order $N = p_1 p_2 \dots p_m$ (where $p_1, p_2, \dots, p_m$ are distinct primes), we let $e : G \times G \to G_T$ denote its bilinear map (also referred to as a pairing). We note that both G and G_T are cyclic groups of order N. For each p_i, G has a subgroup of order p_i denoted by G_{p_i}. We let $g_1, \dots, g_m$ denote generators of G_{p_1} through G_{p_m} respectively. Each element $g \in G$ can be expressed as $g = g_1^{a_1} g_2^{a_2} \cdots g_m^{a_m}$ for some $a_1, \dots, a_m \in \mathbb{Z}_N$, where each a_i is unique modulo p_i. We will refer to $g_i^{a_i}$ as the "G_{p_i} component" of g. When a_i is congruent to zero modulo p_i, we say that g has no G_{p_i} component. The subgroups $G_{p_1}, \dots, G_{p_m}$ are "orthogonal" under the bilinear map e, meaning that if $h \in G_{p_i}$ and $u \in G_{p_j}$ for $i \neq j$, then $e(h, u) = 1$, where 1 denotes the identity element in G_T.

General Subgroup Decision Assumption The general subgroup decision assumption for composite order bilinear groups (formulated in [3]) is a family of static complexity assumptions based on the intuition that it should be hard to determine which components are present in a random group element, except for what can be trivially determined by testing for orthogonality with other given group elements. More precisely, for each non-empty subset $S \subseteq [m]$, there is an associated subgroup of order $\prod_{i \in S} p_i$ in G, which we will denote by G_S. For two distinct, non-empty subsets S_0 and S_1, we assume it is hard to distinguish a random element of G_{S_0} from a random element of G_{S_1}, when one is only given random elements of $G_{S_2}, \dots, G_{S_k}$ where for each $2 \leq j \leq k$, either $S_j \cap S_0 = \emptyset = S_j \cap S_1$ or $S_j \cap S_0 \neq \emptyset \neq S_j \cap S_1$.

More formally, we let $\mathcal{G}$ denote a group generation algorithm, which takes in m and a security parameter λ and outputs a bilinear group G of order $N = p_1 \cdots p_m$, where $p_1, \dots, p_m$ are distinct primes. The General Subgroup Decision Assumption with respect to $\mathcal{G}$ is defined as follows.

Definition 1. *General Subgroup Decision Assumption. Let $S_0, S_1, S_2, \dots, S_k$ be non-empty subsets of $[m]$ such that for each $2 \leq j \leq k$, either $S_j \cap S_0 = \emptyset = S_j \cap S_1$ or $S_j \cap S_0 \neq \emptyset \neq S_j \cap S_1$. Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (N = p_1 \cdots p_m, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$Z_0 \xleftarrow{R} G_{S_0},\ Z_1 \xleftarrow{R} G_{S_1},\ Z_2 \xleftarrow{R} G_{S_2}, \dots, Z_k \xleftarrow{R} G_{S_k},$$

$$D := (\mathbb{G}, Z_2, \dots, Z_k).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0, 1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}[\mathcal{A}(D, Z_0) = 1] - \mathbb{P}[\mathcal{A}(D, Z_1) = 1]|$$

is negligible in the security parameter λ.

We note that this assumption holds in the generic group model, assuming it is hard to find a non-trivial factor of the group order N.

Restricting to challenge sets differing by one element We observe that it suffices to consider challenge sets S_0 and S_1 of the form $S_1 = S_0 \cup \{i\}$ for some $i \in [m]$, $i \notin S_0$. We refer to this restricted class of subgroup decision assumptions as the 1-General Subgroup Decision Assumption. To see that the 1-general subgroup decision assumption implies the general subgroup decision assumption, we show that any instance of the general subgroup decision assumption is implied by a sequence of the more restricted instances. More precisely, for general S_0, S_1, we let U denote the set $S_0 \cup S_1 - S_0$. For any i in U, the 1-general subgroup decision assumption implies that it hard to distinguish a random element of G_{S_0} from a random element of $G_{S_0 \cup \{i\}}$, even given random elements from $G_{S_2}, \ldots, G_{S_k}$. That is because each of the sets $S_2, \ldots, S_k$ either does not intersect S_1 or S_0 and hence does not intersect S_0 or $S_0 \cup \{i\} \subseteq S_1$, or intersects both S_0 and $S_0 \cup \{i\}$. We can now incrementally add the other elements of U using instances of the 1-general subgroup decision assumption, ultimately showing that it is hard to distinguish a random element of G_{S_0} from a random element of $G_{S_0 \cup S_1}$. We can reverse the process and subtract one element at a time from $S_0 \cup S_1$ until we arrive at S_1. Thus, the seemingly more restrictive 1-general subgroup decision assumption implies the general subgroup decision assumption.

2.2 Prime Order Bilinear Groups

We now let G denote a bilinear group of prime order p, with bilinear map $e : G \times G \to G_T$. More generally, one may have a bilinear map $e : G \times H \to G_T$, where G and H are different groups. For simplicity in this paper, we will always consider groups where $G = H$.

In addition to referring to individual elements of G, we will also consider "vectors" of group elements. For $\vec{v} = (v_1, \ldots, v_n) \in \mathbb{Z}_p^n$ and $g \in G$, we write $g^{\vec{v}}$ to denote a n-tuple of elements of G:

$$g^{\vec{v}} := (g^{v_1}, g^{v_2}, \ldots, g^{v_n}).$$

We can also perform scalar multiplication and vector addition in the exponent. For any $a \in \mathbb{Z}_p$ and $\vec{v}, \vec{w} \in \mathbb{Z}_p^n$, we have:

$$g^{a\vec{v}} := (g^{av_1}, \ldots, g^{av_n}), \quad g^{\vec{v}+\vec{w}} = (g^{v_1+w_1}, \ldots, g^{v_n+w_n}).$$

We define e_n to denote the product of the componentwise pairings:

$$e_n(g^{\vec{v}}, g^{\vec{w}}) := \prod_{i=1}^{n} e(g^{v_i}, g^{w_i}) = e(g, g)^{\vec{v} \cdot \vec{w}}.$$

Here, the dot product is taken modulo p.

Dual Pairing Vector Spaces We will employ the concept of dual pairing vector spaces from [31, 32]. For a fixed (constant) dimension n, we will choose two random bases $\mathbb{B} := (\vec{b}_1, \ldots, \vec{b}_n)$ and $\mathbb{B}^* := (\vec{b}_1^*, \ldots, \vec{b}_n^*)$ of $\mathbb{Z}_p^n$, subject to the constraint that they are "dual orthonormal", meaning that

$$\vec{b}_i \cdot \vec{b}_j^* = 0 \pmod{p},$$

whenever $i \neq j$, and

$$\vec{b}_i \cdot \vec{b}_i^* = \psi$$

for all i, where ψ is a uniformly random element of $\mathbb{Z}_p$. (This is a slight abuse of the terminology "orthonormal", since ψ is not constrained to be 1.)

For a generator $g \in G$, we note that

$$e_n(g^{\vec{b}_i}, g^{\vec{b}_j^*}) = 1$$

whenever $i \neq j$, where 1 here denotes the identity element in G_T.

We note that choosing random dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ can equivalently be thought of as choosing a random basis $\mathbb{B}$, choosing a random vector $\vec{b}_1^*$ subject to the constraint that it is orthogonal to $\vec{b}_2, \ldots, \vec{b}_n$, defining $\psi = \vec{b}_1 \cdot \vec{b}_1^*$, and then choosing $\vec{b}_2^*$ so that it is orthogonal to $\vec{b}_1, \vec{b}_3, \ldots, \vec{b}_n$, and has dot product with $\vec{b}_2$ equal to ψ, and so on. We will later use the notation $(\mathbb{D}, \mathbb{D}^*)$ and $\vec{d}_1, \ldots$, etc. to also denote dual orthonormal bases and their vectors (and even $\mathbb{F}, \mathbb{F}^*$ and $\vec{f}_1$, etc.). This is because we will sometimes be handling more than one pair of dual orthonormal bases at a time, and we use different notation to avoid confusing them.

Decisional Linear Assumption The complexity assumption we will rely on in prime order bilinear groups is the Decisional Linear Assumption. To define this formally, we let $\mathcal{G}$ denote a group generation algorithm, which takes in a security parameter λ and outputs a bilinear group G of order p.

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

$$\mathbb{G} := (p, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g, f, v, w \xleftarrow{R} G,\ c_1, c_2, w \xleftarrow{R} \mathbb{Z}_p,$$

$$D := (g, f, v, f^{c_1}, v^{c_2}).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := \left| \mathbb{P}\left[\mathcal{A}(D, g^{c_1+c_2}) = 1\right] - \mathbb{P}\left[\mathcal{A}(D, g^{c_1+c_2+w}) = 1\right] \right|$$

is negligible in the security parameter λ.

2.3 Previously Investigated Abstract Properties of Bilinear Groups

We now define the abstract properties of projecting and canceling as formulated by Freeman [14]:

Projecting bilinear maps We say a bilinear map $e : G \times H \to G_T$ is projecting if there exist subgroups $G_1 \subset G, H_1 \subset H, G_T' \subset G$ and group homomorphisms π_1, π_2, and π_T mapping G, H, G_T into themselves (respectively) such that G_1 is the kernel of π_1, H_1 is the kernel of π_2, G_T' is the kernel of π_T, and these "projection maps" π_1, π_2, π_T commute with e in the sense that:

$$e(\pi_1(g), \pi_2(h)) = \pi_T(e(g, h))\ \forall g \in G, h \in H.$$

This structure is naturally achieved when the groups G, H, G_T are of composite order $N = p_1 \cdots p_m$ (where $p_1, \ldots, p_m$ are distinct primes), since we may take G_1, H_1, and G'_T to be the subgroups of order p_1 inside G, H, G_T respectively, and define the projections π_1, π_2 to be exponentiation by p_1 while π_T is exponentiation by p_1^2.

In this work, we will not consider projecting pairings and will instead be concerned with canceling pairings:

Canceling bilinear maps We say a bilinear map $e : G \times H \to G_T$ is canceling if there are subgroups $G_1, \ldots, G_m$ of G and $H_1, \ldots, H_m$ of H such that $G \cong G_1 \times \ldots \times G_m$, $H \cong H_1 \times \ldots \times H_m$, and $e(g_i, h_j) = 1$ in G_T whenever $g_i \in G_i, h_j \in H_j$ for $i \neq j$ (here 1 represents the identity element of G_T). This structure is achieved naturally when the groups G, H, G_T are of composite order $N = p_1 \cdots p_m$ (where $p_1, \ldots, p_m$ are distinct primes), since we may set $G_i = G_{p_i}$, $H_i = H_{p_i}$ to be the subgroups of order p_i for each i. This structure is also achieved by dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ in prime order groups, where each subgroup G_i corresponds to the span of vector $\vec{b}_i$ in the exponent, and each subgroup H_i corresponds to the span of the vector $\vec{b}_i^*$ in the exponent. Here, the underlying bilinear map $e : G \times G \to G_T$ acts on two copies of the same group G, but we use *different* bases in the exponents of our pairing e_n on n-tuples of group elements.

3 Our Main Tools

There is an additional feature of composite order groups that is often exploited along with canceling in the security proofs for composite order constructions: we call this *parameter hiding*. In composite order groups, parameter hiding takes the following form. Consider a composite order group G of order $N = p_1 p_2$ and an element $g_1 \in G_{p_1}$ (an element of order p_1). Then if we sample a uniformly random exponent $a \in Z_N$ and produce g_1^a, this reveals nothing about the value of a modulo p_2. More precisely, the Chinese Remainder theorem guarantees that the value of a modulo p_2 conditioned on the value of a modulo p_1 is still uniformly random, and g_1^a only depends on the value of a modulo p_1. This allows a party choosing a to publish g_1^a and still *hide* some information about a, namely its value modulo p_2. Note that this party only needs to know N and g_1: it does not need to know the factorization of N.

This is an extremely useful tool in security proofs, enabling a simulator to choose some secret random exponents, publish the public parameters by raising known subgroup elements to these exponents, and still information-theoretically hide the values of these exponents modulo some of the primes. These hidden values can be leveraged later in the security game to argue that something looks well-distributed in the attacker's view, even if this does not hold in the simulator's view. This sort of trick is crucial in proofs employing the dual system encryption methodology.

Replicating this trick in prime order groups seems challenging, since if one is given g and g^a in a prime order group, a is completely revealed modulo p in an information-theoretic sense. To resolve this issue, we use dual pairing vector spaces. We observe that a form of parameter hiding is achieved by using dual orthonormal bases: one can generate a random pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ for $\mathbb{Z}_p^n$, apply an invertible change of basis matrix A to a subset of these basis vectors, and produce a new pair of dual orthonormal bases which is also randomly distributed, *independently of* A. This allows us to *hide* a random matrix A. We formulate this precisely below.

3.1 Parameter Hiding in Dual Orthonormal Bases

We consider taking dual orthonormal bases and applying a linear change of basis to a subset of their vectors. We do this in such a way that we produce new dual orthonormal bases. In this subsection, we prove that if we start with randomly sampled dual orthonormal bases, then the resulting bases will also be random - in particular, the distribution of the final bases reveals nothing about the change of basis matrix that was employed. This "hidden" matrix can then be leveraged in security proofs as a way of separating the simulator's view from the attacker's.

To describe this formally, we let $m \leq n$ be fixed positive integers and $A \in \mathbb{Z}_p^{m \times m}$ be an invertible matrix. We let $S_m \subseteq [n]$ be a subset of size m ($|S| = m$). For any dual orthonormal bases $\mathbb{B}, \mathbb{B}^*$, we can then define new dual orthonormal bases $\mathbb{B}_A, \mathbb{B}_A^*$ as follows. We let B_m denote the $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are the vectors $\vec{b}_i \in \mathbb{B}$ such that $i \in S_m$. Then $B_m A$ is also an $n \times m$ matrix. We form $\mathbb{B}_A$ by retaining all of the vectors $\vec{b}_i \in \mathbb{B}$ for $i \notin S_m$ and exchanging the $\vec{b}_i$ for $i \in S_m$ with the columns of $B_m A$. To define $\mathbb{B}_A^*$, we similarly let B_m^* denote the $n \times m$ matrix over $\mathbb{Z}_p$ whose columns are the vectors $\vec{b}_i^* \in \mathbb{B}^*$ such that $i \in S_m$. Then $B_m^*(A^{-1})^t$ is also an $n \times m$ matrix, where $(A^{-1})^t$ denotes the transpose of A^{-1}. We form $\mathbb{B}_A^*$ by retaining all of the vectors $\vec{b}_i^* \in \mathbb{B}^*$ for $i \notin S_m$ and exchanging the $\vec{b}_i$ for $i \in S_m$ with the columns of $B_m^*(A^{-1})^t$.

To see that $\mathbb{B}_A$ and $\mathbb{B}_A^*$ are dual orthonormal bases, note that for $i \in S_m$, the corresponding basis vector in $\mathbb{B}_A$ can be expressed as a linear combination of the basis vectors $\vec{b}_j \in \mathbb{B}$ with $j \in S_m$, and the coefficients of this linear combination correspond to a column of A, say the ℓ^{th} column (equivalently, say i is the ℓ^{th} element of S_m). When $\ell \neq \ell'$, the ℓ^{th} column of A is orthogonal to the $(\ell')^{th}$ column of $(A^{-1})^t$. This means that the i^{th} vector of $\mathbb{B}_A$ will be orthogonal to the $(i')^{th}$ vector of $\mathbb{B}_A^*$ whenever $i \neq i'$. Moreover, the ℓ^{th} column of A and the ℓ^{th} column of $(A^{-1})^t$ have dot product equal to 1, so the dot product of the i^{th} vector of $\mathbb{B}_A$ and the i^{th} vector of $\mathbb{B}_A^*$ will be equal to the same value ψ as in the original bases $\mathbb{B}$ and $\mathbb{B}^*$.

For a fixed dimension n and prime p, we let $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} \mathit{Dual}(\mathbb{Z}_p^d)$ denote choosing random dual orthonormal bases $\mathbb{B}$ and $\mathbb{B}^*$ of $\mathbb{Z}_p^n$. Here, $\mathit{Dual}(\mathbb{Z}_p^n)$ denotes the set of dual orthonormal bases.

Lemma 3. *For any fixed positive integers $m \leq n$, any fixed invertible $A \in \mathbb{Z}_p^{m \times m}$ and set $S_m \subseteq [n]$ of size m, if $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} \mathit{Dual}(\mathbb{Z}_p^d)$, then $(\mathbb{B}_A, \mathbb{B}_A^*)$ is also distributed as a random sample from $\mathit{Dual}(\mathbb{Z}_p^d)$. In particular, the distribution of $(\mathbb{B}_A, \mathbb{B}_A^*)$ is independent of A.*

Proof. There is a one-to-one correspondence between $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{B}_A, \mathbb{B}_A^*)$: given $(\mathbb{B}_A, \mathbb{B}_A^*)$, one can recover $(\mathbb{B}, \mathbb{B}^*)$ by applying A^{-1} to the vectors in $\mathbb{B}_A$ whose indices are in S_m, and applying A^t to the corresponding vectors in $\mathbb{B}_A^*$. This shows that every pair of dual orthonormal bases is equally likely to occur as $\mathbb{B}_A, \mathbb{B}_A^*$. □

3.2 Exploiting Hidden Parameters - An Example

We now give a small example of how a hidden change of basis matrix can be exploited by a simulator to hide a correlation from the attacker's view. This example relies simply on pairwise independence of the function $f(x) = ax + b$ for $x, a, b \in \mathbb{Z}_p$ when a and b are chosen uniformly at random. Even this basic case is already quite useful - in particular, we will use this to establish full security for a close analog of the selectively secure Boneh-Boyen IBE scheme.

Lemma 4. *Let $y, z \in \mathbb{Z}_p$, $z \neq y$. We define $\vec{z}$ to be the column vector $(1, z)^t$ and $\vec{y}$ to be the column vector $(y, -1)^t$. Let A be a 2×2 matrix over $\mathbb{Z}_p$ whose entries are chosen uniformly at random and γ, λ be chosen uniformly at random from $\mathbb{Z}_p$. Then the distribution of the vectors*

$\gamma A^{-1}\vec{z}$ and $\lambda A^t\vec{y}$ is negligibly close to the uniform distribution over $\mathbb{Z}_p^2 \times \mathbb{Z}_p^2$. (Note that A is invertible with all but negligible probability.)

Proof. We let a_{ij} denote the i,j entry of A for $i,j \in \{1,2\}$. We may assume that A is invertible, since this holds with overwhelming probability. Up to ignoring a scaling factor (which is irrelevant since our final vectors are randomly scaled by γ, λ), we can then write

$$A^{-1} = \begin{pmatrix} a_{22} & -a_{12} \\ -a_{21} & a_{11} \end{pmatrix}, \; A^t = \begin{pmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{pmatrix}.$$

We then have (still ignoring a scaling factor):

$$A^{-1}\vec{z} = \begin{pmatrix} a_{22} - za_{12} \\ -a_{21} + za_{11} \end{pmatrix}, \; A^t\vec{y} = \begin{pmatrix} a_{11}y - a_{21} \\ a_{12}y - a_{22} \end{pmatrix}.$$

Since $f(x) := a_{11}x - a_{21}$ and $g(x) := a_{12}x - a_{22}$ are both pairwise independent functions of x, and $y \neq z$, we have that these four entries are distributed as uniformly random in $\mathbb{Z}_p^4$. □

Later, we will apply this lemma where y and z are distinct identities in an IBE scheme.

3.3 The Subspace Assumption

We now state a complexity assumption in prime order groups that we will use to simulate the effects of subgroup decision assumptions in composite order groups. We call this the Subspace Assumption. We show that the subspace assumption is implied by the decisional linear assumption.

In prime order groups, basis vectors in the exponent take the place of subgroups. Since we are using dual orthonormal bases, our new concept of orthogonality between "subgroups" becomes asymmetric. If we have dual orthonormal bases $\mathbb{B}, \mathbb{B}^*$ and we think of "subgroup 1" in $\mathbb{B}$ as corresponding to the span of $\vec{b}_1, \ldots, \vec{b}_4$, then this is not orthogonal to the other vectors in $\mathbb{B}$, but it is orthogonal to vectors $\vec{b}_5^*, \ldots, \vec{b}_n^*$ in $\mathbb{B}^*$. Essentially, the notion of a single subgroup has now been split into a pair of "subgroups", one for each side of the pairing, and orthogonality between different subgroups now only holds for elements on opposite sides.

This sort of asymmetry can be quite useful. For example, consider an instance of the general subgroup decision assumption in composite order groups, where the task is to distinguish a random element of G_{p_1} from $G_{p_1p_2}$. In this case, we cannot give out an element of G_{p_2}, since it can trivially be used to break the assumption by pairing it with the challenge term and seeing if the result is the identity. If we instead use dual orthonormal bases in a prime order group, the situation is a bit different. Suppose that given $g^{\vec{v}}$, the task is to distinguish whether the exponent vector $\vec{v}$ is in the span of $\vec{b}_1^*, \vec{b}_2^*$ or in the larger span of $\vec{b}_1^*, \vec{b}_2^*, \vec{b}_3^*$. We cannot give out $g^{\vec{b}_3}$, since one could then break the assumption by testing if $e_n(g^{\vec{v}}, g^{\vec{b}_3}) = e(g,g)^{\vec{v}\cdot\vec{b}_3}$ is the identity, but *we can give out* $g^{\vec{b}_3^*}$.

Our definition of the subspace assumption is motivated by this and our observation in Section 2.1 that the general subgroup decision assumption in composite order groups can be restricted to distinguishing between sets that differ by one element. What this means is that to simulate the uses of the general subgroup decision in composite order groups, one can focus merely on creating an analog for expansion into one new "subgroup" at a time. At its core, our subspace assumption says that if one is given $g^{\vec{v}}$, then it is hard to tell if $\vec{v}$ is randomly chosen from the span of $\vec{b}_1^*, \vec{b}_2^*$ or from the larger span of $\vec{b}_1^*, \vec{b}_2^*, \vec{b}_3^*$, even if one is given scalar multiples of all bases vectors in $\mathbb{B}$ and $\mathbb{B}^*$ in the exponent, *except for* $\vec{b}_3$. We augment this by also given out a random linear combination of $\vec{b}_1, \vec{b}_2, \vec{b}_3$ in the exponent. We then generalize this by replicating

the same structure for k 3-tuples of vectors, with the random linear combinations having the *same* coefficients. (The fact that these coefficients are the same prevents this from following immediately from the assumption for a single 3-tuple applied in hybrid fashion.)

We now give the formal description of the subspace assumption. For a fixed dimension $n \geq 3$ and prime p, we recall that $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^n)$ denotes choosing random dual orthonormal bases $\mathbb{B}$ and $\mathbb{B}^*$ of $\mathbb{Z}_p^n$, and $Dual(\mathbb{Z}_p^n)$ denotes the set of dual orthonormal bases. Our assumption is additionally parameterized by a positive integer $k \leq \frac{n}{3}$.

Definition 5. *(Subspace Assumption) Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (p, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^n),$$

$$g \xleftarrow{R} G,\ \eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \xleftarrow{R} \mathbb{Z}_p,$$

$$U_1 := g^{\mu_1 \vec{b}_1 + \mu_2 \vec{b}_{k+1} + \mu_3 \vec{b}_{2k+1}},\ U_2 := g^{\mu_1 \vec{b}_2 + \mu_2 \vec{b}_{k+2} + \mu_3 \vec{b}_{2k+2}},\ \ldots,\ U_k := g^{\mu_1 \vec{b}_k + \mu_2 \vec{b}_{2k} + \mu_3 \vec{b}_{3k}},$$

$$V_1 := g^{\tau_1 \eta \vec{b}^*_1 + \tau_2 \beta \vec{b}^*_{k+1}},\ V_2 := g^{\tau_1 \eta \vec{b}^*_2 + \tau_2 \beta \vec{b}^*_{k+2}},\ \ldots,\ V_k := g^{\tau_1 \eta \vec{b}^*_k + \tau_2 \beta \vec{b}^*_{2k}}$$

$$W_1 := g^{\tau_1 \eta \vec{b}^*_1 + \tau_2 \beta \vec{b}^*_{k+1} + \tau_3 \vec{b}^*_{2k+1}},\ W_2 := g^{\tau_1 \eta \vec{b}^*_2 + \tau_2 \beta \vec{b}^*_{k+2} + \tau_3 \vec{b}^*_{2k+2}},\ \ldots,\ W_k := g^{\tau_1 \eta \vec{b}^*_k + \tau_2 \beta \vec{b}^*_{2k} + \tau_3 \vec{b}^*_{3k}}$$

$$D := \big(g^{\vec{b}_1}, g^{\vec{b}_2}, \ldots, g^{\vec{b}_{2k}}, g^{\vec{b}_{3k+1}}, \ldots, g^{\vec{b}_n}, g^{\eta \vec{b}^*_1}, \ldots, g^{\eta \vec{b}^*_k}, g^{\beta \vec{b}^*_{k+1}}, \ldots, g^{\beta \vec{b}^*_{2k}}, g^{\vec{b}^*_{2k+1}}, \ldots, g^{\vec{b}^*_n}, U_1, U_2, \ldots, U_k, \mu_3\big).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}\left[\mathcal{A}(D, V_1, \ldots, V_k) = 1\right] - \mathbb{P}\left[\mathcal{A}(D, W_1, \ldots, W_k) = 1\right]|$$

is negligible in the security parameter λ.

We have included in D more terms than will be necessary for many applications of this assumption, and in what follows we will often omit those we do not need. We will work exclusively with the $k = 1$ and $k = 2$ cases. We present the assumption in the form above in order make it more versatile for use in future applications. We additionally note that the form stated above can be further generalized to involve multiple, independently generated dual orthonormal bases $(\mathbb{B}_1, \mathbb{B}^*_1), (\mathbb{B}_2, \mathbb{B}^*_2), \ldots, (\mathbb{B}_j, \mathbb{B}^*_j)$, for any fixed j. The terms in the assumption would be duplicated for each pair of bases, with the same values of $\eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3$. We will not need this generalization for the applications we present. To help the reader see the main structure of this assumption through the burdensome notation, we include heuristic illustrations of the $k = 1$ and $k = 2$ cases below.

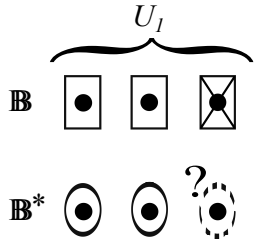

Figure 1: Subspace Assumption with $k = 1$

In these diagrams, the top rows illustrate the U terms, while the bottom rows illustrate the V, W terms. The solid ovals and rectangles indicate the presence of basis vectors. The crossed rectangles indicate basis elements of $\mathbb{B}$ which are present in U_1, U_2 but are not given out in isolation. The dotted ovals adorned by question marks indicate the basis vectors whose presence depends on whether we consider the V's or the W's.

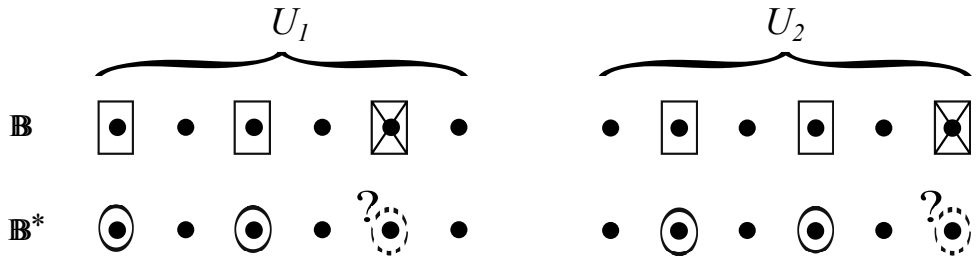

Figure 2: Subspace Assumption with $k = 2$

3.4 Reduction to the Decisional Linear Assumption

We now show that our subspace assumption is implied by the decisional linear assumption.

Lemma 6. *If the decisional linear assumption holds for a group generator $\mathcal{G}$, then the subspace assumption stated in Definition 5 also holds for $\mathcal{G}$.*

Proof. We assume there exists a PPT algorithm $\mathcal{A}$ breaking the subspace assumption with non-negligible advantage (for some fixed positive integers k, n satisfying $n \geq 3k$). We will use this to create a PPT algorithm $\mathcal{B}$ which breaks the decisional linear assumption with non-negligible advantage. $\mathcal{B}$ is given $g, f, v, f^{c_1}, v^{c_2}, T$, where T is either $g^{c_1+c_2}$ or T is a uniformly random element of G. We let ℓ_f denote the discrete logarithm base g of f and ℓ_v denote the discrete logarithm base g of v, i.e. $f = g^{\ell_f}$ and $v = g^{\ell_v}$.

$\mathcal{B}$ simulates the subspace assumption for $\mathcal{A}$ as follows. $\mathcal{B}$ first samples random dual orthonormal bases, denoted by $\vec{d}_1, \ldots, \vec{d}_n$ and $\vec{d}_1^*, \ldots, \vec{d}_n^*$. In other words, $\mathcal{B}$ chooses vectors $\vec{d}_1, \ldots, \vec{d}_n, \vec{d}_1^*, \ldots, \vec{d}_n^*$ randomly, subject to the constraints that $\vec{d}_i \cdot \vec{d}_j^* \equiv 0 \bmod p$ when $i \neq j$, and $\vec{d}_i \cdot \vec{d}_i^* \equiv \psi \bmod p$ for all i from 1 to n, where ψ is a random element of $\mathbb{Z}_p$. Now, $\mathcal{B}$ implicitly sets:

$$\eta\vec{b}_1^* = \vec{d}_{2k+1}^* + \ell_f\vec{d}_1^*,\ \eta\vec{b}_2^* = \vec{d}_{2k+2}^* + \ell_f\vec{d}_2^*,\ \ldots,\ \eta\vec{b}_k^* = \vec{d}_{3k}^* + \ell_f\vec{d}_k^*,$$

$$\beta\vec{b}_{k+1}^* = \vec{d}_{2k+1}^* + \ell_v\vec{d}_{k+1}^*,\ \beta\vec{b}_{k+2}^* = \vec{d}_{2k+2}^* + \ell_v\vec{d}_{k+2}^*,\ \ldots,\ \beta\vec{b}_{2k}^* = \vec{d}_{3k}^* + \ell_v\vec{d}_{2k}^*,$$

$$\vec{b}_{2k+1}^* = \vec{d}_{2k+1}^*,\ \ldots, \vec{b}_n^* = \vec{d}_n^*.$$

We think of this as setting $\eta = \ell_f$ and $\beta = \ell_v$, with $\vec{b}_1^* = \eta^{-1}\vec{d}_{2k+1}^* + \vec{d}_1^*$ for example.

$\mathcal{B}$ sets the dual basis as:

$$\vec{b}_1 = \vec{d}_1,\ \vec{b}_2 = \vec{d}_2,\ \ldots, \vec{b}_{2k} = \vec{d}_{2k},$$

$$\vec{b}_{2k+1} = \vec{d}_{2k+1} - \ell_f^{-1}\vec{d}_1 - \ell_v^{-1}\vec{d}_{k+1},\ \ldots,\ \vec{b}_{3k} = \vec{d}_{3k} - \ell_f^{-1}\vec{d}_k - \ell_v^{-1}\vec{d}_{2k},$$

$$\vec{b}_{3k+1} = \vec{d}_{3k+1},\ \ldots,\ \vec{b}_n = \vec{d}_n.$$

We observe that under these definitions, $\vec{b}_i \cdot \vec{b}_j^* \equiv 0 \bmod p$ whenever $i \neq j$, and $\vec{b}_i \cdot \vec{b}_i^* = \vec{d}_i \cdot \vec{d}_i^* = \psi$ for all i from 1 to n. We note that $\mathcal{B}$ can produce all of $g^{\eta\vec{b}_1^*}, \ldots, g^{\eta\vec{b}_k^*}, g^{\beta\vec{b}_{k+1}^*}, \ldots, g^{\beta\vec{b}_{2k}^*}$, $g^{\vec{b}_{2k+1}^*}, \ldots, g^{\vec{b}_n^*}$, $g^{\vec{b}_1}, \ldots, g^{\vec{b}_{2k}}$, and $g^{\vec{b}_{3k+1}}, \ldots, g^{\vec{b}_n}$, but *cannot produce* $g^{\vec{b}_{2k+1}}, \ldots, g^{\vec{b}_{3k}}$.

We argue that $\eta = \ell_f$, $\beta = \ell_v$, $\vec{b}_1, \ldots, \vec{b}_n$ and $\vec{b}_1^*, \ldots, \vec{b}_n^*$ are properly distributed. To see this, note that given any dual orthonormal bases $\vec{b}_1, \ldots, \vec{b}_n$ and $\vec{b}_1^*, \ldots, \vec{b}_n^*$, and any η, β, one can solve for a unique dual orthonormal bases $\vec{d}_1, \ldots, \vec{d}_n$ and $\vec{d}_1^*, \ldots, \vec{d}_n^*$ (with the same value of ψ) which yields $\vec{b}_1, \ldots, \vec{b}_n$ and $\vec{b}_1^*, \ldots, \vec{b}_n^*$ via the equations above. Thus, $\eta = \ell_f$, $\beta = \ell_v$, $\vec{b}_1, \ldots, \vec{b}_n$ and $\vec{b}_1^*, \ldots, \vec{b}_n^*$ are properly distributed.

Now $\mathcal{B}$ creates $U_1, \ldots, U_k$ as follows. It chooses random values $\mu'_1, \mu'_2, \mu'_3 \in \mathbb{Z}_p$. It sets:

$$U_1 = g^{\mu'_1 \vec{b}_1 + \mu'_2 \vec{b}_{k+1} + \mu'_3 \vec{d}_{2k+1}}.$$

We note that

$$\mu'_1 \vec{b}_1 + \mu'_2 \vec{b}_{k+1} + \mu'_3 \vec{d}_{2k+1} = (\mu'_1 + \ell_f^{-1} \mu'_3) \vec{b}_1 + (\mu'_2 + \ell_v^{-1} \mu'_3) \vec{b}_{k+1} + \mu'_3 \vec{b}_{2k+1}.$$

In other words, $\mathcal{B}$ has implicitly set $\mu_1 = \mu'_1 + \ell_f^{-1} \mu'_3$, $\mu_2 = \mu'_2 + \ell_v^{-1} \mu'_3$, and $\mu_3 = \mu'_3$. We note that these values are uniformly random, and μ_3 is known to $\mathcal{B}$. $\mathcal{B}$ can then form $U_2, \ldots, U_k$ as:

$$U_2 = g^{\mu'_1 \vec{b}_2 + \mu'_2 \vec{b}_{k+2} + \mu'_3 \vec{d}_{2k+2}}, \ \ldots, \ U_k = g^{\mu'_1 \vec{b}_k + \mu'_2 \vec{b}_{2k} + \mu'_3 \vec{d}_{3k}}.$$

$\mathcal{B}$ then implicitly sets $\tau_1 = c_1$ and $\tau_2 = c_2$. We note that

$$\begin{aligned} \tau_1 \eta \vec{b}^*_1 + \tau_2 \beta \vec{b}^*_{k+1} &= (c_1 + c_2) \vec{d}^*_{2k+1} + c_1 \ell_f \vec{d}^*_1 + c_2 \ell_v \vec{d}^*_{k+1}, \\ &\vdots \\ \tau_1 \eta \vec{b}^*_k + \tau_2 \beta \vec{b}^*_{2k} &= (c_1 + c_2) \vec{d}^*_{3k} + c_1 \ell_f \vec{d}^*_k + c_2 \ell_v \vec{d}^*_{2k}. \end{aligned}$$

The terms which are multiples of $c_1 \ell_f$ and $c_2 \ell_v$ are not difficult for $\mathcal{B}$ to produce as exponents of g, since $\mathcal{B}$ has $f^{c_1} = g^{c_1 \ell_f}$ and $v^{c_2} = g^{c_2 \ell_v}$. For the multiples of $c_1 + c_2$, $\mathcal{B}$ needs to use T.

$\mathcal{B}$ computes:

$$T_1 = T^{\vec{d}^*_{2k+1}} \left(f^{c_1}\right)^{\vec{d}^*_1} \left(v^{c_2}\right)^{\vec{d}^*_{k+1}}, \ \ldots, \ T_k = T^{\vec{d}^*_{3k}} \left(f^{c_1}\right)^{\vec{d}^*_k} \left(v^{c_2}\right)^{\vec{d}^*_{2k}}.$$

If $T = g^{c_1 + c_2}$, then these are distributed as $V_1, \ldots, V_k$. If $T = g^{c_1 + c_2 + w}$, then these are distributed as $W_1, \ldots, W_k$, with τ_3 implicitly set to w.

$\mathcal{B}$ gives

$$D := \big(g^{\vec{b}_1}, g^{\vec{b}_2}, \ldots, g^{\vec{b}_{2k}}, g^{\vec{b}_{3k+1}}, \ldots, g^{\vec{b}_n}, g^{\eta \vec{b}^*_1}, \ldots, g^{\eta \vec{b}^*_k}, g^{\beta \vec{b}^*_{k+1}}, \ldots, g^{\beta \vec{b}^*_{2k}},$$

$$g^{\vec{b}^*_{2k+1}}, \ldots, g^{\vec{b}^*_n}, U_1, U_2, \ldots, U_k, \mu_3\big)$$

to $\mathcal{A}$, along with $T_1, \ldots, T_k$. $\mathcal{B}$ can then leverage $\mathcal{A}$'s non-negligible advantage in distinguishing between the distributions $(V_1, \ldots, V_k)$ and $(W_1, \ldots, W_k)$ to achieve a non-negligible advantage in distinguishing $T = g^{c_1 + c_2}$ from $T = g^{c_1 + c_2 + w}$, hence violating the decisional linear assumption. We note that the above reduction can be parallelized for multiple bases $(\mathbb{B}_1, \mathbb{B}^*_1)$, $(\mathbb{B}_2, \mathbb{B}^*_2)$, ..., $(\mathbb{B}_j, \mathbb{B}^*_j)$ by having the simulator sample $(\mathbb{D}_1, \mathbb{D}^*_1), \ldots, (\mathbb{D}_j, \mathbb{D}^*_j)$ independently and follow the same procedure for each. □

4 Analog of the Boneh-Boyen IBE Scheme

In this section, we employ our subspace assumption and our parameter hiding technique for dual orthonormal bases to prove full security for a close analog of the Boneh-Boyen IBE scheme from the decisional linear assumption. This is the same security guarantee achieved for the IBE scheme in [39] and our efficiency is also similar. The advantage of our scheme is that it is a much closer analog to the original Boneh-Boyen IBE, and resultingly has a simpler, more intuitive structure.

Our security proof essentially mirrors the structure of the security proof given in [27], which provides a fully secure variant of the Boneh-Boyen IBE scheme in composite order groups. This serves as an illustrative example of how our techniques can be used to simulate dual system encryption proofs in the prime order setting that were originally presented in composite order groups.

4.1 Review of the Boneh-Boyen Scheme

We begin by reviewing the original Boneh-Boyen scheme [5] in prime order bilinear groups, which was proven to be selectively secure. In this scheme, the public parameters consist of three random elements of G and one element of G_T:

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

Here, α is random element of $\mathbb{Z}_p$, and $\text{MSK} = g^{\alpha}$. Identities are assumed to be elements of $\mathbb{Z}_p$, and a secret key for identity ID is of the form

$$\text{SK}_{ID} = \{g^{\alpha}(u^{ID}h)^r, g^r\},$$

where r is a random value in $\mathbb{Z}_p$ chosen by the master authority when it is called upon to issue this secret key. Messages are assumed to be elements of G_T, and an encryption of a message M to an identity ID takes the form:

$$\text{CT} = \{Me(g,g)^{\alpha s}, g^s, (u^{ID}h)^s\},$$

where s is a random value in $\mathbb{Z}_p$ chosen by the encryptor. Decryption works by computing two pairings and dividing the result to obtain:

$$e(g^{\alpha}(u^{ID}h)^r, g^s)/e(g^r, (u^{ID}h)^s) = e(g,g)^{\alpha s},$$

which can then be divided from $Me(g,g)^{\alpha s}$ to obtain M.

This scheme is quite elegant in its simplicity - every parameter plays a clear role. The randomness s is used to randomize ciphertexts. The randomness r embedded in a user's secret key prevents the user from recovering the master secret key. The parameter h prevents multiplicative manipulations of identities: for example, suppose one user has identity ID and another has identity $2ID$. If the parameter h were absent, the user with identity $2ID$ could take a ciphertext encrypted to ID and raise the last element to the power 2 to obtain a ciphertext for his identity $2ID$. The parameter u prevents users from removing the dependence on identities. For example, if we used g^{ID} in place of u^{ID}, then a user could take the g^r term in his secret key, raise it to the power ID, and use this to strip off the identity-dependent term from the first part of his key.

4.2 Review of the Lewko-Waters Composite Order Variant

The Lewko-Waters IBE scheme [27] takes the Boneh-Boyen IBE and embeds it into the first subgroup of a bilinear group of composite order $N = p_1p_2p_3$. Random elements from G_{p_3} are multiplied to key elements for additional randomization. This results in a scheme that retains the intuitive structure of Boneh-Boyen. In fact its description is almost identical, except that now g, u, h are replaced by $g_1, u_1, h_1 \in G_{p_1}$, and keys are of the form:

$$\text{SK}_{ID} = \{g_1^{\alpha}(u_1^{ID}h_1)^r R_3, g_1^r R_3'\},$$

where R_3, R_3' are randomly chosen elements of G_{p_3}. Since the ciphertext elements $g_1^s, (u_1^{ID}h_1)^s$ are contained in G_{p_1}, these extra terms R_3, R_3' are orthogonal to the ciphertext and do not hinder decryption.

Full security is proven using the dual system encryption methodology. In a dual system, there are two kinds of keys and ciphertexts: normal and semi-functional. Normal keys and ciphertexts are used in the real system, while their semi-functional counterparts are only invoked

in the proof. The relationships between these objects are as follows: normal keys can decrypt both normal and semi-functional ciphertexts, while semi-functional keys can only decrypt normal ciphertexts. When a semi-functional key is used to decrypt a semi-functional ciphertext, decryption will fail with all but negligible probability.

The security proof for a dual system is accomplished via a hybrid argument argument over a sequence of games. The first game is the real security game with normal keys and a normal ciphertext. In the next game, the ciphertext given to the attacker is changed to be semi-functional. Then, the keys given to the attacker are changed to be semi-functional, one by one. Once everything the attacker receives is semi-functional, then it is typically easy to prove security directly.

Semi-functional keys and ciphertexts in the LW IBE are just like normal keys and ciphertexts in the subgroups G_{p_1} and G_{p_3}, with additional random components in G_{p_2}. More precisely, a semi-functional ciphertext is of the form

$$\mathrm{CT} = \{Me(g_1, g_1)^{\alpha s}, g_1^s X_2, (u_1^{ID} h_1)^s X_2'\},$$

where X_2, X_2' are random elements in G_{p_2}. Similarly, a semi-functional key is of the form

$$\mathrm{SK}_{ID} = \{g_1^{\alpha} (u_1^{ID} h_1)^r R_3 Y_2, g_1^r R_3' Y_2'\},$$

where Y_2, Y_2' are random elements in G_{p_2}. Note that these elements in G_{p_2} affect decryption only when a semi-functional key and a semi-functional ciphertext are paired together.

To execute the game transitions in the hybrid proof, one must argue that an attacker's advantage cannot change noticeably between adjacent games. This is done by showing that if one is given a PPT attacker whose advantage noticeably changes, then one can create a PPT simulator which leverages this attacker to break a computational assumption. It is relatively straightforward to use a subgroup decision assumption to change the ciphertext from normal to semi-functional: the simulator will be given $g_1 \in G_{p_1}, g_3 \in G_{p_3}$ and T, and its task will be to decide if $T \in G_{p_1}$ or $T \in G_{p_1 p_2}$. It will set $u_1 = g_1^a$ and $h_1 = g_1^b$, where it knows $a, b \in \mathbb{Z}_N$, and can then implicitly set g_1^s to be the G_{p_1} component of T. It can compute the final element of the ciphertext as T^{aID+b}. If $T \in G_{p_1}$, this is a properly distributed normal ciphertext. If $T \in G_{p_1 p_2}$, this is a properly distributed semi-functional ciphertext (note that the values of a, b modulo p_2 are uniformly random, even conditioned on the public parameters, which only reveal their values modulo p_1).

There is a subtlety inherent in the proof while a particular key is changing from normal to semi-functional. We will refer to this key as the "challenge key." Since the proof provides full security, the simulator will not know ahead of time what the challenge identity will be. Hence, the simulator must be prepared to make the semi-functional ciphertext for any identity, and also to make the challenge key for any identity. The simulator should not know for itself whether the challenge key that it creates is normal or semi-functional - but it seems that it could determine this by creating a semi-functional ciphertext for the same identity and testing if decryption succeeds. To avoid this paradox, the simulator is designed so that if it were to make the semi-functional ciphertext and the challenge key for the same identity, the two objects would be correlated to ensure that decryption would succeed, regardless of whether the semi-functional components are present on the challenge key. In other words, if semi-functional components are present on the key, then they are correlated with the semi-functional components of the ciphertext so that they cancel out upon decryption if the identities are equal. This phenomenon is called *nominal semi-functionality*. This is where the parameter hiding technique is crucial: the same distribution of semi-functional components that is correlated in the simulator's view must look uncorrelated to the attacker, who can only request keys for identities *unequal* to

the identity of the challenge ciphertext. In the LW proof, this is accomplished via a pairwise independent function, $f(ID) := aID + b$ modulo p_2. The value of a modulo p_1 is the discrete log of u_1 base g_1, while the value of b modulo p_1 is the discrete log of h_1 base g_1. Information-theoretically, the public parameters reveal $a \bmod p_1$ and $b \bmod p_1$, but the values of $a \bmod p_2$ and $b \bmod p_2$ remain hidden. The pairwise independence of the function f modulo p_2 can thus be invoked to argue that the semi-functional components of the challenge ciphertext and challenge key appear properly distributed in the attacker's view. For more details of this argument, see [27].

The subgroup decision assumption used for this step in the proof is as follows: given random elements in G_{p_1}, G_{p_3}, $G_{p_1p_2}$, and $G_{p_2p_3}$, it should be hard to distinguish a random element of $G_{p_1p_3}$ from a random element of G. The element of G_{p_1} is used to make the public parameters, the element of G_{p_3} is used to randomize normal keys, the element of $G_{p_1p_2}$ is used to make the semi-functional ciphertext, the element of $G_{p_2p_3}$ is used to make the semi-functional keys, and the element of unknown type is used to make the challenge key (its G_{p_1} component is implicitly set to be g_1^r). Because the G_{p_2} components of the challenge ciphertext and the (possibly present) G_{p_2} components of the challenge key both enter via group elements that also have G_{p_1} components, the exponents of these elements must conform to the structure of the scheme that is enforced in the G_{p_1} subgroup - this is what causes nominal semi-functionality when the identities are the same: the cancelation that happens in the G_{p_1} subgroup is mirrored in the G_{p_2} subgroup. Essentially, what we get is a second copy of the scheme occurring in the G_{p_2} subgroup for the challenge key and challenge ciphertext, but with "fresh" parameters $a \bmod p_2$ and $b \bmod p_2$ that are not constrained by the public parameters. This hides the structure in G_{p_2} via pairwise independence when the identities are unequal.

4.3 Our Construction

We now construct an analog of the Boneh-Boyen IBE scheme in prime order bilinear groups that can be proven fully secure by mimicking the LW proof strategy. We will use dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*)$ of $\mathbb{Z}_p^6$, where p is the prime order of our bilinear group G. Public parameters and ciphertexts will have exponents described in terms of the basis vectors in $\mathbb{D}$, while secret keys will have exponents described in terms of $\mathbb{D}^*$. The first four basis vectors of each will constitute the "normal space" (like G_{p_1} in the LW scheme), and the last two basis vectors of each will constitute the "semi-functional space" (like G_{p_2} in the LW scheme).

By using dual pairing vector spaces, we avoid the need to simulate G_{p_3}. In the LW scheme, the purpose of G_{p_3} is to allow the creation of other semi-functional keys while a challenge key is changing from normal to semi-functional. More precisely, it allows the subgroup decision assumption to give out an element of $G_{p_2p_3}$ that can be used to generate semi-functional keys when the task is to distinguish a random element of $G_{p_1p_3}$ from a random element of G. We note that if we did not use G_{p_3} here and instead tried to create all of the semi-functional keys from a term in $G_{p_1p_2}$, then these keys would not be properly randomized in the G_{p_2} subgroup because the structure of the scheme is enforced in the G_{p_1} subgroup. Pairwise independence cannot save us here because there are many keys. However, the asymmetry of dual pairing vector spaces avoids this issue: while we are expanding the challenge key into the "semi-functional space" in $\mathbb{D}^*$, we can still know a basis for the semi-functional space of $\mathbb{D}^*$ in the exponent - it is only the corresponding terms in the semi-functional space of $\mathbb{D}$ that we do not have access to in isolation. This allows us to make the other semi-functional keys without needing to create an analog of the G_{p_3} subgroup.

As we reviewed above, the core of the Boneh-Boyen scheme is a cancelation between terms in two pairings, one with the identity appearing on the ciphertext side and the other with

the identity appearing on the key side. This is combined with a mechanism for preventing multiplication manipulation of the identity. In our scheme, this core cancelation is duplicated: instead of having one cancelation, we have two, each with its own random coefficients. The first cancelation will occur for the $\vec{d}_1, \vec{d}_2$ and $\vec{d}_1^*, \vec{d}_2^*$ components, and the second will occur for the $\vec{d}_3, \vec{d}_4$ and $\vec{d}_3^*, \vec{d}_4^*$ components.

This expansion gives us room to use the subspace assumption with parameter $k = 2$ to transition from 4-dimensional exponents for normal keys and ciphertexts to 6-dimensional exponents for semi-functional keys and ciphertexts. Having a 2-dimensional semi-functional space allows us to implement nominal semi-functionality. We will elaborate on this below after defining the semi-functional objects for our scheme. To prevent multiplicative manipulations of the identities in our scheme is rather easy, since the orthogonality of the dual bases allows us to "tie" all the components of the keys and ciphertexts together without causing cross interactions that interfere with decryption.

We assume that messages M are elements of G_T (the target group of the bilinear map) and that identities ID are elements of $\mathbb{Z}_p$.

Setup$(\lambda) \to \text{MSK}, \text{PP}$ The setup algorithm takes in the security parameter λ and chooses a bilinear group G of sufficiently large prime order p. We let $e : G \times G \to G_T$ denote the bilinear map. We set $n = 6$. The algorithm samples random dual orthonormal bases, $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} \mathit{Dual}(\mathbb{Z}_p^n)$. We let $\vec{d}_1, \ldots, \vec{d}_6$ denote the elements of $\mathbb{D}$ and $\vec{d}_1^*, \ldots, \vec{d}_6^*$ denote the elements of $\mathbb{D}^*$. It also chooses random values $\alpha, \theta, \sigma \in \mathbb{Z}_p$. The public parameters are computed as:

$$\text{PP} := \left\{ G, p, e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4} \right\}.$$

(We note that $\vec{d}_1 \cdot \vec{d}_1^* = \psi$ by definition of $\mathbb{D}, \mathbb{D}^*$, but we write out the dot product when we feel it is more instructive.) The master secret key is:

$$\text{MSK} := \left\{ g^{\theta \vec{d}_1^*}, g^{\alpha\theta \vec{d}_1^*}, g^{\theta \vec{d}_2^*}, g^{\sigma \vec{d}_3^*}, g^{\sigma \vec{d}_4^*} \right\}.$$

KeyGen$(\text{MSK}, ID) \to \text{SK}_{ID}$ The key generation algorithm chooses random values $r_1, r_2 \in \mathbb{Z}_p$ and forms the secret key as:

$$\text{SK}_{ID} := g^{(\alpha + r_1 ID)\theta \vec{d}_1^* - r_1 \theta \vec{d}_2^* + r_2 ID \sigma \vec{d}_3^* - r_2 \sigma \vec{d}_4^*}.$$

Encrypt$(M, ID, \text{PP}) \to \text{CT}$ The encryption algorithm chooses random values $s_1, s_2 \in \mathbb{Z}_p$ and forms the ciphertext as:

$$\text{CT} := \left\{ C_1 := M \left(e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^*} \right)^{s_1}, \;\; C_2 := g^{s_1 \vec{d}_1 + s_1 ID \vec{d}_2 + s_2 \vec{d}_3 + s_2 ID \vec{d}_4} \right\}.$$

Decrypt$(\text{CT}, \text{SK}_{ID}) \to M$ The decryption algorithm computes the message as:

$$M := C_1 / e_n(\text{SK}_{ID}, C_2).$$

Recall that $n = 6$, so this requires six pairings.

4.4 Correctness

We observe that when the ciphertext is encrypted under ID, then:

$$e_n(\mathrm{SK}_{ID}, C_2) = e(g,g)^{s_1(\alpha + r_1 ID)\theta \vec{d}_1 \cdot \vec{d}_1^* - s_1 ID r_1 \theta \vec{d}_2 \cdot \vec{d}_2^* + s_2 r_2 ID \sigma \vec{d}_3 \cdot \vec{d}_3^* - s_2 ID r_2 \sigma \vec{d}_4 \cdot \vec{d}_4^*}.$$

Since $\vec{d}_1 \cdot \vec{d}_1^* = \vec{d}_2 \cdot \vec{d}_2^* = \vec{d}_3 \cdot \vec{d}_3^* = \vec{d}_4 \cdot \vec{d}_4^* = \psi$, this exponent is equal to:

$$(s_1\alpha\theta + s_1 r_1 ID\theta - s_1 r_1 ID\theta + s_2 r_2 ID\sigma - s_2 r_2 ID\sigma)\psi = s_1\alpha\theta\psi.$$

Noting that

$$C_1 = Me(g,g)^{s_1\alpha\theta\psi},$$

correctness follows.

A visual representation of the structure of our construction is below in Figure 3. In our illustration, we leave out the α contribution as well as the θ and σ parameters in order to get an uncluttered look at the core structure of the cancelations that occur in our decryption algorithm. We indicate the semi-functional space to be the span of the vectors $\vec{d}_5, \vec{d}_6$ for ciphertexts and the span of the vectors $\vec{d}_5^*, \vec{d}_6^*$ for keys. Semi-functional ciphertexts and keys will include random vectors in these spaces. These are formally defined in the next subsection.

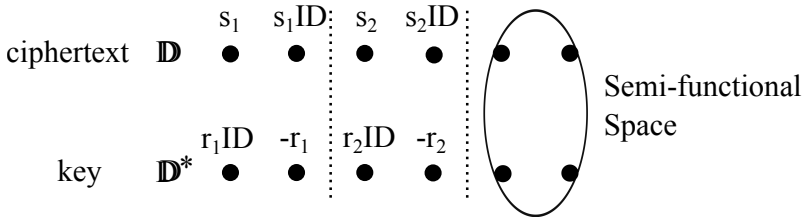

Figure 3: Cancelation in our construction

4.5 Semi-functional Algorithms

We choose to define our semi-functional objects by providing algorithms that generate them. We note that these algorithms are only provided for definitional purposes, and are not part of the IBE system. In particular, they do not need to be efficiently computable from the public parameters and master secret key alone.

KeyGenSF The semi-functional key generation algorithm chooses random values $r_1, r_2, t_5, t_6 \in \mathbb{Z}_p$ and forms the secret key as

$$\mathrm{SK}_{ID} := g^{(\alpha + r_1 ID)\theta \vec{d}_1^* - r_1 \theta \vec{d}_2^* + r_2 ID \sigma \vec{d}_3^* - r_2 \sigma \vec{d}_4^* + t_5 \vec{d}_5^* + t_6 \vec{d}_6^*}.$$

This is distributed like a normal key with additional random multiples of $\vec{d}_5^*$ and $\vec{d}_6^*$ added in the exponent.

EncryptSF The semi-functional encryption algorithm chooses random values $s_1, s_2, z_5, z_6 \in \mathbb{Z}_p$ and forms the ciphertext as:

$$\mathrm{CT} := \left\{ C_1 := M\left(e(g,g)^{\alpha\theta\vec{d}_1 \cdot \vec{d}_1^*}\right)^{s_1},\ C_2 := g^{s_1\vec{d}_1 + s_1 ID \vec{d}_2 + s_2 \vec{d}_3 + s_2 ID \vec{d}_4 + z_5 \vec{d}_5 + z_6 \vec{d}_6} \right\}.$$

This is distributed like a normal ciphertext with additional random multiples of $\vec{d_5}$ and $\vec{d_6}$ added in the exponent.

We observe that if one applies the decryption procedure with a semi-functional key and a normal ciphertext, decryption will succeed because $\vec{d_5^*}, \vec{d_6^*}$ are orthogonal to all of the vectors in exponent of C_2, and hence have no effect on decryption. Similarly, decryption of a semi-functional ciphertext by a normal key will also succeed because $\vec{d_5}, \vec{d_6}$ are orthogonal to all of the vectors in the exponent of the key. When *both* the ciphertext and key are semi-functional, the result of $e_n(\mathrm{SK}_{ID}, C_2)$ will have an additional term, namely

$$e(g,g)^{t_5 z_5 \vec{d_5} \cdot \vec{d_5^*} + t_6 z_6 \vec{d_6} \cdot \vec{d_6^*}} = e(g,g)^{(t_5 z_5 + t_6 z_6)\psi}.$$

Decryption will then fail unless $t_5 z_5 + t_6 z_6 \equiv 0 \bmod p$. If this modular equation holds, we say that the key and ciphertext pair is *nominally semi-functional*. We note that this is possible, even when none of t_5, z_5, t_6, z_6 are congruent to zero modulo p - this is why we have designated a semi-functional space of dimension 2. Requiring the 1-dimensional version of this modular equation, i.e. $t_5 z_5 \equiv 0 \bmod p$, would be equivalent to requiring that either t_5 or z_5 be congruent to zero modulo p.

4.6 Proof of Security

We now prove the following theorem:

Theorem 7. *Under the decisional linear assumption, the IBE scheme presented in Section 4.3 is fully secure.*

We prove this using a hybrid argument over a sequence of games, following the LW strategy. We start with the real security game, denoted by Game_{real}. We let q denote the number of keys requested by the attacker. We define the following additional games.

Game_i for $i = 0, 1, \ldots, q$ Game_i is like Game_{real}, except the ciphertext given to the attacker is semi-functional (i.e. generated by a call to EncryptSF instead of Encrypt) and the first i keys given to the attacker are semi-functional (generated by KeyGenSF). The remaining keys are normal. We note that in Game_0, all of the keys are normal, and in Game_q, all of the keys are semi-functional.

Game_{final} Game_{final} is like Game_q, except that the ciphertext is a semi-functional encryption of a *random* message in G_T, instead of one of the messages supplied by the attacker.

We transition from Game_{real} to Game_0, then to Game_1, and so on, until we arrive at Game_q. We prove that with each transition, the attacker's advantage cannot change by a non-negligible amount. As a last step, we transition to Game_{final}, where it is clear that the attacker's advantage is zero. These transitions are accomplished in the following lemmas, all using the subspace assumption. We let $Adv_{\mathcal{A}}^{real}$ denote the advantage of an algorithm $\mathcal{A}$ in the real game, $Adv_{\mathcal{A}}^{i}$ denote its advantage in Game_i, and $Adv_{\mathcal{A}}^{final}$ denote its advantage in Game_{final}.

We begin with the transition from Game_{real} to Game_0. At the analogous step in the LW proof, a subgroup decision assumption is used to expand the ciphertext from G_{p_1} into $G_{p_1 p_2}$. Here, we use the subspace assumption with $k = 2$ to expand the ciphertext exponent vector from the span of $\vec{d_1}, \ldots, \vec{d_4}$ into the larger span of $\vec{d_1}, \ldots, \vec{d_6}$. We use a very basic instance of the parameter hiding technique to argue that the resulting coefficients of $\vec{d_5}$ and $\vec{d_6}$ are randomly distributed: this is done by initially embedding a random 2×2 change of basis matrix A into our setting of the basis vectors $\vec{d_5}, \vec{d_6}$.

A visual representation of the use of the subspace assumption in this step is provided in the following figure.

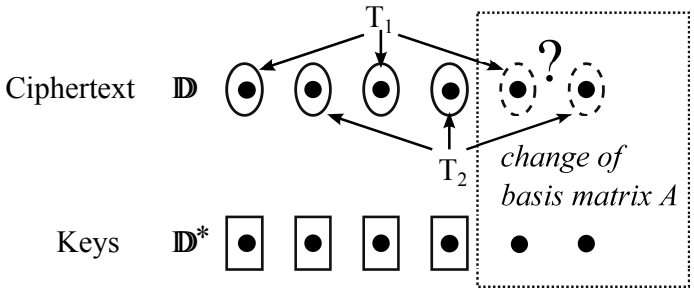

Figure 4: Ciphertext expands into semi-functional space

In Figure 4, solid ovals and rectangles denote the definite presence of vectors in the exponents, and the dotted ovals denote elements that may or may not be present. We have illustrated T_1, T_2 as for the basis $\mathbb{B}$, and we note that $\mathbb{D}$ is obtained from $\mathbb{B}$ by applying the change of basis matrix A to the last two basis vectors.

Lemma 8. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{real}_{\mathcal{A}} - Adv^{0}_{\mathcal{A}}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 6$.*

Proof. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4}, g^{\vec{b}^*_5}, g^{\vec{b}^*_6}, U_1, U_2, \mu_3\right),$$

along with T_1, T_2. It is $\mathcal{B}$'s task to decide whether T_1, T_2 are distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6}$. In this proof, we will not need to use the terms $g^{\vec{b}^*_5}, g^{\vec{b}^*_6}, U_1, U_2, \mu_3$, so these can be ignored.

$\mathcal{B}$ will simulate either Game_{real} or Game_0 with $\mathcal{A}$, depending on the distribution of T_1, T_2. To compute the public parameters and master secret key, $\mathcal{B}$ first chooses a random matrix $A \in \mathbb{Z}_p^{2\times 2}$ (with all but negligible probability, A is invertible). We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ by:

$$\vec{f}_1 = \eta\vec{b}^*_1,\ \vec{f}_2 = \eta\vec{b}^*_2,\ \vec{f}_3 = \beta\vec{b}^*_3,\ \vec{f}_4 = \beta\vec{b}^*_4,\ \vec{f}_5 = \vec{b}^*_5,\ \vec{f}_6 = \vec{b}^*_6,$$

$$\vec{f}^*_1 = \eta^{-1}\vec{b}_1,\ \vec{f}^*_2 = \eta^{-1}\vec{b}_2,\ \vec{f}^*_3 = \beta^{-1}\vec{b}_3,\ \vec{f}^*_4 = \beta^{-1}\vec{b}_4,\ \vec{f}^*_5 = \vec{b}_5,\ \vec{f}^*_6 = \vec{b}_6.$$

Now $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{F}_A, \mathbb{D}^* = \mathbb{F}^*_A$, where A is applied as a change of basis matrix to $\vec{f}_5, \vec{f}_6$ and $(A^{-1})^t$ is applied as a change of basis matrix to $\vec{f}^*_5, \vec{f}^*_6$, as described in Section 3.1. We note that for $i = 1, \ldots, 4$, $\vec{d}_i = \vec{f}_i$ and $\vec{d}^*_i = \vec{f}^*_i$. We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed, and reveal no information about A. This follows from Lemma 3, since $\mathbb{F}, \mathbb{F}^*$ are still distributed as a random pair of dual orthonormal bases.

$\mathcal{B}$ chooses random values $\alpha, \theta', \sigma' \in \mathbb{Z}_p$, and implicitly sets $\theta = \theta'\eta$, $\sigma = \sigma'\beta$. We note that $\mathcal{B}$ can compute $e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}$ as $\left(e_n(g^{\vec{b}_1}, g^{\eta\vec{b}^*_1})\right)^{\alpha\theta'}$. $\mathcal{B}$ can also produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}$, and $g^{\theta\vec{d}^*_1}, g^{\theta\vec{d}^*_2}, g^{\sigma\vec{d}^*_3}, g^{\sigma\vec{d}^*_4}$. Though $\mathcal{B}$ *cannot* produce $g^{\vec{d}^*_5}$ or $g^{\vec{d}^*_6}$, these will not be needed for creating normal keys.

$\mathcal{B}$ gives $\mathcal{A}$ the public parameters:

$$\text{PP} := \left\{G, p, e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}\right\}.$$

The master secret key,

$$\text{MSK} := \left\{ g^{\theta \vec{d}_1^*}, g^{\alpha\theta \vec{d}_1^*}, g^{\theta \vec{d}_2^*}, g^{\sigma \vec{d}_3^*}, g^{\sigma \vec{d}_4^*} \right\},$$

is known to $\mathcal{B}$. This allows $\mathcal{B}$ to respond to all of $\mathcal{A}$'s key queries by calling the normal key generation algorithm, and giving the resulting keys to $\mathcal{A}$.

At some point, $\mathcal{A}$ submits two messages, M_0 and M_1, along with a challenge identity, ID^*. $\mathcal{B}$ chooses a random bit $b \in \{0, 1\}$ and encrypts M_b as follows. It sets:

$$C_2 := T_1 (T_2)^{ID^*}.$$

This implicitly sets $s_1 = \tau_1$ and $s_2 = \tau_2$. It also computes:

$$C_1 := M_b \left(e_n(T_1, g^{\vec{b}_1}) \right)^{\theta' \alpha} = M_b \left(e(g,g)^{\alpha\theta \vec{d}_1 \cdot \vec{d}_1^*} \right)^{s_1}.$$

It gives the ciphertext $\text{CT} = \{C_1, C_2\}$ to $\mathcal{A}$.

Now, if T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^*}$, then this is a properly distributed normal encryption of M_b. In this case, $\mathcal{B}$ has properly simulated Game_{real}. If T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^* + \tau_3 \vec{b}_5^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^* + \tau_3 \vec{b}_6^*}$ instead, then the ciphertext element C_2 has an additional term of

$$\tau_3 \vec{b}_5^* + ID^* \tau_3 \vec{b}_6^* \tag{1}$$

in its exponent. The coefficients here in the basis $\vec{b}_5^*, \vec{b}_6^*$ form the vector $(\tau_3, ID^* \tau_3)$. To compute the coefficients in the basis $\vec{d}_5, \vec{d}_6$, we multiply the matrix A^{-1} by the transpose of this vector, obtaining $\tau_3 A^{-1} (1, ID^*)^t$. Since A is random (everything else given to $\mathcal{A}$ has been distributed independently of A), these coefficients are uniformly random. Therefore, in this case, $\mathcal{B}$ has properly simulated Game_0. This allows $\mathcal{B}$ to leverage $\mathcal{A}$'s non-negligible difference in advantage between Game_{real} and Game_0 to achieve a non-negligible advantage against the subspace assumption. □

We now handle the transition from Game_{i-1} to Game_i. At this step in the LW proof, a subgroup decision assumption is used to expand the i^{th} secret key from $G_{p_1 p_3}$ into $G = G_{p_1 p_2 p_3}$. Analogously, we will use the subspace assumption to expand the i^{th} secret key exponent vector from the span of $\vec{d}_1^*, \ldots, \vec{d}_4^*$ into the larger span of $\vec{d}_1^*, \ldots, \vec{d}_6^*$. We will embed a 2×2 change of basis matrix A and set $\mathbb{D} = \mathbb{B}_A$ and $\mathbb{D}^* = \mathbb{B}_A^*$, where A is applied to $\vec{b}_5, \vec{b}_6$ to form $\vec{d}_5, \vec{d}_6$. As in the LW proof, we cannot be given an object that resides solely in the semi-functional space of the ciphertext (e.g. we cannot be given $g^{\vec{d}_5}, g^{\vec{d}_6}$), but we are given objects that have semi-functional components attached to normal components, and we can use these to create the semi-functional ciphertext. In the LW proof, a term in $G_{p_1 p_2}$ in used. Here, an exponent vector that is a linear combination of $\vec{b}_1, \vec{b}_3, \vec{b}_5$ and another exponent vector that is a linear combination of $\vec{b}_2, \vec{b}_4, \vec{b}_6$ are used. In our case, making the other normal and semi-functional keys is straightforward, since we are given scalar multiples of all of the vectors of $\mathbb{D}^*$ in the exponent. We use the fact that the matrix A is hidden from the attacker in order to argue that the semi-functional parts of the ciphertext and i^{th} key appear well-distributed.

The following diagram illustrates how the subspace assumption is used in this transition.

In Figure 5, solid ovals and rectangles indicate basis vectors which are definitely present in the exponents, while the dashed ovals indicate basis vectors which may or may not be present. The X's indicate basis vectors which the simulator does not have by themselves in the exponent, but only has access to them as attached to other vectors. We have illustrated T_1, T_2, U_1, U_2 as for bases $\mathbb{B}, \mathbb{B}^*$, and we note that $\mathbb{D}, \mathbb{D}^*$ are obtained from $\mathbb{B}, \mathbb{B}^*$ by applying the change of basis matrix A to the last two vectors of $\mathbb{B}$ and applying $(A^{-1})^t$ to the last two vectors of $\mathbb{B}^*$.

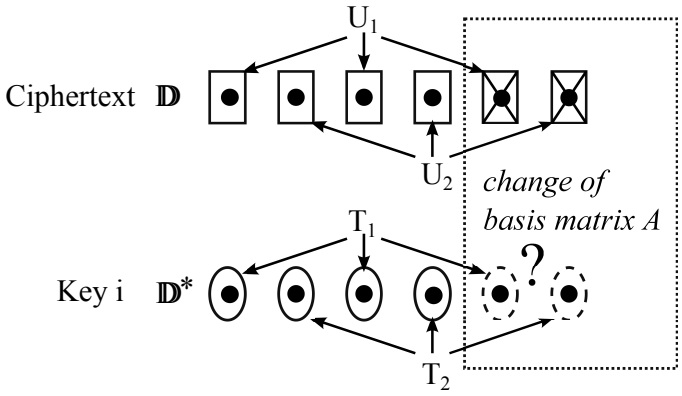

Figure 5: Key i expands into semi-functional space

Lemma 9. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv^i_{\mathcal{A}} - Adv^{i-1}_{\mathcal{A}}$ is non-negligible for some $i = 1, \ldots, q$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 6$.*

Proof. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4}, g^{\vec{b}^*_5}, g^{\vec{b}^*_6}, U_1, U_2, \mu_3\right),$$

along with T_1, T_2. It is $\mathcal{B}$'s task to decide whether T_1, T_2 are distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6}$. (We note that knowledge of μ_3 will not be needed by $\mathcal{B}$ in this proof.)

$\mathcal{B}$ will simulate either Game_i or Game_{i-1} with $\mathcal{A}$, depending on the distribution of T_1, T_2. $\mathcal{B}$ chooses a random $A \in \mathbb{Z}_p^{2\times 2}$ (with all but negligible probability, A is invertible). $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{B}_A$ and $\mathbb{D} = \mathbb{B}^*_A$, where the change of basis matrix A is applied to $\vec{b}_5, \vec{b}_6$ and the change of basis matrix $(A^{-1})^t$ is applied to $\vec{b}^*_5, \vec{b}^*_6$, as described in Section 3.1. Note that the first four basis vectors are unchanged:

$$\vec{d}_1 = \vec{b}_1, \ \ldots, \ \vec{d}_4 = \vec{b}_4,$$

$$\vec{d}^*_1 = \vec{b}^*_1, \ \ldots, \ \vec{d}^*_4 = \vec{b}^*_4.$$

We note that $\mathbb{D}, \mathbb{D}^*$ are well-distributed, and reveal no information about A (by Lemma 3). $\mathcal{B}$ can produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}$. $\mathcal{B}$ also implicitly sets $\theta = \eta$ and $\sigma = \beta$. We note that this allows it to produce $g^{\theta\vec{d}^*_1} = g^{\eta\vec{b}^*_1}$, and similarly for $g^{\theta\vec{d}^*_2}, g^{\sigma\vec{d}^*_3}, g^{\sigma\vec{d}^*_4}$. It chooses a random value $\alpha \in \mathbb{Z}_p$ for itself, enabling it to compute the public parameters and master secret key as:

$$\text{PP} := \left\{G, p, e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1} = \left(e_n(g^{\vec{b}_1}, g^{\eta\vec{b}^*_1})\right)^{\alpha}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}\right\}$$

$$\text{MSK} := \left\{g^{\theta\vec{d}^*_1}, \ \left(g^{\theta\vec{d}^*_1}\right)^{\alpha}, \ g^{\theta\vec{d}^*_2}, \ g^{\sigma\vec{d}^*_3}, \ g^{\sigma\vec{d}^*_4}\right\}.$$

$\mathcal{B}$ can then produce normal keys by running the normal key generation algorithm. Since $\mathcal{B}$ also knows $g^{\vec{b}^*_5}$ and $g^{\vec{b}^*_6}$, it can easily produce semi-functional keys. More precisely, it can create random linear combinations of $g^{\vec{d}^*_5}$ and $g^{\vec{d}^*_6}$ in the exponent by taking random combinations of $\vec{b}^*_5$ and $\vec{b}^*_6$. This is equivalent because the span of $\vec{d}^*_5$ and $\vec{d}^*_6$ is equal to the span of $\vec{b}^*_5$ and $\vec{b}^*_6$.

$\mathcal{B}$ gives PP to $\mathcal{A}$. To answer the first $i-1$ key queries that $\mathcal{A}$ makes, $\mathcal{B}$ runs the semi-functional key generation algorithm to produce semi-functional keys and gives these to $\mathcal{A}$. When $\mathcal{A}$ makes the i^{th} query for identity ID_i, $\mathcal{B}$ responds with:

$$SK_{ID_i} := \left(g^{\eta \vec{b}_1^*}\right)^{\alpha} T_1^{ID_i} \left(T_2\right)^{-1}.$$

This implicitly sets $r_1 = \tau_1$ and $r_2 = \tau_2$. If T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^*}$, then this is a properly distributed normal key. If T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^* + \tau_3 \vec{b}_5^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^* + \tau_3 \vec{b}_6^*}$, then this is a semi-functional key, whose exponent vector includes

$$ID_i \tau_3 \vec{b}_5^* - \tau_3 \vec{b}_6^* \tag{2}$$

as its component in the span of $\vec{b}_5^*, \vec{b}_6^*$. To respond to the remaining key queries, $\mathcal{B}$ simply runs the normal key generation algorithm.

At some point, $\mathcal{A}$ submits two messages, M_0 and M_1, along with a challenge identity, ID^*. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and produces a semi-functional encryption of M_b as follows. It sets:

$$C_2 = U_1 \left(U_2\right)^{ID^*}.$$

This implicitly sets $s_1 = \mu_1$ and $s_2 = \mu_2$. The "semi-functional part" of the exponent vector here (i.e. the part in the span of $\vec{d}_5 = \vec{b}_5$ and $\vec{d}_6 = \vec{b}_6$) is:

$$\mu_3 \vec{b}_5 + ID^* \mu_3 \vec{b}_6. \tag{3}$$

We observe that if $ID^* = ID_i$ (which is not allowed), then the vectors (2) and (3) would be orthogonal, resulting in a nominally semi-functional ciphertext and key pair. The other element of the ciphertext is formed as:

$$C_1 = M_b \left(g^{\eta \vec{b}_1^*}, U_1\right)^{\alpha} = M_b \left(e(g,g)^{\alpha \theta \vec{d}_1^*}\right)^{s_1}.$$

The ciphertext $\mathrm{CT} = \{C_1, C_2\}$ is given to $\mathcal{A}$.

We now argue that since $ID^* \neq ID_i$, in $\mathcal{A}$'s view the vectors (2) and (3) are distributed as random vectors in the spans of $\{\vec{d}_5^*, \vec{d}_6^*\}$ and $\{\vec{d}_5, \vec{d}_6\}$ respectively. To see this, we take the coefficients of vectors (2) and (3) in terms of the bases $\vec{b}_5^*, \vec{b}_6^*$ and $\vec{b}_5, \vec{b}_6$ respectively and translate them into coefficients in terms of the bases $\vec{d}_5^*, \vec{d}_6^*$ and $\vec{d}_5, \vec{d}_6$. Using the change of basis matrix A, we obtain the new coefficients (in vector form) as:

$$\tau_3 A^t (ID_i, -1)^t,\ \mu_3 A^{-1} (1, ID^*).$$

Since the distribution of everything given to $\mathcal{A}$ *except* for the i^{th} key and the challenge ciphertext is independent of the random matrix A and $ID^* \neq ID_i$, we can apply Lemma 4 to conclude that these coefficients are uniformly random. Thus, $\mathcal{B}$ has properly simulated Game_i in this case.

In summary, $\mathcal{B}$ has properly simulated either Game_{i-1} or Game_i for $\mathcal{A}$, depending on the distribution of T_1, T_2. It can therefore leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to obtain a non-negligible advantage against the subspace assumption.

□

The final step of the LW proof uses an assumption that it is not technically an instance of the general subgroup decision assumption, but is of a similar flavor. Namely, it is assumed that

given $g_1, g_2, g_3, g_1^{\alpha} X_2, g_1^{s} Y_2$, it is hard to distinguish $e(g_1, g_1)^{\alpha s}$ from a random element of G_T. Here, g_1, g_2, g_3 are randomly chosen generators of $G_{p_1}, G_{p_2}, G_{p_3}$ respectively, α, s are randomly chosen from $\mathbb{Z}_N$, and X_2, Y_2 are randomly chosen elements of G_{p_2}. The term $g_1^{\alpha} X_2$ is used to make semi-functional keys, $g_1^{s} Y_2$ is used to make semi-functional ciphertexts, and T is used to blind the message. When $T = e(g_1, g_1)^{\alpha s}$, this yields a properly distributed semi-functional ciphertext. When T is random, this yields a properly distributed semi-functional encryption of a random message.

In our case, we use a slightly different strategy. Instead of enacting a change directly on the blinding factor of the message, we use the subspace assumption with $k = 1$ twice to randomize each appearance of s_1 in the C_2 term of the ciphertext, thereby severing its link with the blinding factor. The end result is the same - we obtain a semi-functional encryption of a random message. This randomization of s_1 is accomplished by first expanding an exponent vector from the span of $\vec{d_5}, \vec{d_6}$ into the larger span of $\vec{d_5}, \vec{d_6}, \vec{d_2}$ and then expanding an exponent vector from the span of $\vec{d_5}, \vec{d_6}$ into the larger span of $\vec{d_5}, \vec{d_6}, \vec{d_1}$. We note that the knowledge of the μ_3 value in the subspace assumption is used here to ensure that while we are doing the first expansion, for example, we can make the two occurrences of r_1 in the keys match consistently (this is necessary because $g^{\vec{d_2^*}}$ by itself will not be known during this step).

The following diagram illustrates how the subspace assumption is used for the first of these two steps (the second step is similar).

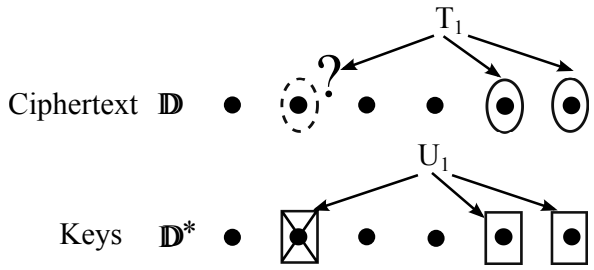

Figure 6: The coefficient of $\vec{d_2}$ in the ciphertext is randomized

In Figure 6, we only illustrate how T_1 and U_1 affect the keys and ciphertext, and we neglect many other terms. One should also note that U_1 will be raised to a fresh random power for each key - a subtlety which our diagram does not capture.

Lemma 10. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q} - Adv_{\mathcal{A}}^{final}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 6$.*

We prove this lemma in two steps. As a first step, we consider an intermediary game, called Game_q':

Game_q' This is exactly like Game_q, except that in the C_2 term of the challenge ciphertext, the coefficient of $\vec{d_2}$ is changed from being $s_1 ID^*$ to a fresh random value in $\mathbb{Z}_p$. We denote the advantage of an algorithm $\mathcal{A}$ in this game by $Adv_{\mathcal{A}}^{q'}$.

We first prove:

Lemma 11. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{q} - Adv_{\mathcal{A}}^{q'}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 6$.*

Proof. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, g^{\vec{b}_5}, g^{\vec{b}_6}, g^{\eta\vec{b}^*_1}, g^{\beta\vec{b}^*_2}, g^{\vec{b}^*_3}, g^{\vec{b}^*_4}, g^{\vec{b}^*_5}, g^{\vec{b}^*_6}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$.

To define the public parameters, $\mathcal{B}$ implicitly sets:

$$\vec{d}_1 = \vec{b}^*_6,\ \vec{d}_2 = \vec{b}^*_3,\ \vec{d}_3 = \vec{b}^*_5,\ \vec{d}_4 = \vec{b}^*_4,\ \vec{d}_5 = \vec{b}^*_2, \vec{d}_6 = \vec{b}^*_1.$$

We note that this enables $\mathcal{B}$ to produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}$ for the public parameters. $\mathcal{B}$ additionally sets:

$$\vec{d}^*_1 = \vec{b}_6,\ \vec{d}^*_2 = \vec{b}_3,\ \vec{d}^*_3 = \vec{b}_5,\ \vec{d}^*_4 = \vec{b}_4,\ \vec{d}^*_5 = \vec{b}_2,\ \vec{d}^*_6 = \vec{b}_1.$$

This ensures that $\mathbb{D}, \mathbb{D}^*$ are properly distributed dual orthonormal bases. We note that $\mathcal{B}$ can produce $g^{\vec{d}^*_1}, g^{\vec{d}^*_3}, \ldots, g^{\vec{d}^*_6}$, but *does not know* $g^{\vec{d}^*_2}$.

$\mathcal{B}$ chooses random values $\theta, \sigma, \alpha \in \mathbb{Z}_p$ for itself. It can compute $e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}$ as $\left(e_n(g^{\vec{b}^*_6}, g^{\vec{b}_6})\right)^{\alpha\theta}$. It gives $\mathcal{A}$ the public parameters:

$$\text{PP} := \left\{G, p, e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}\right\}.$$

We note that $\mathcal{B}$ does not know the full master secret key, because it does not know $g^{\vec{d}^*_2} = g^{\vec{b}_3}$. It does know U_1 and μ_3, however, where $U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3}$. This will allow it to produce semi-functional keys as follows. When $\mathcal{A}$ requests a key for some identity ID, $\mathcal{B}$ chooses random values $r'_1, r_2, t'_5, t'_6 \in \mathbb{Z}_p$. It will set $r_1 = \mu_3 r'_1$. It forms the secret key as:

$$\text{SK}_{ID} := (U_1)^{-\theta r'_1} g^{(\alpha+\mu_3 r'_1 ID)\theta\vec{d}^*_1 + r_2 ID\sigma\vec{d}^*_3 - r_2\sigma\vec{d}^*_4 + t'_5\vec{d}^*_5 + t'_6\vec{d}^*_6}.$$

We note that the coefficient of $\vec{d}^*_2 = \vec{b}_3$ here is equal to $-\mu_3 r'_1\theta = -r_1\theta$, as required. The coefficients of $\vec{d}^*_5 = \vec{b}_2$ and $\vec{d}^*_6 = \vec{b}_1$ are uniformly random (since they are additively randomized by t'_5, t'_6), so this is a properly distributed semi-functional key. The simulator's knowledge of μ_3 was helpful here, in that it allowed the simulator to form the r_1 coefficient for use with $\vec{d}^*_1$ as well as $\vec{d}^*_2$.

At some point, $\mathcal{A}$ submits messages M_0, M_1 and a challenge identity, ID^*. $\mathcal{B}$ samples a random bit $b \in \{0,1\}$ and forms the challenge ciphertext as follows. It chooses $s_1, s_2 \in \mathbb{Z}_p$ randomly. It sets:

$$C_1 := M_b e(g,g)^{\alpha s_1},\ C_2 := g^{s_1\vec{d}_1 + s_1 ID^*\vec{d}_2 + s_2\vec{d}_3 + s_2 ID^*\vec{d}_4} T_1.$$

Now, if $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$, then the exponent vector of T_1 is a random linear combination of $\vec{d}_5$ and $\vec{d}_6$, making this a well-distributed semi-functional ciphertext in Game_q. If the exponent of T_1 additionally has $\tau_3\vec{b}^*_3 = \tau_3\vec{d}_2$, then this randomizes the coefficient of $\vec{d}_2$, yielding a ciphertext distributed as in Game'_q. Therefore, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference of advantage between these two games to achieve a non-negligible advantage against the subspace assumption. □

Lemma 12. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv^{q'}_{\mathcal{A}} - Adv^{final}_{\mathcal{A}}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 6$.*

Proof. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, g^{\vec{b}_5}, g^{\vec{b}_6}, g^{\eta\vec{b}^*_1}, g^{\beta\vec{b}^*_2}, g^{\vec{b}^*_3}, g^{\vec{b}^*_4}, g^{\vec{b}^*_5}, g^{\vec{b}^*_6}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$. We recall that $U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3}$.

To define the public parameters, $\mathcal{B}$ implicitly sets:

$$\vec{d}_1 = \vec{b}^*_3,\ \vec{d}_2 = \vec{b}^*_4,\ \vec{d}_3 = \vec{b}^*_5,\ \vec{d}_4 = \vec{b}^*_6,\ \vec{d}_5 = \vec{b}^*_1,\ \vec{d}_6 = \vec{b}^*_2$$

$$\vec{d}^*_1 = \vec{b}_3,\ \vec{d}^*_2 = \vec{b}_4,\ \vec{d}^*_3 = \vec{b}_5,\ \vec{d}^*_4 = \vec{b}_6,\ \vec{d}^*_5 = \vec{b}_1,\ \vec{d}^*_6 = \vec{b}_2.$$

We note that $\mathbb{D}$ and $\mathbb{D}^*$ are properly distributed dual orthonormal bases, and $\mathcal{B}$ can produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}, g^{\vec{d}^*_2}, \ldots, g^{\vec{d}^*_6}$.

$\mathcal{B}$ chooses α', θ, σ randomly from $\mathbb{Z}_p$. It will implicitly set $\alpha = \alpha'\mu_3$. It can then compute $e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}$ as $\left(e_n(g^{\vec{b}_4}, g^{\vec{b}^*_4})\right)^{\alpha'\mu_3\theta}$, for instance, since $\vec{d}_1 \cdot \vec{d}^*_1 = \vec{b}^*_3 \cdot \vec{b}_3 = \vec{b}^*_4 \cdot \vec{b}_4$. It can then give $\mathcal{A}$ the public parameters:

$$\text{PP} := \left\{G, p, e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_4}\right\}.$$

When $\mathcal{A}$ requests a key for an identity ID, $\mathcal{B}$ responds as follows. It chooses random values r'_1, r_2, t'_5, t'_6 and implicitly sets $r_1 = r'_1\mu_3$. It forms the key as:

$$\text{SK}_{ID} = U_1^{(\alpha'+r'_1 ID)\theta} g^{-r'_1\mu_3\theta\vec{d}^*_2 + r_2 ID\sigma\vec{d}^*_3 - r_2\sigma\vec{d}^*_4 + t'_5\vec{d}^*_5 + t'_6\vec{d}^*_6}.$$

We note that the coefficient of $\vec{d}^*_1 = \vec{b}_3$ here is $\mu_3(\alpha' + r'_1 ID)\theta = (\alpha + r_1 ID)\theta$, as required. Also, the coefficients of $\vec{d}^*_5 = \vec{b}_1$ and $\vec{d}^*_6 = \vec{b}_2$ are uniformly random (since t'_5, t'_6 are random). Thus, $\mathcal{B}$ produces properly distributed semi-functional keys.

When $\mathcal{A}$ submits messages M_0, M_1 and ID^*, $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and encrypts M_b as follows. It chooses s_1, s_2, w randomly from $\mathbb{Z}_p$ and sets the ciphertext as:

$$\left\{C_1 := M_b \left(e(g,g)^{\alpha\theta\vec{d}_1\cdot\vec{d}^*_1}\right)^{s_1},\ C_2 := g^{s_1\vec{d}_1 + w\vec{d}_2 + s_2\vec{d}_3 + s_2 ID\vec{d}_4} T_1\right\}.$$

Now, if $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$, then the exponent vector of T_1 is a random linear combination of $\vec{d}_5$ and $\vec{d}_6$, making this a well-distributed semi-functional ciphertext in Game'_q. If the exponent of T_1 additionally has $\tau_3\vec{b}^*_3 = \tau_3\vec{d}_1$, then this randomizes the coefficient of $\vec{d}_1$, yielding a ciphertext distributed as in Game_{final} (since now the distribution of C_2 is independent of s_1, which makes C_1 a random group element in G_T). Therefore, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference of advantage between these two games to achieve a non-negligible advantage against the subspace assumption. □

Combining Lemmas 11 and 12, we obtain Lemma 10. Along with Lemmas 6, 8, and 9, this completes the proof of Theorem 7.

5 Unbounded HIBE

As a second demonstration of our tools, we consider a variant of the Lewko-Waters unbounded HIBE construction [29]. The composite order construction we present is simpler than the one presented in [29], at the cost of using more subgroups. Since we will ultimately simulate these subgroups in a prime order group, such a cost is no longer a significant detriment.

In designing our prime order translation and proof, we will proceed along a path that is very similar to the path we took to translate the more basic IBE scheme. However, we now must take care to preserve delegation ability throughout our proof. The only place where this becomes non-trivial is in the final step of the proof, where everything is already semi-functional and we want to transition to a random message. Before, we used the subspace assumption to expand from the semi-functional space into the normal space of the ciphertext, and in doing so, we randomized the blinding factor. However, this strategy is problematic when we must enable delegation, since it will result in a simulator who does not know some important basis vectors in the exponent for the normal space of the keys. This seems to prevent the simulator from having what it needs to equip users with delegation capabilities.

To avoid this issue, we employ a different strategy for the final step of the proof. Instead of expanding the semi-functional space into the normal space to randomize the blinding factor, we expand the part of the normal space involved in forming the blinding factor into the semi-functional space. Essentially, this moves us from a blinding factor determined from the exponents in the normal space to a blinding factor that is additionally affected by some terms embedded in the semi-functional space. Our blinding factor can now be seen as random, since the exponents in the semi-functional space can be re-randomized independently of these embedded terms.

The details of our composite order construction, its prime order translation, and security proofs in both settings can be found in Appendix B.

6 Further Discussion

In applying our tools to the both IBE and unbounded HIBE applications, we see that there is some flexibility in how we choose the construction, organize the hybrid games, and embed the subspace assumption in our reductions. All of these considerations interact, allowing us to make tradeoffs. For example, we were able to make our construction more compact in the IBE case and our final proof step simpler, but our way of embedding the subspace assumption in the final stage was problematic for applications requiring delegation of keys. This was easily solved by expanding our construction and hybrid sequence a bit and embedding the subspace assumption in the final stages in a different way in the HIBE setting.

The amount of flexibility available in applying our tools make them suitably versatile to handle a wider variety of applications as well. In particular, they can be applied in the attribute-based encryption setting. We suspect that applying our techniques to the composite order ABE constructions in [25] would result in a system and proof quite similar to the functional encryption schemes presented by Okamoto and Takashima in [33], who obtain security from the decisional linear assumption through dual pairing vector spaces. As in [33], one could obtain small universe ABE by having a distinct pair of dual orthornomal bases associated with each attribute. Each such pair would have constant dimension, with room for both a normal space and a semi-functional space. The proof would proceed by first expanding the ciphertext attributes all into their respective semi-functional spaces, and then expanding the secret keys one by one. We would argue that the coefficients of the semi-functional basis vectors in each key appear to share a truly random vector, even when one is constructed to be nominal, meaning that it actually shares zero. This would follow from the observation that for attributes which the ciphertext does not contain, the corresponding bases vectors of the semi-functional space on the ciphertext side are hidden, and therefore a hidden change of basis matrix makes the corresponding coefficients in the associated semi-functional space of the key appear random. Since the attacker cannot ask for a key capable of decrypting the challenge ciphertext, this hides enough shares so that the shared value in the semi-functional spaces appears random.

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] M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In *TCC*, 2011.

[4] 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.

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

[6] D. Boneh and X. Boyen. Secure identity based encryption without random oracles. In *CRYPTO*, pages 443–459, 2004.

[7] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *EUROCRYPT*, pages 440–456, 2005.

[8] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. In *CRYPTO*, pages 213–229, 2001.

[9] D. Boneh, E. Goh, and K. Nissim. Evaluating 2-dnf formulas on ciphertexts. In *TCC*, pages 325–342, 2005.

[10] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. In *EUROCRYPT*, pages 573–592, 2006.

[11] R. Canetti, S. Halevi, and J. Katz. A forward-secure public-key encryption scheme. In *EUROCRYPT*, pages 255–271, 2003.

[12] D. Cash, D. Hofheinz, E. Kiltz, and C. Peikert. Bonsai trees, or how to delegate a lattice basis. In *EUROCRYPT*, pages 523–552, 2010.

[13] 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.

[14] D. M. Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.

[15] S. Garg, A. Kumarasubramanian, A. Sahai, and B. Waters. Building efficient fully collusion-resilient traitor tracing and revocation schemes. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2010.

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

[17] C. Gentry and S. Halevi. Hierarchical identity based encryption with polynomially many levels. In *TCC*, pages 437–456, 2009.

[18] 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.

[19] C. Gentry and A. Silverberg. Hierarchical id-based cryptography. In *ASIACRYPT*, pages 548–566, 2002.

[20] 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.

[21] J. Groth, R. Ostrovsky, and A. Sahai. Non-interactive zaps and new techniques for nizk. In *CRYPTO*, pages 97–111, 2006.

[22] J. Groth, R. Ostrovsky, and A. Sahai. Perfect non-interactive zero knowledge for np. In *EUROCRYPT*, pages 339–358, 2006.

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

[24] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.

[25] 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.

[26] A. Lewko, Y. Rouselakis, and B. Waters. Achieving leakage resilience through dual system encryption. In *TCC*, pages 70–88, 2011.

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

[28] A. Lewko and B. Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.

[29] A. Lewko and B. Waters. Unbounded hibe and attribute-based encryption. In *EUROCRYPT*, pages 547–567, 2011.

[30] S. Meiklejohn, H. Shacham, and D. M. Freeman. Limitations on transformations from composite-order to prime-order groups: The case of round-optimal blind signatures. In *ASIACRYPT*, pages 519–538, 2010.

[31] T. Okamoto and K. Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008.

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

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

[34] 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.

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

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

[37] E. Shi and B. Waters. Delegating capabilities in predicate encryption systems. In *Automata, Languages and Programming*, volume 5126 of *LNCS*, pages 560–578. Springer, 2008.

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

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

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

A Standard Definitions for IBE and HIBE

A.1 Identity-Based Encryption

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

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

KeyGen$(\text{MSK}, ID) \rightarrow \text{SK}_{ID}$ The key generation algorithm takes in the master secret key and an identity ID and produces a secret key SK_{ID} for that identity.

Encrypt$(\text{PP}, M, ID) \rightarrow \text{CT}$ The encryption algorithm takes in the public parameters PP, a message M, and an identity ID, and outputs a ciphertext CT encrypted under that identity.

Decrypt$(\text{CT}, \text{SK}_{ID}) \rightarrow M$ The decryption algorithm takes in a ciphertext CT and a secret key SK_{ID} and outputs the message M when the CT is encrypted under the same ID.

A.1.1 Security Definition

Security is defined by the following game, played by a challenger and an attacker.

Setup The challenger runs the Setup algorithm to generate PP and MSK. It gives PP to the attacker.

Phase 1 The attacker requests keys for identities ID, and is provided with corresponding secret keys SK_{ID}, which the challenger generates by running the key generation algorithm.

Challenge The attacker gives the challenger two messages M_0 and M_1 and a challenge identity ID^*. This identity must not have been queried in Phase 1. The challenger sets $b \in \{0, 1\}$ randomly, and encrypts M_b under ID^* by running the encryption algorithm. It sends the ciphertext to the attacker.

Phase 2 This is the same as Phase 1, with the added restriction a secret key for ID^* cannot be requested.

Guess The attacker must output a guess b' for b.

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

Definition 13. *An identity-based encryption scheme is secure if all polynomial time attackers achieve at most a negligible advantage in the above security game.*

This is the full, IND-CPA definition of security. The weaker notion of selective security is defined similarly, except that the attacker must declare ID^* at the beginning of the game, before seeing PP.

A.2 Hierarchical Identity-Based Encryption

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

Setup$(\lambda) \to \text{PP}, \text{MSK}$ The setup algorithm takes in the security parameter λ and outputs the public parameters PP and the master secret key MSK.

KenGen$(\text{MSK}, (ID_1, \ldots, ID_j)) \to \text{SK}$ The key generation algorithm takes in the master secret key and an identity vector $(ID_1, \ldots, ID_j)$ and outputs a private key SK for that identity vector.

Delegate$(\text{PP}, \text{SK}, ID_{j+1}) \to \text{SK}'$ The delegation algorithm takes in a secret key for the identity vector $(ID_1, \ldots, ID_j)$ and an identity component ID_{j+1} and outputs a secret key SK$'$ for the identity vector $(ID_1, \ldots, ID_{j+1})$.

Encrypt$(\text{PP}, M, (ID_1, \ldots, ID_j)) \to \text{CT}$ The encryption algorithm takes in the public parameters PP, a message M, and an identity vector $(ID_1, \ldots, ID_j)$ and outputs a ciphertext CT.

Decrypt$(\text{CT}, \text{SK}) \to M$ The decryption algorithm takes in a ciphertext CT and a secret key SK and outputs the message M, if secret key is for an identity vector which is a prefix of the ciphertext identity vector.

We could alternatively only require the decryption algorithm to work when the identity vector for the ciphertext matches the secret key exactly. In this case, someone who had a secret key for a prefix of this identity vector could delegate to themselves the required secret key and still decrypt.

A.2.1 Security definition

We give the complete form of the security definition [37] which keeps track of how keys are generated and delegated. Security is defined by the following game, played by a challenger and an attacker.

Setup The challenger runs the Setup algorithm to generate PP and MSK. It gives PP to the attacker. We let S denote the set of private keys that the challenger has created but not yet given to the attacker. Initially, $S = \emptyset$.

Phase 1 The attacker makes Create, Delegate, and Reveal key queries. To make a Create query, the attacker specifies an identity vector. 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 in the set S for identity vector $(ID_1, \ldots, ID_j)$ and an identity component ID_{j+1}. In response, the challenger makes a key for this new identity vector $(ID_1, \ldots, ID_{j+1})$ by running the delegation algorithm. It adds this delegated 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 $(ID^*_1, \ldots, ID^*_{j^*})$. This identity vector must satisfy the property that no revealed identity in Phase 1 was a prefix of it. The challenger sets $b \in \{0, 1\}$ randomly, and encrypts M_b under $(ID^*_1, \ldots, ID^*_{j^*})$. It sends the ciphertext to the attacker.

Phase 2 This is the same as Phase 1, with the added restriction that any revealed identity vector must not be a prefix of $(ID^*_1, \ldots, ID^*_{j^*})$.

Guess The attacker must output a guess b' for b.

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

Definition 14. *A Hierarchical Identity-Based Encryption scheme is secure if all polynomial time attackers achieve at most a negligible advantage in the above security game.*

We note that for schemes where a delegated key is identically distributed to a key produced by a fresh call to the key generation algorithm, one can equivalently use a simplified game, where there are no Create or Delegate queries and instead there are only key requests, which are fulfilled by calling the key generation algorithm and providing the attacker with the resulting key. The constructions we consider in this paper have this feature, and so we use the simplified version of the security game.

B Unbounded HIBE

The main idea of the Lewko-Waters unbounded HIBE construction is to tie together separate IBE instances for each level, where these instances share the *same* public parameters but each have their own random exponents. The instances are tied together by a secret sharing of the master secret key exponent that takes place across the components of the identity vector for each secret key. This approach allows one to form keys and ciphertexts for identity vectors of arbitrary depth from public parameters consisting of a constant number of group elements.

A nested dual system encryption approach is employed to prove security. In addition to using semi-functional keys and ciphertexts, [29] also introduces ephemeral semi-functional keys and ciphertexts. These additional objects are used in the proof to extend the dual system encryption methodology to cope with the small size of the public parameters. As for the LW IBE scheme, the information-theoretic part of the proof is accomplished via pairwise independence: because the public parameters are in the subgroup G_{p_1}, the values of their exponents modulo the other primes remain hidden. This entropy that remains (conditioned on the public parameters) can be used to argue that the attacker cannot distinguish a nominally semi-functional key

and ciphertext IBE pair from a regular semi-functional key and ciphertext IBE pair *when the identities are unequal*. Extending this argument to the unbounded HIBE setting presents a few challenges. First, keys and ciphertexts now consist of potentially *many* IBE instances - and the constant size of the public parameters only enables us to hide nominality for one instance at a time. Second, even when a key cannot decrypt, some of the components of its identity vector may appear also as components in the ciphertext identity vector.

In [29], these problems are addressed by employing a nested hybrid strategy and an encoding for identity vectors. The encoding ensures that if an identity vector $(ID_1, \ldots, ID_j)$ is not a prefix of another identity vector $(ID_1^*, \ldots, ID_{j^*}^*)$, then ID_j does not appear *anywhere* in the set $\{ID_1^*, \ldots, ID_{j^*}^*\}$. All of the semi-functional and ephemeral semi-functional terms for keys will appear in terms corresponding to this last ID_j only. For the ciphertext, these terms will appear everywhere.

The nested hybrid strategy is used to isolate changes in behavior between the keys and ciphertext to occur for one IBE instance at a time. The hybrid proceeds as follows. One should imagine that there are three separate orthogonal spaces: the normal space, the semi-functional space, and the ephemeral semi-functional space. First the ciphertext changes from being normal to semi-functional. More precisely, the ciphertext consists of many IBE instances, and all of these expand into the semi-functional space at once. Next, the first key becomes ephemeral semi-functional. This means that the part of the key corresponding to the IBE instance for the last coordinate of its identity vector expands into the ephemeral semi-functional space. Since this is orthogonal to the regular semi-functional space present in the ciphertext terms, decryption capability is not yet affected. Now, the IBE instances in the ciphertext will also expand into the ephemeral semi-functional space, one by one. Each time, the last part of the key and the changing part of the ciphertext will form a nominal pair in the ephemeral space - in other words, if the ID components were equal, these terms in the ephemeral space would still cancel out. However, the ID components are guaranteed to be unequal by our encoding, and so these terms will appear well-distributed in the ephemeral space in the attacker's view. At this point, the key appears unable to decrypt the ciphertext. Now, the last part of the key expands into the regular semi-functional space, and drops out of the ephemeral semi-functional space. We then remove the ephemeral semi-functional space from the ciphertext and repeat this process for the next key. By the end, we have a semi-functional ciphertext and semi-functional keys incapable of decrypting - at this point, security is easy to prove directly. It is crucial to note that throughout this hybrid process, there is at most one key at a time that has terms in the ephemeral space. This is what allows one to employ the information-theoretic argument that hides nominality for only key and one piece of the ciphertext at a time. For more discussion of this, see [29].

In [29], a scheme is presented in bilinear groups of composite order $N = p_1p_2p_3$, a product of three distinct primes. We present here a simpler version of the scheme using group order $N = p_1p_2p_3p_4p_5$, a product of five distinct primes. The subgroup G_{p_1} is where the main scheme takes place, while G_{p_4} serves as the semi-functional space and G_{p_5} serves as the ephemeral semi-functional space. G_{p_2} is used to additionally randomize ciphertexts, while G_{p_3} is used to additionally randomize keys. These extra randomizing subspaces are used to enable us to make all of the other keys and ciphertext pieces which are not changing during a particular step of our hybrid. These will not be needed in the prime order version, since the asymmetry of the subspace assumption replaces this. Recall, when one is changing from a 2-dimensional subspace in $\mathbb{B}^*$ to a 3-dimensional subspace, one can actually be given a complete basis of $\mathbb{B}^*$ is the exponent, it is only a vector of $\mathbb{B}$ that one is missing. As for the IBE scheme, this property makes simulating the extra randomizing subgroups unnecessary.

B.1 A Simplified Composite Order Construction

We assume that identity vectors have components in $\mathbb{Z}_N$, and are encoded so that if an identity vector $(ID_1, \ldots, ID_j)$ is not a prefix of another identity vector $(ID'_1, \ldots, ID'_{j'})$, then ID_j does not appear in the set $\{ID'_1, \ldots, ID'_{j'}\}$. We also assume that messages are elements of G_T.

Setup$(\lambda) \to \mathrm{PP}, \mathrm{MSK}$ The setup algorithm chooses a bilinear group G of order $N = p_1 p_2 p_3 p_4 p_5$ (where p_1, p_2, p_3, p_4, and p_5 are distinct primes). We let G_{p_i} denote the subgroup of order p_i in G. It then chooses $u_1, g_1, h_1, v_1 \in G_{p_1}$, $g_2 \in G_{p_2}$, and $\alpha \in \mathbb{Z}_N$. The public parameters are published as:

$$\mathrm{PP} := \{N, u_1, g_1, h_1, v_1, g_2, e(g_1, g_1)^\alpha\}.$$

The master secret key consists of α and a generator of G_{p_3}.

Encrypt$(M, \mathrm{PP}, (ID_1, \ldots, ID_j)) \to \mathrm{CT}$ The encryption algorithm chooses $s \in \mathbb{Z}_N$ randomly and chooses a random $t_i \in \mathbb{Z}_N$ for each i from 1 to j. It also chooses random elements $R_i, R'_i, R''_i \in G_{p_2}$ for each i. It creates the ciphertext as:

$$C_0 = M e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} R_i,\ C_{2,i} = g_1^{t_i} R'_i,\ C_{3,i} = (u_1^{ID_i} h_1)^{t_i} R''_i\ \forall i.$$

KeyGen$((ID_1, \ldots, ID_j), \mathrm{MSK}) \to \mathrm{SK}_{ID}$ The key generation algorithm chooses random values $y_1, \ldots, y_j \in \mathbb{Z}_N$, subject to the constraint that $\sum_{i=1}^{j} y_i = \alpha$. For each i from 1 to j, it also chooses a random value $r_i \in \mathbb{Z}_N$ and random elements $W_i, W'_i, W''_i \in G_{p_3}$. It sets the key to be:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W'_i,\ K_{3,i} = g_1^{r_i} W''_i\ \forall i.$$

Delegate$(\mathrm{SK}_{ID}, \mathrm{PP}, ID_{j+1}) \to \mathrm{SK}_{ID|ID_{j+1}}$ Given a key $\{K_{1,i}, K_{2,i}, K_{3,i}\}$ for $(ID_1, \ldots, ID_j)$, the delegation algorithm creates a key for $(ID_1, \ldots, ID_{j+1})$ as follows. It chooses random values y'_i subject to the constraint that $\sum_{i=1}^{j+1} y'_i = 0$, random values $r'_i \in \mathbb{Z}_N$, and random elements $U_i, U'_i, U''_i \in G_{p_3}$. It creates the new key as:

$$K'_{1,i} = K_{1,i} g_1^{y'_i} U_i,\ K'_{2,i} = K_{2,i} v_1^{y'_i} (u_1^{ID_i} h_1)^{r'_i} U'_i,\ K'_{3,i} = K_{3,i} g_1^{r'_i} U''_i,\ \forall i \in [j+1],$$

where we define $K_{1,j+1}, K_{2,j+1}, K_{3,j+1}$ as the identity element. We note that the new key is fully re-randomized.

Decryption$(\mathrm{CT}, \mathrm{SK}_{ID}) \to M$ If the identity vector $(ID_1, \ldots, ID_j)$ of the secret key is a prefix of the identity vector of the ciphertext, the decryption algorithm computes the blinding factor as:

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

The message is then computed as:

$$M = C_0 / B.$$

Correctness When the identity vector $(ID_1, \ldots, ID_j)$ corresponding to the secret key is a prefix of the identity vector corresponding to the ciphertext, then

$$B = \prod_{i=1}^{j} e(C_{1,i}, K_{1,i}) e(C_{3,i}, K_{3,i}) / e(C_{2,i}, K_{2,i})$$

$$= \prod_{i=1}^{j} e(g_1, g_1)^{sy_i} e(g_1, v_1)^{y_i t_i} e(g_1, u_1)^{t_i ID_i r_i} e(g_1, h_1)^{t_i r_i} / \left(e(g_1, v_1)^{y_i t_i} e(g_1, u_1)^{t_i ID_i r_i} e(g_1, h_1)^{t_i r_i} \right)$$

$$= \prod_{i=1}^{j} e(g_1, g_1)^{sy_i} = e(g_1, g_1)^{\alpha s}.$$

B.2 Proof of Security from Variants of the Subgroup Decision Assumption

Our proof of security will rely on the following assumptions. Except for Assumption 4, the rest are instances of the general subgroup decision assumption discussed in Section 2.1. All of these assumptions hold in the generic group model assuming that it is hard to find a non-trivial factor of the group order N. This can be easily verified using the techniques in [24]. We will also use this additional assumption about the hardness of factoring N explicitly in our proof.

Definition 15. *Assumption 1. Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (N = p_1 \cdots p_5, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ g_4 \xleftarrow{R} G_{p_4},\ T_0 \xleftarrow{R} G_{p_1},\ T_1 \xleftarrow{R} G_{p_1 p_5}$$

$$D := (\mathbb{G},\ g_1,\ \ldots,\ g_4).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}[\mathcal{A}(D, T_0) = 1] - \mathbb{P}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in the security parameter λ.

Definition 16. *Assumption 2. Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (N = p_1 \cdots p_5, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1, Y_1 \xleftarrow{R} G_{p_1},\ g_2, X_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ g_4 \xleftarrow{R} G_{p_4},\ X_5, Y_5 \xleftarrow{R} G_{p_5},\ T_0 \xleftarrow{R} G_{p_1 p_2},\ T_1 \xleftarrow{R} G_{p_1 p_2 p_5}$$

$$D := (\mathbb{G},\ g_1,\ \ldots,\ g_4,\ X_2 X_5,\ Y_1 Y_5).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}[\mathcal{A}(D, T_0) = 1] - \mathbb{P}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in the security parameter λ.

Definition 17. *Assumption 3. Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (N = p_1 \cdots p_5, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ g_3, Y_3 \xleftarrow{R} G_{p_3},\ X_4, Y_4 \xleftarrow{R} G_{p_4},\ X_5 \xleftarrow{R} G_{p_5},\ T_0 \xleftarrow{R} G_{p_3 p_5},\ T_1 \xleftarrow{R} G_{p_3 p_4}$$

$$D := (\mathbb{G},\ g_1,\ \ldots,\ g_3,\ X_4 X_5,\ Y_3 Y_4).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}[\mathcal{A}(D, T_0) = 1] - \mathbb{P}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in the security parameter λ.

Definition 18. *Assumption 4. Given a group generator $\mathcal{G}$, we define the following distribution:*

$$\mathbb{G} := (N = p_1 \cdots p_5, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1},\ g_2 \xleftarrow{R} G_{p_2},\ g_3 \xleftarrow{R} G_{p_3},\ g_4, X_4, Y_4 \xleftarrow{R} G_{p_4},\ a, b, c \xleftarrow{R} \mathbb{Z}_N,\ T_0 := e(g_1, g_1)^{abc},\ T_1 \xleftarrow{R} G_T$$

$$D := (\mathbb{G},\ g_1,\ \ldots,\ g_4,\ g_1^a,\ g_1^b,\ g_1^c X_4,\ g_1^{ab} Y_4).$$

We assume that for any PPT algorithm $\mathcal{A}$ (with output in $\{0,1\}$),

$$Adv_{\mathcal{G},\mathcal{A}} := |\mathbb{P}[\mathcal{A}(D, T_0) = 1] - \mathbb{P}[\mathcal{A}(D, T_1) = 1]|$$

is negligible in the security parameter λ.

We now prove:

Theorem 19. *Under Assumptions 1 - 4 above and the assumption that it is hard to find a non-trivial factor of N, our composite order unbounded HIBE construction is fully secure.*

We first define the semi-functional and ephemeral semi-functional objects that will be used in our proof. We define these objects by providing algorithms which generate them, however we stress that these algorithms are used for definitional purposes, and are not part of the real system. In particular, it is not necesssary for these algorithms to run efficiently when given only the public parameters or MSK, etc. The subgroup G_{p_4} will play the role of the semi-functional space, while the subgroup G_{p_5} will play the role of the ephemeral semi-functional space.

KeyGenSF The semi-functional key generation algorithm first produces a normal key, $\{K'_{1,i}, K'_{2,i}, K'_{3,i}\ \forall i = 1, \ldots, j\}$, for an identity vector $(ID_1, \ldots, ID_j)$ as in the normal key generation algorithm. It then chooses random elements $R_4, W_4 \in G_{p_4}$ and defines the semi-functional key as: $K_{1,i} = K'_{1,i}, K_{2,i} = K'_{2,i}, K_{3,i} = K'_{3,i}$ for i from 1 to $j-1$, $K_{1,j} = K'_{1,j}, K_{2,j} = K'_{2,j} R_4$, and $K_{3,j} = K'_{3,j} W_4$. In other words, a semi-functional key is distributed like a normal key, except that there are additional random elements of G_{p_4} attached to $K_{2,j}$ and $K_{3,j}$.

EncryptSF The semi-functional encryption algorithm first produces a normal ciphertext, $\{C_0, C'_{1,i}, C'_{2,i}, C'_{3,i}\ \forall i = 1, \ldots, j\}$, encrypted to an identity vector $(ID_1, \ldots, ID_j)$ as in the normal encryption algorithm. It then chooses random elements $R_4^i, W_4^i, Z_4^i \in G_{p_4}$ for each i from 1 to j. It defines the semi-functional ciphertext as: $C_0, C_{1,i} = C'_{1,i} R_4^i, C_{2,i} = C'_{2,i} W_4^i, C_{3,i} = C'_{3,i} Z_4^i$ for all i from 1 to j. In other words, a semi-functional ciphertext is distributed like a normal ciphertext, except that there are additional random elements of G_{p_4} attached to all terms except C_0.

KeyGenESF The ephemeral semi-functional key generation algorithm first produces a normal key, $\{K'_{1,i}, K'_{2,i}, K'_{3,i}\ \forall i = 1, \ldots, j\}$, for an identity vector $(ID_1, \ldots, ID_j)$ as in the normal key generation algorithm. It then chooses random elements $R_5, W_5 \in G_{p_5}$ and defines the semi-functional key as: $K_{1,i} = K'_{1,i}, K_{2,i} = K'_{2,i}, K_{3,i} = K'_{3,i}$ for i from 1 to $j-1$, $K_{1,j} = K'_{1,j}, K_{2,j} = K'_{2,j} R_5$, and $K_{3,j} = K'_{3,j} W_5$. In other words, an ephemeral semi-functional key is distributed like a normal key, except that there are additional random elements of G_{p_5} attached to $K_{2,j}$ and $K_{3,j}$.

EncryptESF(ℓ) The ephemeral semi-functional encryption algorithm takes in an additional parameter ℓ, which ranges from 0 to j, where j is the length of the identity vector being encrypted under. To produce an ephemeral semi-functional ciphertext of type ℓ, the algorithm first produces a normal ciphertext, $\{C_0, C'_{1,i}, C'_{2,i}, C'_{3,i}\ \forall i = 1, \ldots, j\}$, encrypted to an identity vector $(ID_1, \ldots, ID_j)$ as in the normal encryption algorithm. It then chooses random elements $R_4^i, W_4^i, Z_4^i \in G_{p_4}$ for each i from 1 to j and random elements $R_5^i, W_5^i, Z_5^i \in G_{p_5}$ for each i from 1 to ℓ. It defines the semi-functional ciphertext as: $C_0, C_{1,i} = C'_{1,i} R_4^i R_5^i, C_{2,i} = C'_{2,i} W_4^i W_5^i, C_{3,i} = C'_{3,i} Z_4^i Z_5^i$ for all i from 1 to ℓ, and $C_{1,i} = C'_{1,i} R_4^i, C_{2,i} = C'_{2,i} W_4^i, C_{3,i} = C'_{3,i} Z_4^i$ for i from $\ell + 1$ to j. In other words, an epehemeral semi-functional ciphertext of type ℓ is distributed like a semi-functional ciphertext, except that there are additional random elements of G_{p_5} attached to terms $C_{1,i}, C_{2,i}, C_{3,i}$ for i from 1 to ℓ.

Hybrid Organization We employ a hybrid argument over a sequence of games. Game_{Real} is the real security game. Game_0 is like the real security game, except the ciphertext is now semi-functional. In $\text{Game}_{E_{k,\ell}}$, the first $k-1$ keys are semi-functional, key k is ephemeral semi-functional, the remaining keys are normal, and the ciphertext is ephemeral semi-functional of type ℓ. In $\text{Game}_{P_{k,\ell}}$, the first k keys are semi-functional, the remaining keys are normal, and the ciphertext is ephemeral semi-functional of type ℓ. In Game_{Final}, the keys are all semi-functional and the ciphertext is a semi-functional encryption of a random message.

The sequence of games proceeds as follows. We begin Game_{Real}, then move to Game_0. Next, we move through the games $\text{Game}_{E_{1,\ell}}$ as ℓ goes from 0 to the depth of the ciphertext. We then proceed backwards through the games $\text{Game}_{P_{1,\ell}}$ as ℓ decreases from the ciphertext depth to 0. We then go to $\text{Game}_{E_{2,\ell}}$, and do the loop again. When we arrive at $\text{Game}_{P_{q,0}}$, where q is the number of key queries, all keys are semi-functional, and the ciphertext is semi-functional. We then conclude with Game_{Final}. We prove these games are indistinguishable in the following lemmas.

Lemma 20. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $\text{Game}_{Real}Adv_{\mathcal{A}} - \text{Game}_0 Adv_{\mathcal{A}} = \epsilon$ for ϵ non-negligible. Then there exits a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 1.*

Proof. We invoke Assumption 1 with some of the roles of the primes interchanged. We assume our algorithm $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, g_{p_5}$ and T, where T is either a random element of G_{p_2} or a random element of $G_{p_2 p_4}$. $\mathcal{B}$ chooses a random $\alpha \in \mathbb{Z}_N$, and random elements $u_1, v_1, h_1 \in G_{p_1}$. It sets $g_1 = g_{p_1}$ and $g_2 = g_{p_2}$. It then gives the public parameters $\{N, u_1, g_1, h_1, v_1, g_2, e(g_1, g_1)^\alpha\}$ to the attacker $\mathcal{A}$. We note that $\mathcal{B}$ can form normal keys in response to $\mathcal{A}$'s key queries by using the key generation algorithm, since $\mathcal{B}$ knows the master secret key. At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0, 1\}$ randomly and also chooses random values $s, t_i, a_i, b_i, c_i \in \mathbb{Z}_N$ for each i from 1 to j^*. It forms the challenge ciphertext as:

$$C_0 = M_\beta e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} T^{a_i},\ C_{2,i} = g_1^{t_i} T^{b_i},\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} T^{c_i}\ \forall i.$$

We note that the values of a_i, b_i, c_i modulo p_2 and modulo p_4 are uncorrelated. So if $T \in G_{p_2}$, this is a properly distributed normal ciphertext. If $T \in G_{p_2 p_4}$, this is a properly distributed semi-functional ciphertext. Hence, $\mathcal{B}$ has either simulated Game_{Real} or Game_0, depending on the value of T, and so it can use $\mathcal{A}$'s output to break Assumption 1. □

Lemma 21. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $\text{Game}_{P_{k-1,0}}Adv_{\mathcal{A}} - \text{Game}_{E_{k,0}}Adv_{\mathcal{A}} = \epsilon$ for some non-negligible ϵ. Then we can build a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 1.*

Proof. Our algorithm $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}, T$. It chooses random values $\alpha, y_u, y_v, y_h \in \mathbb{Z}_N$ and sets $g_1 = g_{p_1}, g_2 = g_{p_2}, u_1 = g_1^{y_u}, v_1 = g_1^{y_v}, h_1 = g_1^{y_h}$. It gives the public parameters $\{N, e(g_1, g_1)^\alpha, g_1, g_2, u_1, v_1, h_1\}$ to $\mathcal{A}$. We note that $\mathcal{B}$ can make normal keys because it knows the master secret key. To make the first $i-1$ semi-functional keys, $\mathcal{B}$ proceeds as follows. For each key request $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ chooses random values $y_1, \ldots, y_j \in \mathbb{Z}_N$ such that their sum is α, random values $r_i \in \mathbb{Z}_N$, random elements $W_i, W_i', W_i'' \in G_{p_3}$ for each i from 1 to j, and random elements $Z', Z'' \in G_{p_4}$ (note that $\mathcal{B}$ knows generators of G_{p_3} and G_{p_4}). It sets:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W_i',\ K_{3,i} = g_1^{r_i} W_i'' \ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} W_j,\ K_{2,j} = v_1^{y_j} (u_1^{ID_j} h_1)^{r_j} W_j' Z',\ K_{3,j} = g_1^{r_j} W_j'' Z''.$$

To make the k^{th} requested key for some $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ chooses random values $y_1, \ldots, y_j \in \mathbb{Z}_N$ such that their sum is α, random values $r_i \in \mathbb{Z}_N$ for $i < j$, and random elements $W_i, W_i', W_i'' \in G_{p_3}$. It forms the key as:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W_i',\ K_{3,i} = g_1^{r_i} W_i'' \ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} W_j,\ K_{2,j} = v_1^{y_j} T^{y_u ID_j + y_h} W_j',\ K_{3,j} = T W_j''.$$

We note that if $T \in G_{p_1}$, this is a properly distributed normal key, with r_j equal to the discrete log of T base g_1. If $T \in G_{p_1 p_5}$, this is a properly distributed ephemeral semi-functional key, since the value of $y_u ID_j + y_h$ is randomly distributed modulo p_5.

At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0, 1\}$ randomly, along with random values $s, t_i \in \mathbb{Z}_N$, random elements $R_i, R_i', R_i'' \in G_{p_2}$, and random elements $Z_i, Z_i', Z_i'' \in G_{p_4}$ for all i from to j^* (note that $\mathcal{B}$ knows generators of G_{p_2} and G_{p_4}). It then forms the ciphertext as:

$$C_0 = M_\beta e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} R_i Z_i,\ C_{2,i} = g_1^{t_i} R_i' Z_i',\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} R_i'' Z_i'' \ \forall i.$$

If $T \in G_{p_1}$, $\mathcal{B}$ has properly simulated $\text{Game}_{P_{k-1},0}$. If $T \in G_{p_1 p_5}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{E_{k,0}}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 1. □

Lemma 22. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $Game_{E_{k,\ell}} Adv_{\mathcal{A}} - Game_{E_{k,\ell+1}} Adv_{\mathcal{A}} = \epsilon$ for some non-negligible ϵ. Then we can build a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 2.*

Proof. Our algorithm $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}, X_2 X_5, Y_1 Y_5, T$. It chooses random exponents $\alpha, y_u, y_v, y_h \in \mathbb{Z}_N$, and sets $g_1 = g_{p_1}$, $g_2 = g_{p_2}$, $u_1 = g_1^{y_u}$, $v_1 = g_1^{y_v}$, and $h_1 = g_1^{y_h}$. It gives the public parameters $\{N, g_1, g_2, u_1, v_1, h_1, e(g_1, g_1)^\alpha\}$ to $\mathcal{A}$.

$\mathcal{B}$ responds to the first $k-1$ key requests by making semi-functional keys, which it can easily make because it knows the master secret key and a generator of G_{p_4}. When the attacker requests the k^{th} key for $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ makes an ephemeral semi-functional key as follows. It chooses random values $y_i \in \mathbb{Z}_N$ such that $\sum_{i=1}^{j} y_i = \alpha$, random values $r_i \in \mathbb{Z}_N$ for i from 1 to $j-1$, a random $r_j' \in \mathbb{Z}_N$, and random elements $W_i, W_i', W_i'' \in G_{p_3}$ for i from 1 to j. It creates the key as:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W_i',\ K_{3,i} = g_1^{r_i} W_i'' \ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} W_j,\ K_{2,j} = v_1^{y_j} (Y_1 Y_5)^{y_u ID_j + y_h} W_j',\ K_{3,j} = (Y_1 Y_5) W_j''.$$

We note that this is a properly distributed ephemeral semi-functional key, with r_j equal to the discrete log of Y_1 base g_1.

At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly, along with random values $s, t_i \in \mathbb{Z}_N$ for $i \leq j^*, i \neq \ell+1$, a random value $t'_{\ell+1} \in \mathbb{Z}_N$, random values $a_i, b_i, c_i \in \mathbb{Z}_N$ for $i \leq \ell$, random elements $R_i, R'_i, R''_i \in G_{p_2}$ for $\ell+1 < i \leq j^*$, and random elements $Z_i, Z'_i, Z''_i \in G_{p_4}$ for $i \leq j^*$. $\mathcal{B}$ forms the challenge ciphertext as:

$$C_0 = M_\beta e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} R_i Z_i (X_2X_5)^{a_i},\ C_{2,i} = g_1^{t_i} Z'_i (X_2X_5)^{b_i},\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} (X_2X_5)^{c_i}\ \forall i \leq \ell,$$

$$C_{1,\ell+1} = g_1^s T^{t'_{\ell+1} y_v} Z_{\ell+1},\ C_{2,\ell+1} = T^{t'_{\ell+1}} Z'_{\ell+1},\ C_{3,\ell+1} = T^{t'_{\ell+1}(y_u ID^*_{\ell+1} + y_h)} Z''_{\ell+1},$$

$$C_{1,i} = g_1^s v_1^{t_i} R_i Z_i,\ C_{2,i} = g_1^{t_i} R'_i Z'_i,\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} R''_i Z''_i\ \forall i > \ell+1.$$

We note that $t_{\ell+1}$ is equal to $t'_{\ell+1}$ multiplied by the log base g_1 of the G_{p_1} part of T.

We assume our identities have the property that $(ID_1, \ldots, ID_j)$ being unable to decrypt a ciphertext for $(ID_1^*, \ldots, ID_{j^*}^*)$ implies that ID_j *cannot be equal to any of* $ID_1^*, \ldots, ID_{j^*}^*$. In this case, the values $y_u ID^*_{\ell+1} + y_h$ and $y_u ID_j + y_h$ are randomly distribute modulo p_5 in the attacker's view, since $y_u ID + y_h$ is a pairwise independent function of ID. Here, we are using the assumption that it is hard to find a non-trivial factor of N, which means that the attacker cannot produce (with non-negligible probability) ID components in $\mathbb{Z}_N$ which are unequal modulo N but are equal modulo p_5. Thus, if $T \in G_{p_1p_2}$, the simulator has made a properly distributed semi-functional ciphertext of type ℓ. If $T \in G_{p_1p_2p_5}$, the simulator has made a properly distributed semi-functional ciphertext of type $\ell+1$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 2. □

Lemma 23. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $Game_{E_{k,j^*}} Adv_{\mathcal{A}} - Game_{P_{k,j^*}} Adv_{\mathcal{A}} = \epsilon$, where j^* is the depth of challenge ciphertext and ϵ is non-negligible. Then we can build a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 3.*

Proof. The algorithm $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, X_4X_5, Y_3Y_4, T$. It chooses random values $\alpha, y_u, y_v, y_h \in \mathbb{Z}_N$ and sets $g_1 = g_{p_1}$, $g_2 = g_{p_2}$, $u_1 = g_1^{y_u}$, $v_1 = g_1^{y_v}$, $h_1 = g_1^{y_h}$. It gives the public parameters $\{N, g_1, g_2, u_1, v_1, h_1, e(g_1,g_1)^\alpha\}$ to $\mathcal{A}$.

We suppose $\mathcal{A}$ requests a key for $(ID_1, \ldots, ID_j)$ as one of the first $k-1$ key requests. Then $\mathcal{B}$ creates a semi-functional key as follows. $\mathcal{B}$ chooses random values $y_i \in \mathbb{Z}_N$ such that $\sum_{i=1}^j y_i = \alpha$, random values $r_i \in \mathbb{Z}_N$, random values $a, b \in \mathbb{Z}_N$, and random elements $W_i, W'_i, W''_i \in G_{p_3}$ for i from 1 to j. It creates the key as:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W'_i,\ K_{3,i} = g_1^{r_i} W''_i\ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} W_j,\ K_{2,j} = v_1^{y_j} (u_1^{ID_j} h_1)^{r_j} W'_j (Y_3Y_4)^a,\ K_{3,j} = g_1^{r_j} W''_j (Y_3Y_4)^b.$$

We note that this a properly distributed semi-functional key. For key requests after the k^{th} key request, the simulator can make normal keys because it knows the master secret key.

We suppose $\mathcal{A}$ requests the k^{th} key for $(ID_1, \ldots, ID_j)$. Then $\mathcal{B}$ creates the key as follows. $\mathcal{B}$ chooses random values $y_i \in \mathbb{Z}_N$ such that $\sum_{i=1}^j y_i = \alpha$, random values $r_i \in \mathbb{Z}_N$, a random value $a \in \mathbb{Z}_N$, and random elements $W_i, W'_i, W''_i \in G_{p_3}$ for all i from 1 to j. It sets:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W'_i,\ K_{3,i} = g_1^{r_i} W''_i\ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} W_j,\ K_{2,j} = v_1^{y_j} (u_1^{ID_j} h_1)^{r_j} W'_j T,\ K_{3,j} = g_1^{r_j} W''_j T^a.$$

We note that if $T \in G_{p_3p_5}$, this is a properly distributed ephemeral semi-functional key. If $T \in G_{p_3p_4}$, this is a properly distributed semi-functional key.

At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly, along with random values $s, t_i, a_i, b_i, c_i \in \mathbb{Z}_N$ and random elements $R_i, R_i', R_i'' \in G_{p_2}$ for i from 1 to j^*. It forms the challenge ciphertext as:

$$C_0 = M_\beta e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} R_i (X_4 X_5)^{a_i},\ C_{2,i} = g_1^{t_i} R_i' (X_4 X_5)^{b_i},\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} R_i'' (X_4 X_5)^{c_i}.$$

We note that this is a properly distributed semi-functional ciphertext of type j^*.

If $T \in G_{p_3 p_5}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{E_{k,j^*}}$. If $T \in G_{p_3 p_4}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{P_{k,j^*}}$. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 3. □

Lemma 24. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $Game_{P_{k,\ell}} Adv_{\mathcal{A}} - Game_{P_{k,\ell-1}} Adv_{\mathcal{A}} = \epsilon$ for some non-negligible ϵ. Then we can build a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 2.*

Proof. The algorithm $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}, X_2 X_5, Y_1 Y_5, T$. It chooses random values $\alpha, y_u, y_v, y_h \in \mathbb{Z}_N$ and sets $g_1 = g_{p_1}$, $g_2 = g_{p_2}$, $u_1 = g_1^{y_u}$, $v_1 = g_1^{y_v}$, $h_1 = g_1^{y_h}$. It gives the public parameters $\{N, g_1, g_2, u_1, v_1, h_1, e(g_1, g_1)^\alpha\}$ to $\mathcal{A}$.

We note that the simulator can easily make both normal and semi-functional keys since it knows the master secret key and a generator of G_{p_4}. This allows it to respond to all key queries from $\mathcal{A}$.

At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly, along with random values $s, t_i \in \mathbb{Z}_N$ for $i \leq j^*, i \neq \ell$, a random value $t_\ell' \in \mathbb{Z}_N$, random values $a_i, b_i, c_i \in \mathbb{Z}_N$ for $i < \ell$, random elements $R_i, R_i', R_i'' \in G_{p_2}$ for $i \leq j^*, i \neq \ell$, and random elements $Z_i, Z_i', Z_i'' \in G_{p_4}$ for $i \leq j^*$. $\mathcal{B}$ forms the challenge ciphertext as:

$$C_0 = M_\beta e(g_1, g_1)^{\alpha s},\ C_{1,i} = g_1^s v_1^{t_i} R_i Z_i (X_2 X_5)^{a_i},\ C_{2,i} = g_1^{t_i} Z_i' (X_2 X_5)^{b_i},\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} (X_2 X_5)^{c_i}\ \forall i < \ell,$$

$$C_{1,\ell} = g_1^s T^{t_\ell' y_v} Z_\ell,\ C_{2,\ell} = T^{t_\ell'} Z_\ell',\ C_{3,\ell} = T^{t_\ell'(y_u ID_\ell^* + y_h)} Z_\ell'',$$

$$C_{1,i} = g_1^s v_1^{t_i} R_i Z_i,\ C_{2,i} = g_1^{t_i} R_i' Z_i',\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i} R_i'' Z_i''\ \forall i > \ell.$$

We note that t_ℓ is equal to t_ℓ' multiplied by the log base g_1 of the G_{p_1} part of T.

If $T \in G_{p_1 p_2}$, the simulator has made a properly distributed ephemeral semi-functional ciphertext of type $\ell - 1$. If $T \in G_{p_1 p_2 p_5}$, the simulator has made a properly distributed semi-functional ciphertext of type ℓ. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 2. □

Lemma 25. *Suppose there exists a PPT attacker $\mathcal{A}$ such that $Game_{P_{q,0}} Adv_{\mathcal{A}} - Game_{Final} Adv_{\mathcal{A}} = \epsilon$, where q is number of key queries and ϵ is non-negligible. Then we can build a PPT algorithm $\mathcal{B}$ with non-negligible advantage in breaking Assumption 4.*

Proof. The simulator $\mathcal{B}$ is given $N, g_{p_1}, g_{p_2}, g_{p_3}, g_{p_4}, g_{p_1}^a, g_{p_1}^b, g_{p_1}^c X_4, g_{p_1}^{ab} Y_4, T$. It chooses random values $y_u, y_h \in \mathbb{Z}_N$ and sets $g_1 = g_{p_1}$, $g_2 = g_{p_2}$, $u_1 = g_1^{y_u}$, and $h_1 = g_1^{y_h}$. It sets $v_1 = g_{p_1}^b$. It implicitly sets $\alpha = a$ by setting $e(g_1, g_1)^\alpha = e(g_1, g_1^a)$. It gives the public parameters $\{N, g_1, g_2, u_1, v_1, h_1, e(g_1, g_1)^\alpha\}$ to $\mathcal{A}$.

To respond to a key request for $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ chooses random values $y_i \in \mathbb{Z}_N$ such that $\sum_{i=1}^j y_i = 0$, random values $r_i \in \mathbb{Z}_N$, random elements $W_i, W_i', W_i'' \in G_{p_3}$ for all i from 1 to j, and random elements $Z_4, Z_4' \in G_{p_4}$. It creates the key as:

$$K_{1,i} = g_1^{y_i} W_i,\ K_{2,i} = v_1^{y_i} (u_1^{ID_i} h_1)^{r_i} W_i',\ K_{3,i} = g_1^{r_i} W_i''\ \forall i < j,$$

$$K_{1,j} = g_1^{y_j} g_1^a W_j,\ K_{2,j} v_1^{y_j} g_1^{ab} Y_4 (u_1^{ID_j} h_1)^{r_j} W_j' Z_4,\ K_{3,j} = g_1^{r_j} W_j'' Z_4'.$$

We note that this is a properly distributed semi-functional key.

At some point, $\mathcal{A}$ requests a ciphertext for an identity $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1. $\mathcal{B}$ chooses $\beta \in \{0,1\}$ randomly, as well as random values $t_i' \in \mathbb{Z}_N$, random elements $R_i, R_i', R_i'' \in G_{p_2}$, and random elements $U_i, U_i', U_i'' \in G_{p_4}$ for all i from 1 to j^*. It will implicitly set $s = -bc$ and $t_i = c + t_i'$. It forms the ciphertext as:

$$C_0 = M_\beta T^{-1}, C_{1,i} = (g_1^b)^{t_i'} R_i U_i,\ C_{2,i} = g_1^{t_i'}(g_1^c X_4) R_i' U_i',\ C_{3,i} = (u_1^{ID_i^*} h_1)^{t_i'} (g_1^c X_4)^{y_u ID_i^* + y_h} R_i'' U_i''.$$

If $T = e(g_1, g_1)^{abc}$, this is a properly distributed semi-functional encryption of M_β. If T is a random element of G_T, this is a properly distributed semi-functional encryption of a random message. Hence, $\mathcal{B}$ can use the output of $\mathcal{A}$ to break Assumption 4. □

This completes the proof of Theorem 19.

B.3 The Prime Order Translation

To design our prime order translation of the composite order scheme above, we proceed very similarly to our IBE translation. We do one thing differently - instead of attaching the α term to a basis vector that also plays a separate role, we expand α to be a 2-dimensional entity and to have its own separate space. This results in a scheme that may be slightly larger in dimension than necessary, but it allows us to make the final stage of our proof more clear. As a result, our normal space will be covered by the first 6 vectors of $\mathbb{D}, \mathbb{D}^*$, while the next 2 vectors serve as the semi-functional space and the final two vectors serve as the ephemeral semi-functional space (we work with a 10-dimensional space in total).

Setup$(\lambda) \to \text{PP}, \text{MSK}$ The setup algorithm takes in the security parameter λ and chooses a bilinear group G of sufficiently large prime order p. We let $e : G \times G \to G_T$ denote the bilinear map. We set $n = 10$. The algorithm samples random dual orthonormal bases, $(\mathbb{D}, \mathbb{D}^*) \xleftarrow{R} \mathit{Dual}(\mathbb{Z}_p^n)$. We let $\vec{d}_1, \ldots, \vec{d}_n$ denote the elements of $\mathbb{D}$ and $\vec{d}_1^*, \ldots, \vec{d}_n^*$ denote the elements of $\mathbb{D}^*$. It also chooses random exponents $\alpha_1, \alpha_2, \theta, \sigma, \gamma, \xi \in \mathbb{Z}_p$. The public parameters are

$$\text{PP} := \{G, p, e(g,g)^{\alpha_1 \vec{d}_1 \cdot \vec{d}_1^*}, e(g,g)^{\alpha_2 \vec{d}_2 \cdot \vec{d}_2^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\},$$

and the master secret key is

$$\text{MSK} := \{G, p, \alpha_1, \alpha_2, g^{\vec{d}_1^*}, g^{\vec{d}_2^*}, g^{\gamma \vec{d}_1^*}, g^{\xi \vec{d}_2^*}, g^{\theta \vec{d}_3^*}, g^{\theta \vec{d}_4^*}, g^{\sigma \vec{d}_5^*}, g^{\sigma \vec{d}_6^*}\}.$$

KeyGen$(\text{MSK}, (ID_1, \ldots, ID_j)) \to \text{SK}_{ID}$ The key generation algorithm chooses random values $r_1^i, r_2^i \in \mathbb{Z}_p$ for each i from 1 to j. It also chooses random values $y_1, \ldots, y_j \in \mathbb{Z}_p$ and $w_1, \ldots, w_j \in \mathbb{Z}_p$ up to the constraints that $y_1 + y_2 + \cdots + y_j = \alpha_1$ and $w_1 + w_2 + \cdots + w_j = \alpha_2$. For each i from 1 to j, it computes:

$$K_i := g^{y_i \vec{d}_1^* + w_i \vec{d}_2^* + r_1^i ID_i \theta \vec{d}_3^* - r_1^i \theta \vec{d}_4^* + r_2^i ID_i \sigma \vec{d}_5^* - r_2^i \sigma \vec{d}_6^*}.$$

The secret key is formed as:

$$\text{SK}_{ID} := \{g^{\gamma \vec{d}_1^*}, g^{\xi \vec{d}_2^*}, g^{\theta \vec{d}_3^*}, g^{\theta \vec{d}_4^*}, g^{\sigma \vec{d}_5^*}, g^{\sigma \vec{d}_6^*}, K_1, \ldots, K_j\}.$$

Delegate$(\mathrm{SK}_{ID}, ID_{j+1}) \to \mathrm{SK}_{ID|ID_{j+1}}$ The delegation algorithm chooses random values $\omega_1^i, \omega_2^i \in \mathbb{Z}_p$ for each i from 1 to $j+1$. It also chooses random values $y'_1, \ldots, y'_{j+1}, w'_1, \ldots, w'_{j+1} \in \mathbb{Z}_p$ subject to the constraint that $y'_1 + \cdots y'_{j+1} = 0 = w'_1 + \cdots w'_{j+1}$. Letting $g^{\gamma\vec{d}_1^*}, g^{\xi\vec{d}_2^*}, g^{\theta\vec{d}_3^*}, g^{\theta\vec{d}_4^*}, g^{\sigma\vec{d}_5^*}, g^{\sigma\vec{d}_6^*}, K_1, \ldots, K_j$ denote the elements of SK_{ID}, $\mathrm{SK}_{ID|ID_{j+1}}$ is formed as:

$$\mathrm{SK}_{ID|ID_{j+1}} := \{g^{\gamma\vec{d}_1^*}, g^{\xi\vec{d}_2^*}, g^{\theta\vec{d}_3^*}, g^{\theta\vec{d}_4^*}, g^{\sigma\vec{d}_5^*}, g^{\sigma\vec{d}_6^*},$$

$$K_1 \cdot g^{y'_1\gamma\vec{d}_1^* + w'_1\xi\vec{d}_2^* + \omega_1^1 ID_1\theta\vec{d}_3^* - \omega_1^1\theta\vec{d}_4^* + \omega_2^1 ID_1\sigma\vec{d}_5^* - \omega_2^1\sigma\vec{d}_6^*},$$

$$\ldots, K_j \cdot g^{y'_j\gamma\vec{d}_1^* + w'_j\xi\vec{d}_2^* + \omega_1^j ID_1\theta\vec{d}_3^* - \omega_1^j\theta\vec{d}_4^* + \omega_2^j ID_1\sigma\vec{d}_5^* - \omega_2^j\sigma\vec{d}_6^*},$$

$$g^{y'_{j+1}\gamma\vec{d}_1^* + w'_{j+1}\xi\vec{d}_2^* + \omega_1^{j+1} ID_{j+1}\theta\vec{d}_3^* - \omega_1^{j+1}\theta\vec{d}_4^* + \omega_2^{j+1} ID_{j+1}\sigma\vec{d}_5^* - \omega_2^{j+1}\sigma\vec{d}_6^*}\}.$$

We note that $y_1 + \gamma y'_1, \ldots, y_j + \gamma y'_j, \gamma y'_{j+1}$ are randomly distributed up to the constraint that their sum is α_1, and similarly $w_1 + \xi w'_1, \ldots, w_j + \xi w'_j, \xi w'_{j+1}$ are randomly distributed up to the constraint that their sum is α_2. Also, $r_1^i + \omega_1^i$ and $r_2^i + \omega_2^i$ are uniformly random for each i. The delegation process therefore produces a secret key which is identically distributed to one obtained from calling the key generation algorithm directly.

Encrypt$(\mathrm{PP}, M, (ID_1, \ldots, ID_j)) \to \mathrm{CT}$ The encryption algorithm chooses random values $s_1, s_2 \in \mathbb{Z}_p$, as well as random values t_1^i, t_2^i for each i from 1 to j. It computes

$$C_0 := Me(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*},$$

as well as

$$C_i := g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i t_1^i \vec{d}_4 + t_2^i\vec{d}_5 + ID_i t_2^i \vec{d}_6}$$

for each i from 1 to j. The ciphertext is $\mathrm{CT} := \{C_0, C_1, \ldots, C_j\}$.

Decrypt$(\mathrm{CT}, \mathrm{SK}_{ID}) \to M$ When the identity vector $(ID_1, \ldots, ID_j)$ corresponding to the secret key is a prefix of the identity vector corresponding to the ciphertext, the decryption algorithm computes

$$B := \prod_{i=1}^{j} e_n(C_i, K_i)$$

and computes the message as:

$$M = C_0/B.$$

B.3.1 Correctness

To verify the correctness of our scheme, we observe that when the identity vector $(ID_1, \ldots, ID_j)$ corresponding to the secret key is a prefix of the identity vector corresponding to the ciphertext:

$$\prod_{i=1}^{j} e_n(C_i, K_i) = \prod_{i=1}^{j} e(g,g)^{(s_1 y_i + s_2 w_i + t_1^i r_1^i ID_i - t_1^i r_1^i ID_i + t_2^i r_2^i ID_i - t_2^i r_2^i ID_i)\psi}$$

$$= e(g,g)^{(s_1(y_1 + \cdots + y_j) + s_2(w_1 + \cdots + w_j))\psi} = e(g,g)^{(\alpha_1 s_1 + \alpha_2 s_2)\psi}.$$

Here, ψ denotes $\vec{d}_i \cdot \vec{d}_i^*$, which is the same for all i.

B.4 Semi-functional Algorithms

We formally define semi-functional and ephemeral semi-functional keys and ciphertexts to be the distributions produced by the following algorithms.

KeyGenSF To generate a semi-functional key for $(ID_1, \ldots, ID_j)$, the semi-functional key generation algorithm chooses random values $z_7, z_8 \in \mathbb{Z}_p$, random values $r_1^i, r_2^i \in \mathbb{Z}_p$ for each i from 1 to j as well as random values $y_1, \ldots, y_j \in \mathbb{Z}_p$ such that $y_1 + \cdots + y_j = \alpha_1$ and random values $w_1, \ldots, w_j \in \mathbb{Z}_p$ such that $w_1 + \cdots + w_j = \alpha_2$. It forms the secret key as:

$$K_i := g^{y_i\vec{d}_1^* + w_i\vec{d}_2^* + r_1^i ID_i\theta\vec{d}_3^* - r_1^i\theta\vec{d}_4^* + r_2^i ID_i\sigma\vec{d}_5^* - r_2^i\sigma\vec{d}_6^*}$$

for each i from 1 to $j-1$ and

$$K_j := g^{y_j\vec{d}_1^* + w_j\vec{d}_2^* + r_1^j ID_j\theta\vec{d}_3^* - r_1^j\theta\vec{d}_4^* + r_2^j ID_j\sigma\vec{d}_5^* - r_2^j\sigma\vec{d}_6^* + z_7\vec{d}_7^* + z_8\vec{d}_8^*}$$

The other elements of the secret key (which are only used for delegation) are defined exactly as in the normal key generation algorithm. In summary, a semi-functional key is distributed like a normal key with additional random multiples of $\vec{d}_7^*$ and $\vec{d}_8^*$ added in the exponent of K_j.

EncryptSF To generate a semi-functional ciphertext for $(ID_1, \ldots, ID_j)$, the semi-functional encryption algorithm chooses random values $t_1^i, t_2^i, v_7^i, v_8^i \in \mathbb{Z}_p$ for each i from 1 to j as well as random values $s_1, s_2 \in \mathbb{Z}_p$. It forms the ciphertext as:

$$C_0 := Me(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*},$$

$$C_i := g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i t_1^i\vec{d}_4 + t_2^i\vec{d}_5 + ID_i t_2^i\vec{d}_6 + v_7^i\vec{d}_7 + v_8^i\vec{d}_8}$$

for all i from 1 to j. This is distributed like a normal ciphertext with additional random multiples of $\vec{d}_7$ and $\vec{d}_8$ added in the exponents.

KeyGenESF To generate an ephemeral semi-functional key for $(ID_1, \ldots, ID_j)$, the ephemeral semi-functional key generation algorithm chooses random values $z_9, z_{10} \in \mathbb{Z}_p$, random values $r_1^i, r_2^i \in \mathbb{Z}_p$ for each i from 1 to j as well as random values $y_1, \ldots, y_j \in \mathbb{Z}_p$ such that $y_1 + \cdots + y_j = \alpha_1$ and random values $w_1, \ldots, w_j \in \mathbb{Z}_p$ such that $w_1 + \cdots + w_j = \alpha_2$. It forms the secret key as:

$$K_i := g^{y_i\vec{d}_1^* + w_i\vec{d}_2^* + r_1^i ID_i\theta\vec{d}_3^* - r_1^i\theta\vec{d}_4^* + r_2^i ID_i\sigma\vec{d}_5^* - r_2^i\sigma\vec{d}_6^*}$$

for each i from 1 to $j-1$ and

$$K_j := g^{y_j\vec{d}_1^* + w_j\vec{d}_2^* + r_1^j ID_j\theta\vec{d}_3^* - r_1^j\theta\vec{d}_4^* + r_2^j ID_j\sigma\vec{d}_5^* - r_2^j\sigma\vec{d}_6^* + z_9\vec{d}_9^* + z_{10}\vec{d}_{10}^*}.$$

The other elements of the secret key (which are only used for delegation) are defined exactly as in the normal key generation algorithm. In summary, an ephemeral semi-functional key is distributed like a normal key with additional random multiples of $\vec{d}_9^*$ and $\vec{d}_{10}^*$ added in the exponent of K_j.

EncryptESF(ℓ) To generate an ephemeral semi-functional ciphertext of type ℓ for $(ID_1, \ldots, ID_j)$, the ephemeral semi-functional encryption algorithm chooses random values $t_1^i, t_2^i, v_7^i, v_8^i, v_9^i, v_{10}^i \in \mathbb{Z}_p$ for each i from 1 to ℓ, random values $t_1^i, t_2^i, v_7^i, v_8^i \in \mathbb{Z}_p$ for each i from $\ell + 1$ to j, as well as random values $s_1, s_2 \in \mathbb{Z}_p$. It forms the ciphertext as:

$$C_0 := Me(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*},$$

$$C_i := g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i t_1^i \vec{d}_4 + t_2^i \vec{d}_5 + ID_i t_2^i \vec{d}_6 + v_7^i \vec{d}_7 + v_8^i \vec{d}_8 + v_9^i \vec{d}_9 + v_{10}^i \vec{d}_{10}}$$

for i from 1 to ℓ and

$$C_i := g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i t_1^i \vec{d}_4 + t_2^i \vec{d}_5 + ID_i t_2^i \vec{d}_6 + v_7^i \vec{d}_7 + v_8^i \vec{d}_8}$$

for i from $\ell + 1$ and j.

This is distributed like a normal ciphertext with additional random multiples of $\vec{d}_7$, $\vec{d}_8$, $\vec{d}_9$, $\vec{d}_{10}$ added in the exponents for $C_1, \ldots, C_\ell$ and random multiples of $\vec{d}_7$, $\vec{d}_8$ added in the exponents for $C_{\ell+1}, \ldots, C_j$.

B.5 Security Proof

We now prove the following theorem:

Theorem 26. *Under the decisional linear assumption, the HIBE scheme presented in Section B.3 is fully secure.*

We prove this using a hybrid argument over a sequence of games. This precisely follows the structure of the security proof for the composite order system. Game_{Real} is the real security game. Game_0 is like the real security game, except with a semi-functional ciphertext. In $\text{Game}_{E_{m,\ell}}$, the first $m-1$ keys are semi-functional, key m is ephemeral semi-functional, the remaining keys are normal, and the ciphertext is semi-functional of type ℓ. In $\text{Game}_{P_{m,\ell}}$, the first m keys are semi-functional, the remaining keys are normal, and the ciphertext is semi-functional of type ℓ. In Game_{Final}, the ciphertext is an encryption of a random message. We will describe the Game_{Final} more completely later.

The sequence of games proceeds as follows. We begin with Game_{Real}, then move to Game_0. Next, we move through the games $\text{Game}_{E_{1,\ell}}$ as ℓ goes from 0 to the depth of the ciphertext, which we denote by j^*. We then proceed backwards through the games $\text{Game}_{P_{1,\ell}}$ as ℓ decreases from j^* to 0. We then go to $\text{Game}_{E_{2,\ell}}$, and do the loop again. When we arrive at $\text{Game}_{P_{q,0}}$, where q is the number of key queries, all keys are semi-functional, and the ciphertext is semi-functional. We then conclude with Game_{Final}. We prove these games are indistinguishable in the following lemmas.

Lemma 27. *If there exists a PPT algorithm $\mathcal{A}$ such that $Adv_{\mathcal{A}}^{real} - Adv_{\mathcal{A}}^{0}$ is non-negligible, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 10$.*

We will prove this lemma via a hybrid argument over the length of the identity vector for the challenge ciphertext. We let j denote this length. For ℓ from 0 to j, we define $\text{Game}_{0,\ell}$ to be the same as Game_{Real}, except that $C_1, \ldots, C_\ell$ in the challenge ciphertext are distributed as in a semi-functional ciphertext and $C_{\ell+1}, \ldots, C_j$ are distributed as in a normal ciphertext. In other words, $C_1, \ldots, C_\ell$ have random multiples of $\vec{d}_6, \vec{d}_7$ attached to them, while the rest of the C_i's do not. We note that $\text{Game}_{0,0} = \text{Game}_{Real}$ and $\text{Game}_{0,j} = \text{Game}_0$. Lemma 27 is then implied by the following:

Lemma 28. *For each ℓ from 0 to $j^* - 1$, if there exists a PPT algorithm $\mathcal{A}$ that achieves a non-negligible difference in advantage between $Game_{0,\ell}$ and $Game_{0,\ell+1}$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 10$.*

Proof. We suppose that such an $\mathcal{A}$ exists for some fixed ℓ. We create $\mathcal{B}$ as follows. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\vec{b}_7}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}_1^*}, g^{\eta\vec{b}_2^*}, g^{\beta\vec{b}_3^*}, g^{\beta\vec{b}_4^*}, g^{\vec{b}_5^*}, \ldots, g^{\vec{b}_{10}^*}\right),$$

along with T_1, T_2. It is $\mathcal{B}$'s task to decide whether T_1, T_2 are distributed as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^*}$ or as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^* + \tau_3\vec{b}_5^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^* + \tau_3\vec{b}_6^*}$.

We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ by:

$$\vec{f}_1 = \vec{b}_9^*,\ \vec{f}_2 = \vec{b}_{10}^*\ \vec{f}_3 = \eta\vec{b}_1^*,\ \vec{f}_4 = \eta\vec{b}_2^*,\ \vec{f}_5 = \beta\vec{b}_3^*,\ \vec{f}_6 = \beta\vec{b}_4^*,\ \vec{f}_7 = \vec{b}_5^*,\ \vec{f}_8 = \vec{b}_6^*,\ \vec{f}_9 = \vec{b}_7^*,\ \vec{f}_{10} = \vec{b}_8^*,$$

$$\vec{f}_1^* = \vec{b}_9,\ \vec{f}_2^* = \vec{b}_{10},\ \vec{f}_3^* = \eta^{-1}\vec{b}_1,\ \vec{f}_4^* = \eta^{-1}\vec{b}_2,\ \vec{f}_5^* = \beta^{-1}\vec{b}_3,\ \vec{f}_6^* = \beta^{-1}\vec{b}_4,\ \vec{f}_7^* = \vec{b}_5,\ \vec{f}_8^* = \vec{b}_6,\ \vec{f}_9^* = \vec{b}_7,\ \vec{f}_{10}^* = \vec{b}_8.$$

$\mathcal{B}$ chooses a random matrix $A \in \mathbb{Z}_p^{2\times 2}$. With all but negligible probability, A is invertible. $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{F}_A$ and $\mathbb{D}^* = \mathbb{F}_A^*$, where A is applied as a change of basis matrix to $\vec{f}_7$ and $\vec{f}_8$ (and $(A^{-1})^t$ is applied to $\vec{f}_7^*, \vec{f}_8^*$). We note that $\mathbb{D}$ and $\mathbb{D}^*$ are properly distributed, and reveal no information about A.

$\mathcal{B}$ chooses random values $\alpha_1, \alpha_2, \gamma, \xi, \theta', \sigma' \in \mathbb{Z}_p$ and implicitly sets $\theta = \theta'\eta$, $\sigma = \sigma'\beta$. We note that $\mathcal{B}$ can then produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_{10}}$ as well as $g^{\vec{d}_1^*}, g^{\vec{d}_2^*}, g^{\gamma\vec{d}_1^*}, g^{\xi\vec{d}_2^*}, g^{\theta\vec{d}_3^*}, g^{\theta\vec{d}_4^*}, g^{\sigma\vec{d}_5^*}, g^{\sigma\vec{d}_6^*}$. Thus, $\mathcal{B}$ knows the public parameters as well as the master secret key. $\mathcal{B}$ gives PP to $\mathcal{A}$, and responds to $\mathcal{A}$'s key requests by producing normal secret keys via the normal key generation algorithm.

At some point, $\mathcal{A}$ specifies an identity vector $(ID_1^*, \ldots, ID_{j^*}^*)$ and two messages M_0, M_1 for the challenge ciphertext. $\mathcal{B}$ chooses a random $b \in \{0, 1\}$ and encrypts M_b as follows. It chooses a random $s_1, s_2 \in \mathbb{Z}_p$ and computes

$$C_0 = M_b e(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*}.$$

For i from to 1 to ℓ, it chooses random values $t_1^i, t_2^i, v_7^i, v_8^i \in \mathbb{Z}_p$. It sets:

$$C_i = g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i^* t_1^i \vec{d}_4 + t_2^i\vec{d}_5 + ID_i^* t_2^i \vec{d}_6 + v_7^i\vec{d}_7 + v_8^i\vec{d}_8}.$$

These ciphertext pieces are distributed as in a semi-functional ciphertext.

For $\ell + 1$, $\mathcal{B}$ implicitly sets $t_1^{\ell+1} = \tau_1$ and $t_2^{\ell+1} = \tau_2$. It forms $C_{\ell+1}$ as:

$$C_{\ell+1} = g^{s_1\vec{d}_1 + s_2\vec{d}_2} T_1 (T_2)^{ID_{\ell+1}^*}.$$

For each i from $\ell + 2$ to j^*, $\mathcal{B}$ chooses random values $t_1^i, t_2^i \in \mathbb{Z}_p$ and sets:

$$C_i = g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i^* t_1^i \vec{d}_4 + t_2^i\vec{d}_5 + ID_i^* t_2^i \vec{d}_6}.$$

These ciphertext pieces are distributed as in a normal ciphertext.

If T_1, T_2 are equal to $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^*}$, then $C_{\ell+1}$ is distributed as in a normal ciphertext, and $\mathcal{B}$ has properly simulated $\text{Game}_{0,\ell}$. If T_1, T_2 are equal to $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^* + \tau_3\vec{b}_5^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^* + \tau_3\vec{b}_6^*}$, then $C_{\ell+1}$ includes a linear combination of $\vec{d}_7$ and $\vec{d}_8$. The coefficients of this combination are the coordinates of the vector

$$\tau_3 A^{-1}(1, ID_{\ell_1}^*)^t.$$

Since A is random and independent of everything else given to $\mathcal{A}$, this vector is uniformly random in $\mathcal{A}$'s view. Thus, $\mathcal{B}$ has properly simulated $\text{Game}_{0,\ell+1}$ in this case. This allows $\mathcal{B}$ to use $\mathcal{A}$'s non-negligible difference in advantage between these two games in order to obtain a non-negligible advantage against our subspace assumption. □

Lemma 29. *For each m from 1 to q, if there exists a PPT algorithm $\mathcal{A}$ such that $\mathcal{A}$ achieves a non-negligible difference in advantage between $Game_{P(m-1,0)}$ and $Game_{E(m,0)}$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 10$.*

Proof. We suppose that such an $\mathcal{A}$ exists for some fixed m. We create $\mathcal{B}$ as follows. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\vec{b}_7}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}^*_1}, g^{\eta\vec{b}^*_2}, g^{\beta\vec{b}^*_3}, g^{\beta\vec{b}^*_4}, g^{\vec{b}^*_5}, \ldots, g^{\vec{b}^*_{10}}\right),$$

along with T_1, T_2. It is $\mathcal{B}$'s task to decide whether T_1, T_2 are distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6}$.

We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ by:

$$\vec{f}_1 = \vec{b}_9,\ \vec{f}_2 = \vec{b}_{10},\ \vec{f}_3 = \vec{b}_1,\ \vec{f}_4 = \vec{b}_2,\ \vec{f}_5 = \vec{b}_3,\ \vec{f}_6 = \vec{b}_4,\ \vec{f}_7 = \vec{b}_7,\ \vec{f}_8 = \vec{b}_8,\ \vec{f}_9 = \vec{b}_5,\ \vec{f}_{10} = \vec{b}_6,$$

$$\vec{f}^*_1 = \vec{b}^*_9,\ \vec{f}^*_2 = \vec{b}^*_{10},\ \vec{f}^*_3 = \vec{b}^*_1,\ \vec{f}^*_4 = \vec{b}^*_2,\ \vec{f}^*_5 = \vec{b}^*_3,\ \vec{f}^*_6 = \vec{b}^*_4,\ \vec{f}^*_7 = \vec{b}^*_7,\ \vec{f}^*_8 = \vec{b}^*_8,\ \vec{f}^*_9 = \vec{b}^*_5,\ \vec{f}^*_{10} = \vec{b}^*_6.$$

$\mathcal{B}$ chooses a random matrix $A \in \mathbb{Z}_p^{2\times 2}$. With all but negligible probability, A is invertible. $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{F}_A$ and $\mathbb{D}^* = \mathbb{F}^*_A$, where A is applied as a change of basis matrix to f_9, f_{10}, and $(A^{-1})^t$ is applied to f^*_9, f^*_{10}. We observe that $\mathbb{D}$ and $\mathbb{D}^*$ are properly distributed, and reveal no information about A.

$\mathcal{B}$ chooses $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ randomly. This allows it to compute $e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1}$ as $e_n(g^{\vec{b}_9}, g^{\vec{b}^*_9})^{\alpha_1}$ and $e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}^*_2} = e_n(g^{\vec{b}_{10}}, g^{\vec{b}^*_{10}})^{\alpha_2}$. This allows $\mathcal{B}$ to give $\mathcal{A}$ the pubic parameters:

$$\text{PP} = \left\{G, p, e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1}, e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}^*_2}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\right\}.$$

$\mathcal{B}$ random chooses $\gamma, \xi \in \mathbb{Z}_p$ and implicitly sets $\theta = \eta$ and $\sigma = \beta$. We observe that $\mathcal{B}$ knows the MSK, since $g^{\theta\vec{d}^*_3} = g^{\eta\vec{b}^*_1}$, $g^{\theta\vec{d}^*_4} = g^{\eta\vec{b}^*_2}$, $g^{\sigma\vec{d}^*_5} = g^{\beta\vec{b}^*_3}$, $g^{\sigma\vec{d}^*_6} = g^{\beta\vec{b}^*_4}$.

For the first $m-1$ key requests $\mathcal{A}$ makes, $\mathcal{B}$ will respond by making a semi-functional key. $\mathcal{B}$ can do this by creating a normal key via the normal key generation algorithm and then appending random multiples of $\vec{d}^*_7 = \vec{b}^*_7$ and $\vec{d}^*_8 = \vec{b}^*_8$ to the exponent of K_j (since $\mathcal{B}$ knows $g^{\vec{b}^*_7}, g^{\vec{b}^*_8}$.

To make the m^{th} key for identity vector $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ makes $K_1, \ldots, K_{j-1}$ as in the normal key generation algorithm. To make K_j, it implicitly sets $r^j_1 = \tau_1$ and $r^j_2 = \tau_2$. It computes:

$$K_j = g^{y_j\vec{d}^*_1 + w_j\vec{d}^*_2} \cdot T_1^{ID_j} \cdot T_2.$$

If T_1, T_2 are equal to $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4}$, then this exponent has no multiples of $\vec{d}^*_7, \ldots, \vec{d}^*_{10}$ and is distributed as in a normal key. If T_1, T_2 are equal to $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_3+\tau_3\vec{b}^*_5}, g^{\tau_1\eta\vec{b}^*_2+\tau_2\beta\vec{b}^*_4+\tau_3\vec{b}^*_6}$, then this exponent includes a linear combination of $\vec{d}^*_9$ and $\vec{d}^*_{10}$. The coefficients of this combination are equal to the entries of the vector $\tau_3 A^t(ID_j, -1)^t$. These coefficients are uniformly random in $\mathcal{A}$'s view, since everything else the attacker is given is distributed independently of the matrix A. Hence, in this case, $\mathcal{B}$ produces a well-distributed ephemeral semi-functional key.

For the remaining key requests, $\mathcal{B}$ responds by creating normal keys via the normal key generation algorithm. $\mathcal{B}$ can also produce a well-distributed semi-functional ciphertext, since

it knows the public parameters as well as $g^{\vec{d}_7} = g^{\vec{b}_7}$ and $g^{\vec{d}_8} = g^{\vec{b}_8}$. In summary, $\mathcal{B}$ properly simulates either $\text{Game}_{P(m-1,0)}$ or $\text{Game}_{E(m,0)}$, depending on the values of T_1, T_2. Thus, it can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the subspace assumption. □

Lemma 30. *For each m from 1 to q and each ℓ from 1 to j^*, if there exists a PPT algorithm $\mathcal{A}$ such that $\mathcal{A}$ achieves a non-negligible difference in advantage between $Game_{E(m,\ell-1)}$ and $Game_{E(m,\ell)}$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 10$.*

Proof. We suppose that there exists such an $\mathcal{A}$ for some fixed m and ℓ. We create $\mathcal{B}$ as follows. $\mathcal{B}$ is given:

$$D = \left(g^{\vec{b}_1}, \ldots, g^{\vec{b}_4}, g^{\vec{b}_7}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}_1^*}, g^{\eta\vec{b}_2^*}, g^{\beta\vec{b}_3^*}, g^{\beta\vec{b}_4^*}, g^{\vec{b}_5^*}, \ldots, g^{\vec{b}_{10}^*}\right),$$

along with T_1, T_2. It is $\mathcal{B}$'s task to decide whether T_1, T_2 are distributed as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^*}$ or as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_3^* + \tau_3\vec{b}_5^*}, g^{\tau_1\eta\vec{b}_2^* + \tau_2\beta\vec{b}_4^* + \tau_3\vec{b}_6^*}$.

We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ by:

$$\vec{f}_1 = \vec{b}_9^*,\ \vec{f}_2 = \vec{b}_{10}^*,\ \vec{f}_3 = \eta\vec{b}_1^*,\ \vec{f}_4 = \eta\vec{b}_2^*,\ \vec{f}_5 = \beta\vec{b}_3^*,\ \vec{f}_6 = \beta\vec{b}_4^*,\ \vec{f}_7 = \vec{b}_7^*,\ \vec{f}_8 = \vec{b}_8^*,\ \vec{f}_9 = \vec{b}_5^*,\ \vec{f}_{10} = \vec{b}_6^*,$$

$$\vec{f}_1^* = \vec{b}_9,\ \vec{f}_2^* = \vec{b}_{10},\ \vec{f}_3^* = \eta^{-1}\vec{b}_1,\ \vec{f}_4^* = \eta^{-1}\vec{b}_2,\ \vec{f}_5^* = \beta^{-1}\vec{b}_3,\ \vec{f}_6^* = \beta^{-1}\vec{b}_4,\ \vec{f}_7^* = \vec{b}_7,\ \vec{f}_8^* = \vec{b}_8,\ \vec{f}_9^* = \vec{b}_5,\ \vec{f}_{10}^* = \vec{b}_6.$$

$\mathcal{B}$ chooses a random matrix $A \in \mathbb{Z}_p^{2\times 2}$. With all but negligible probability, A is invertible. It sets $\mathbb{D} = \mathbb{F}_A$ and $\mathbb{D}^* = \mathbb{F}_A^*$, where A is applied as a change of basis matrix to $\vec{f}_9, \vec{f}_{10}$, and $(A^{-1})^t$ is applied as a change of basis matrix to $\vec{f}_9^*, \vec{f}_{10}^*$. We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed, and reveal no information about A.

$\mathcal{B}$ chooses random values $\alpha_1, \alpha_2 \in \mathbb{Z}_p$. This allows it to compute $e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*}$ as $e_n(g^{\vec{b}_9}, g^{\vec{b}_9^*})^{\alpha_1}$, and $e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}_2^*}$ can be computed similarly. Since $\mathcal{B}$ has $g^{\vec{d}_1} = g^{\vec{b}_9^*}$, $g^{\vec{d}_2} = g^{\vec{b}_{10}^*}$, $g^{\vec{d}_3} = g^{\eta\vec{b}_1^*}$, …, $g^{\vec{d}_6} = g^{\beta\vec{b}_4^*}$, $\mathcal{B}$ is able to produce the public parameters

$$\text{PP} = \left\{G, p, e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*}, e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}_2^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\right\}$$

and gives them to $\mathcal{A}$. It also chooses random values $\gamma, \xi, \theta', \sigma' \in \mathbb{Z}_p$. It implicitly sets $\theta = \eta\theta'$ and $\sigma = \beta\sigma'$. This allows it to create the terms $g^{\gamma\vec{d}_1^*} = (g^{\vec{b}_9})^\gamma$, $g^{\xi\vec{d}_2^*} = (g^{\vec{b}_{10}})^\xi$, $g^{\theta\vec{d}_3^*} = (g^{\vec{b}_1})^{\theta'}$, …, $g^{\sigma\vec{d}_6^*} = (g^{\vec{b}_4})^{\sigma'}$ for the secret keys. We note that $\mathcal{B}$ knows the MSK, as well $g^{\vec{d}_7^*} = g^{\vec{b}_7}$ and $g^{\vec{d}_8^*} = g^{\vec{b}_8}$.

For the first $m-1$ key requests, $\mathcal{B}$ responds by creating normal keys using the normal key generation algorithm, and then multiplying the resulting K_j by random powers of $g^{\vec{d}_7^*}, g^{\vec{d}_8^*}$. This produces properly distributed semi-functional keys. To respond to the m^{th} key request for an identity vector $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ makes $K_1, \ldots, K_{j-1}$ as in the normal key generation algorithm. It implicitly sets $r_1^j = \mu_1(\theta')^{-1}$ and $r_2^j = \mu_2(\sigma')^{-1}$. It sets:

$$K_j = g^{y_j\vec{d}_1^* + w_j\vec{d}_2^*} \cdot U_1^{ID_j} \cdot (U_2)^{-1}.$$

This produces an ephemeral semi-functional key, where the coefficients of $g^{\vec{d}_9^*}$ and $g^{\vec{d}_{10}^*}$ are the entries of the vector

$$\mu_3 A^t(ID_j, -1).$$

To respond to the remaining key requests, $\mathcal{B}$ creates normal keys using the normal key generation algorithm.

At some point, $\mathcal{A}$ specifies an identity vector $(ID_1^*, \ldots, ID_{j^*}^*)$ for the challenge ciphertext and messages M_0, M_1. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and encrypts M_b as follows. It chooses random values $s_1, s_2 \in \mathbb{Z}_p$ and sets

$$C_0 = M_b \left(e(g,g)^{\alpha_1 \vec{d}_1 \cdot \vec{d}_1^*}\right)^{s_1} \left(e(g,g)^{\alpha_2 \vec{d}_2 \cdot \vec{d}_2^*}\right)^{s_2}.$$

For i from to 1 to $\ell - 1$, it chooses random values $t_1^i, t_2^i, v_7^i, v_8^i, v_9^i, v_{10}^i \in \mathbb{Z}_p$. It forms:

$$\begin{aligned} C_i &= g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i^* t_1^i \vec{d}_4 + t_2^i \vec{d}_5 + ID_i^* t_2^i \vec{d}_6 + v_7^i \vec{d}_7 + v_8^i \vec{d}_8 + v_9^i \vec{d}_9 + v_{10}^i \vec{d}_{10}} \\ &= (g^{\vec{b}_9^*})^{s_1} \cdot (g^{\vec{b}_{10}^*})^{s_2} \cdot (g^{\eta \vec{b}_1^*})^{t_1^i} \cdot (g^{\eta \vec{b}_2^*})^{ID_i^* t_1^i} \cdot (g^{\beta \vec{b}_3^*})^{t_2^i} \cdot (g^{\beta \vec{b}_4^*})^{ID_i^* t_i^2} \cdot (g^{\vec{b}_7^*})^{v_7^i} \cdot (g^{\vec{b}_8^*})^{v_8^i} \cdot (g^{\vec{b}_5^*})^{v_9^i} \cdot (g^{\vec{b}_6^*})^{v_{10}^i}. \end{aligned}$$

These terms have both semi-functional and ephemeral semi-functional components.

For ℓ, $\mathcal{B}$ chooses random values $v_7^\ell, v_8^\ell \in \mathbb{Z}_p$ and implicitly sets $t_1^\ell = \tau_1$ and $t_2^\ell = \tau_2$. It computes:

$$C_\ell = g^{s_1\vec{d}_1 + s_2\vec{d}_2} \cdot T_1 \cdot T_2^{ID_\ell^*} \cdot g^{v_7^\ell \vec{d}_7 + v_8^\ell \vec{d}_8}.$$

For $i > \ell$, $\mathcal{B}$ chooses random values $t_1^i, t_2^i, v_7^i, v_8^i \in \mathbb{Z}_p$ and sets:

$$C_i = g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i^* t_1^i \vec{d}_4 + t_2^i \vec{d}_5 + ID_i^* t_2^i \vec{d}_6 + v_7^i \vec{d}_7 + v_8^i \vec{d}_8}.$$

If T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^*}$, then C_ℓ has no multiples of $\vec{d}_9, \vec{d}_{10}$ in its exponent. Hence, the ciphertext is distributed as in $\text{Game}_{E(m,\ell-1)}$. If T_1, T_2 are equal to $g^{\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_3^* + \tau_3 \vec{b}_5^*}, g^{\tau_1 \eta \vec{b}_2^* + \tau_2 \beta \vec{b}_4^* + \tau_3 \vec{b}_6^*}$, then the exponent of C_ℓ does include a linear combination of $\vec{d}_9, \vec{d}_{10}$ in the exponent. The coefficients of this combination are equal to the entries of the vector

$$\tau_3 A^{-1}(1, ID_\ell^*).$$

Since we have guaranteed by our encoding of identity vectors that $ID_\ell^* \neq ID_j$, we may invoke Lemma 4 to conclude that the K_j term of the m^{th} key and the C_ℓ term of the ciphertext have coefficients in the ephemeral semi-functional space that are uniformly random in $\mathcal{A}$'s view. Hence, in this case, $\mathcal{B}$ has properly simulated $\text{Game}_{E(m,\ell)}$. Thus, $\mathcal{B}$ can leverage the non-negligible difference in advantage achieved by $\mathcal{A}$ to achieve a non-negligible advantage against the subspace assumption.

□

Lemma 31. *For each m from 1 to q, if there exists a PPT algorithm $\mathcal{A}$ achieving a non-negligible difference in advantage between $\text{Game}_{E(m,j^*)}$ and $\text{Game}_{P(m,j^*)}$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 10$.*

Proof. We will prove this lemma in two steps. We define an intermediary game, denoted $\text{Game}_{EP(m,j^*)}$, in which the K_j term of the m^{th} key has random multiples of all of $\vec{d}_7^*, \vec{d}_8^*, \vec{d}_9^*, \vec{d}_{10}^*$ in its exponent (both ephemeral semi-functional and regular semi-functional terms). In other respects, $\text{Game}_{EP(m,j^*)}$ is identical to $\text{Game}_{E(m,j^*)}$ and $\text{Game}_{P(m,j^*)}$.

We first suppose there exists a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between $\text{Game}_{E(m,j^*)}$ and $\text{Game}_{EP(m,j^*)}$ for some m. We create a PPT algorithm $\mathcal{B}$

achieving non-negligible advantage against the subspace assumption with $k = 1$ and $n = 10$. $\mathcal{B}$ is given

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}^*_1}, g^{\beta\vec{b}^*_2}, g^{\vec{b}^*_3}, \ldots, g^{\vec{b}^*_{10}}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$.

We define dual orthonormal bases $\mathbb{F}, \mathbb{F}^*$ as:

$$\vec{f}_1 = \vec{b}_4,\ \vec{f}_2 = \vec{b}_5,\ \vec{f}_3 = \vec{b}_6,\ \vec{f}_4 = \vec{b}_7,\ \vec{f}_5 = \vec{b}_8,\ \vec{f}_6 = \vec{b}_9,\ \vec{f}_7 = \vec{b}_{10},\ \vec{f}_8 = \vec{b}_3,\ \vec{f}_9 = \vec{b}_2,\ \vec{f}_{10} = \vec{b}_1,$$

$$\vec{f}^*_1 = \vec{b}^*_4,\ \vec{f}^*_2 = \vec{b}^*_5,\ \vec{f}^*_3 = \vec{b}^*_6,\ \vec{f}^*_4 = \vec{b}^*_7,\ \vec{f}^*_5 = \vec{b}^*_8,\ \vec{f}^*_6 = \vec{b}^*_9,\ \vec{f}^*_7 = \vec{b}^*_{10},\ \vec{f}^*_8 = \vec{b}^*_3,\ \vec{f}^*_9 = \vec{b}^*_2,\ \vec{f}^*_{10} = \vec{b}^*_1.$$

$\mathcal{B}$ chooses a random matrix $A \in \mathbb{Z}_p^{2\times 2}$. With all but negligible probability, A is invertible. $\mathcal{B}$ implicitly sets $\mathbb{D} = \mathbb{F}_A$ and $\mathbb{D}^* = \mathbb{F}^*_A$, where A is applied as a change of basis matrix to $\vec{f}_7, \vec{f}_8$ and $(A^{-1})^t$ is applied as a change of basis matrix to $\vec{f}^*_7, \vec{f}^*_8$. We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed, and reveal no information about A.

$\mathcal{B}$ chooses $\alpha_1, \alpha_2, \theta, \sigma, \gamma, \xi \in \mathbb{Z}_p$ for itself, which enables it to produce the public parameters and the master secret key. It gives the public parameters to $\mathcal{A}$. To answer $\mathcal{A}$'s first $m-1$ key requests, $\mathcal{B}$ first produces a normal key using the normal key generation algorithm (it can run this because it knows the master secret key). Now, it chooses random values $z, x \in \mathbb{Z}_p$ and multiplies the final key element K_j by $(g^{\vec{b}^*_{10}})^z(g^{\vec{b}^*_3})^x$. We note that applying the change of basis matrix $(A^{-1})^t$ does not change that the span of $\vec{d}^*_7, \vec{d}^*_8$ is equal to the span of $\vec{b}^*_{10}, \vec{b}^*_3$. Hence, this is distributed as a random linear combination of $\vec{d}^*_7, \vec{d}^*_8$ and so the resulting key is a properly distributed semi-functional key.

To create the m^{th} key for $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ first creates a normal key by running the normal key generation algorithm. It then multiplies K_j by T_1. If $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$, then this adds a random linear combination of $\vec{d}^*_9, \vec{d}^*_{10}$ to the exponent of K_j. In this case, the key is properly distributed as in $\text{Game}_{E(m,j^*)}$. If $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$, then this adds a random linear combination of $\vec{d}^*_9, \vec{d}^*_{10}$ as well as $\tau_3\vec{b}^*_3$. Because we have employed the *random* change of basis matrix $(A^{-1})^t$, $\vec{b}^*_3$ is itself a random linear combination of $\vec{d}^*_7, \vec{d}^*_8$. Therefore, the key is properly distributed as in $\text{Game}_{EP(m,j^*)}$ in this case. (We note that this is the only place where we will use the randomness of A: everything else will be distributed independently of A.)

When $\mathcal{A}$ declares $(ID^*_1, \ldots, ID^*_{j^*})$ and M_0, M_1, $\mathcal{B}$ creates the challenge ciphertext as follows. It chooses a random bit $b \in \{0, 1\}$, random values $s_1, s_2 \in \mathbb{Z}_p$, as well as random values $t^i_1, t^i_2, \nu^i_7, \nu^i_8, \nu^i_9, \nu^i_{10} \in \mathbb{Z}_p$ for each i from 1 to j^*. It computes:

$$C_0 = M_b(e_n(g^{\vec{b}_4}, g^{\vec{b}^*_4}))^{\alpha_1 s_1}(e_n(g^{\vec{b}_5}, g^{\vec{b}^*_5}))^{\alpha_2 s_2} = M_b e(g,g)^{\alpha_1 s_1 \vec{d}_1\cdot\vec{d}^*_2} e(g,g)^{\alpha_2 s_2 \vec{d}_2\cdot\vec{d}^*_2}$$

$$C_i = (g^{\vec{b}_4})^{s_1} \cdot (g^{\vec{b}_5})^{s_2} \cdot (g^{\vec{b}_6})^{t^i_1} \cdot (g^{\vec{b}_7})^{ID^*_i t^i_1} \cdot (g^{\vec{b}_8})^{t^i_2} \cdot (g^{\vec{b}_9})^{ID^*_i t^i_2} \cdot (g^{\vec{b}_{10}})^{\nu^i_7} \cdot U_1^{\nu^i_8} \cdot (g^{\vec{b}_2})^{\nu^i_9} \cdot (g^{\vec{b}_1})^{\nu^i_{10}}$$

for each i from 1 to j^*. We observe that the exponent of C_i is equal to $s_1\vec{d}_1 + s_2\vec{d}_2 + t^i_1\vec{d}_3 + ID^*_i t^i_1\vec{d}_4 + t^i_2\vec{d}_5 + ID^*_i t^i_2\vec{d}_6$ plus a linear combination of $\vec{d}_7, \vec{d}_8, \vec{d}_9, \vec{d}_{10}$. The coefficients of $\vec{d}_7$ and $\vec{d}_8$ here are equal to the entries of the vector $A^{-1}(\nu^i_7, \mu_3\nu^i_8)$, which are uniformly random because ν^i_7 and ν^i_8 are random (note that these are distributed independently of A^{-1}). The coefficients of $\vec{d}_9 = \vec{b}_2$ and $\vec{d}_{10} = \vec{b}_1$ here are $\mu_2 + \nu^i_9$ and $\mu_1 + \nu^i_{10}$, which are also uniformly random because ν^i_9 and ν^i_{10} are uniformly random. Hence the ciphertext is properly distributed.

In summary, $\mathcal{B}$ has properly simulated either $\text{Game}_{E(m,j^*)}$ or $\text{Game}_{EP(m,j^*)}$, depending on the value of T_1. This allows it to leverage the non-negligible difference in $\mathcal{A}$'s advantage between these games to achieve non-negligible advantage against the subspace assumption.

An analogous argument can be used to transition from $\text{Game}_{EP(m,j^*)}$ to $\text{Game}_{P(m,j^*)}$. The only difference is that one now sets $\vec{f}_7 = \vec{b}_1$, $\vec{f}_8 = \vec{b}_2$, and $\vec{f}_9 = \vec{b}_3$. One then applies a random change of basis matrix A to $\vec{f}_9, \vec{f}_{10}$. Now T_1 is multiplied by the K_j term of the m^{th} key so that when the exponent of T_1 is in the span of $\vec{b}_1^*, \vec{b}_2^*$, the key will only have components in the regular semi-functional space, and no appearance of $\vec{d}_9, \vec{d}_{10}$ terms. When the exponent of T_1 includes a multiple of $\vec{b}_3^*$, then the key will have random components in both the regular semi-functional and ephemeral semi-functional spaces. □

Lemma 32. *For each m from 1 to q and each ℓ from 1 to j^*, if there exists a PPT algorithm $\mathcal{A}$ achieving a non-negligible difference in advantage between $\text{Game}_{P(m,\ell)}$ and $\text{Game}_{P(m,\ell-1)}$, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 2$ and $n = 10$.*

Proof. This is identical to the proof of Lemma 30, except that the m^{th} key is now made to be semi-functional, just as the first $m-1$ keys. In this case, the argument that the coefficients of $\vec{d}_9, \vec{d}_{10}$ on C_ℓ are well-distributed when they are present is simpler (does not require pairwise independence), since there are no appearances of $\vec{d}_9^*, \vec{d}_{10}^*$ in any of the keys. □

Finally, we transition to a Game_{Final} in which the ciphertext encrypts a random message. Thus, the bit b will be completely hidden from the attacker, and it follows that any attacker has 0 advantage. We prove this in a few stages. First, we expand the final K_j of each key to additionally include a multiple of $\vec{d}_{10}^*$ in the exponent. This multiple will be the *same* for every key. Next, we expand all of $C_1, \ldots, C_{j^*}$ to include random multiples of $\vec{d}_{10}$ in the exponent (these multiples will differ). Finally, we move from an encryption of M_b to an encryption of a random message by one last application of the subspace assumption. In this last step, we will (roughly) set $\vec{d}_1^*, \vec{d}_2^*$ to be $\vec{b}_1, \vec{b}_2$ and $\vec{d}_1, \vec{d}_2$ to be $\vec{b}_1^*, \vec{b}_2^*$ (this will be true up to scalar adjustments). We will set $\vec{d}_{10}^* = \vec{b}_3$ and $\vec{d}_{10} = \vec{b}_3^*$. This allows us to implicitly set $\alpha_1\vec{d}_1^* + \alpha_2\vec{d}_2^*$ to be $\mu_1\vec{b}_1 + \mu_2\vec{b}_2$, which the subspace assumption only gives us attached to $\mu_3\vec{b}_3$. We implicitly set $s_1\vec{d}_1 + s_2\vec{d}_2$ to be $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$, which we may receive alone in the exponent, or attached to $\tau_3\vec{b}_3^*$. We compute the blinding factor as $e_n(U_1, T_1)$. If $T_1 = g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*}$, then this is computing $(\alpha_1 s_1 + \alpha_2 s_2)\psi$ in the exponent, which is the proper blinding factor. However, if T_1 additionally includes $g^{\tau_3\vec{b}_3^*}$, then we get an extra contribution of $\mu_3\tau_3$ in the exponent - this will look random, since we can hide the value of τ_3 by inserting additional random multiples of $\vec{b}_3^*$ in the exponents of the ciphertext elements. In this case, we have a random blinding factor, which is equivalent to encrypting a random message.

More formally, we define the following additional games:

Game_{SFK+} This game is like $\text{Game}_{P(q,0)}$ (all keys and ciphertext are semi-functional), except that for each key, the final K_j exponent also includes a multiple of $\vec{d}_{10}^*$. This multiple is the same for every key.

Game_{SFCT+} This game is like Game_{SFK+} except that each of $C_1, \ldots, C_{j^*}$ in the ciphertext includes a fresh random multiple of $\vec{d}_{10}$ in its exponent.

Game_{Final} This game is like Game_{SKCT+} except that the message being encrypted is now a random element of G_T, independent of the bit b.

Lemma 33. *If there exists a PPT algorithm $\mathcal{A}$ achieving a non-negligible difference in advantage between $\text{Game}_{P(q,0)}$ and Game_{SFK+}, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 10$.*

Proof. $\mathcal{B}$ is given

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}^*_1}, g^{\beta\vec{b}^*_2}, g^{\vec{b}^*_3}, \ldots, g^{\vec{b}^*_{10}}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$ or as $g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2+\tau_3\vec{b}^*_3}$.

$\mathcal{B}$ implicitly sets:

$$\vec{d}_1 = \vec{b}_1,\ \vec{d}_2 = \vec{b}_2,\ \vec{d}_3 = \vec{b}_4,\ \ldots,\ \vec{d}_9 = \vec{b}_{10},\ \vec{d}_{10} = \vec{b}_3,$$

$$\vec{d}^*_1 = \vec{b}^*_1,\ \vec{d}^*_2 = \vec{b}^*_2,\ \vec{d}^*_3 = \vec{b}^*_4,\ \ldots, \vec{d}^*_9 = \vec{b}^*_{10},\ \vec{d}^*_{10} = \vec{b}^*_3.$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed.

$\mathcal{B}$ implicitly defines $\alpha_1 = \tau_1\eta$ and $\alpha_2 = \tau_2\beta$. We note that $\mathcal{B}$ does not know α_1, α_2, but it can compute $e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1}$ as:

$$e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1} = e_n(T_1, g^{\vec{b}_1}),$$

and can similarly compute $e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}^*_2}$ as $e_n(T_1, g^{\vec{b}_2})$. This allows $\mathcal{B}$ to produce the public parameters:

$$\text{PP} = \left\{G, p, e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1}, e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}^*_2}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\right\}.$$

It gives these to $\mathcal{A}$.

To create secret keys, $\mathcal{B}$ chooses random exponents $\theta, \sigma \in \mathbb{Z}_p$ and implicitly sets $\gamma = \eta$ and $\xi = \beta$. When $\mathcal{A}$ requests a key for an identity vector $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ creates a key as follows. It chooses random values $r^i_1, r^i_2 \in \mathbb{Z}_p$ for each i from 1 to j, as well as random values $y'_1, \ldots, y'_j \in \mathbb{Z}_p$, $w'_1, \ldots, w'_j \in \mathbb{Z}_p$ subject to the constraints $y'_1 + \cdots + y'_j = 0 = w_1 + \cdots + w'_j$. For each i from 1 to $j-1$, it will implicitly set $y_i = y'_i\eta$ and $w_i = w'_i\beta$. For j, it sets $y_j = y'_j\eta + \tau_1\eta$ and $w_j = w'_j\beta + \tau_2\beta$. We note that $y_1, \ldots, y_j$ are distributed as random elements of $\mathbb{Z}_p$ up to the constraint that $y_1 + \cdots + y_j = \alpha_1$, and $w_1, \ldots, w_j$ are distributed as random elements of $\mathbb{Z}_p$ up to the constraint that $w_1 + \cdots + w_j = \alpha_2$. $\mathcal{B}$ computes:

$$\begin{aligned} K_i &= g^{y_i\vec{d}^*_1 + w_i\vec{d}^*_2 + r^i_1 ID_i\theta\vec{d}^*_3 - r^i_1\theta\vec{d}^*_4 + r^i_2 ID_i\sigma\vec{d}^*_5 - r^i_2\sigma\vec{d}^*_6} \\ &= (g^{\eta\vec{b}^*_1})^{y'_i} \cdot (g^{\beta\vec{b}^*_2})^{w'_i} \cdot (g^{\vec{b}^*_4})^{r^i_1 ID_i\theta} \cdot (g^{\vec{b}^*_5})^{-r^i_1\theta} \cdot (g^{\vec{b}^*_6})^{r^i_2 ID_i\sigma} \cdot (g^{\vec{b}^*_7})^{-r^i_2\sigma}. \end{aligned}$$

It also chooses random $z_7, z_8 \in \mathbb{Z}_p$ and computes:

$$K_j = T_1 \cdot (g^{\eta\vec{b}^*_1})^{y'_j} \cdot (g^{\beta\vec{b}^*_2})^{w'_j} \cdot (g^{\vec{b}^*_4})^{r^i_1 ID_i\theta} \cdot (g^{\vec{b}^*_5})^{-r^i_1\theta} \cdot (g^{\vec{b}^*_6})^{r^i_2 ID_i\sigma} \cdot (g^{\vec{b}^*_7})^{-r^i_2\sigma} \cdot (g^{\vec{b}^*_8})^{z_7} \cdot (g^{\vec{b}^*_9})^{z_8}.$$

If $T_1 = g^{\tau_1\eta\vec{b}^*_1+\tau_2\beta\vec{b}^*_2}$, then

$$K_j = g^{y_j\vec{d}^*_1 + w_j\vec{d}^*_2 + r^j_1 ID_j\theta\vec{d}^*_3 - r^j_1\theta\vec{d}^*_4 + r^j_2 ID_j\sigma\vec{d}^*_5 - r^j_2\sigma\vec{d}^*_6 + z_7\vec{d}^*_7 + z_8\vec{d}^*_8}.$$

In this case, $\mathcal{B}$ produces keys that are distributed as in $\text{Game}_{P(q,0)}$. If T_1 additionally has $\tau_3\vec{b}^*_3$ in its exponent, then every K_j will have $\tau_3\vec{d}^*_{10}$ in its exponent. In this case, $\mathcal{B}$ produces keys that are distributed as in Game_{SFK+}.

At some point, $\mathcal{A}$ declares a challenge identity vector $(ID^*_1, \ldots, ID^*_{j^*})$ and two messages M_0, M_1. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and produces a semi-functional encryption of M_b as follows. $\mathcal{B}$ chooses random values $s_1, s_2 \in \mathbb{Z}_p$ and random values $t^i_1, t^i_2, v^i_7, v^i_8 \in \mathbb{Z}_p$ for each i from 1 to j^*. It computes:

$$C_0 = M_b \left(e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}^*_1}\right)^{s_1} \left(e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}^*_2}\right)^{s_2},$$

and for each i from 1 to j^*:

$$\begin{aligned} C_i &= g^{s_1\vec{d}_1+s_2\vec{d}_2+t_1^i\vec{d}_3+ID_i^*t_1^i\vec{d}_4+t_2^i\vec{d}_5+ID_i^*t_2^i\vec{d}_6+v_7^i\vec{d}_7+v_8^i\vec{d}_8} \\ &= (g^{\vec{b}_1})^{s_1}\cdot(g^{\vec{b}_2})^{s_2}\cdot(g^{\vec{b}_4})^{t_1^i}\cdot(g^{\vec{b}_5})^{ID_i^*t_1^i}\cdot(g^{\vec{b}_6})^{t_2^i}\cdot(g^{\vec{b}_7})^{ID_i^*t_2^i}\cdot(g^{\vec{b}_8})^{v_7^i}\cdot(g^{\vec{b}_9})^{v_8^i}. \end{aligned}$$

This is a properly distributed semi-functional ciphertext.

If $T_1 = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*}$, then $\mathcal{B}$ has properly simulated $\text{Game}_{P(q,0)}$. If $T_1 = g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*+\tau_3\vec{b}_3^*}$, then $\mathcal{B}$ has properly simulated Game_{SFK+}. Hence, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the subspace assumption. □

Lemma 34. *If there exists a PPT algorithm $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{SFK+} and Game_{SFCT+}, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 10$.*

Proof. We assume such a $\mathcal{A}$ exists, and we create $\mathcal{B}$ as follows. $\mathcal{B}$ is given

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, \ldots, g^{\vec{b}_{10}^*}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*}$ or as $g^{\tau_1\eta\vec{b}_1^*+\tau_2\beta\vec{b}_2^*+\tau_3\vec{b}_3^*}$.

$\mathcal{B}$ implicitly sets:

$$\vec{d}_1 = \vec{b}_4^*,\ \vec{d}_2 = \vec{b}_5^*,\ \vec{d}_3 = \vec{b}_6^*,\ \vec{d}_4 = \vec{b}_7^*,\ \vec{d}_5 = \vec{b}_8^*, \vec{d}_6 = \vec{b}_9^*,\ \vec{d}_7 = \vec{b}_1^*,\ \vec{d}_8 = \vec{b}_2^*,\ \vec{d}_9 = \vec{b}_{10}^*,\ \vec{d}_{10} = \vec{b}_3^*,$$

$$\vec{d}_1^* = \vec{b}_4,\ \vec{d}_2^* = \vec{b}_5,\ \vec{d}_3^* = \vec{b}_6,\ \vec{d}_4^* = \vec{b}_7,\ \vec{d}_5^* = \vec{b}_8,\ \vec{d}_6^* = \vec{b}_9,\ \vec{d}_7^* = \vec{b}_1,\ \vec{d}_8^* = \vec{b}_2,\ \vec{d}_9^* = \vec{b}_{10},\ \vec{d}_{10}^* = \vec{b}_3.$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed.

$\mathcal{B}$ chooses random values $\alpha_1, \alpha_2, \theta, \sigma, \gamma, \xi \in \mathbb{Z}_p$. This allows it to produce the public parameters:

$$\text{PP} = \left\{G, p, e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*}, e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}_2^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\right\}.$$

We observe that $\mathcal{B}$ also knows the MSK and can produce $g^{\gamma\vec{d}_1^*}, g^{\xi\vec{d}_2^*}, g^{\theta\vec{d}_3^*}, g^{\theta\vec{d}_4^*}, g^{\sigma\vec{d}_5^*}, g^{\sigma\vec{d}_6^*}$ for the secret keys.

When $\mathcal{A}$ requests a key for an identity vector $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ creates $K_1, \ldots, K_j$ as follows. First, it creates $K_1, \ldots, K_{j-1}, K_j'$ as in the normal key generation algorithm. It then chooses random values $z_7, z_8 \in \mathbb{Z}_p$ and sets:

$$K_j = K_j' \cdot U_1 \cdot (g^{\vec{b}_1})^{z_7} \cdot (g^{\vec{b}_2})^{z_8}.$$

This results in K_j values whose exponents include uniformly random linear combinations of $\vec{d}_7^* = \vec{b}_1$ and $\vec{d}_8^* = \vec{b}_2$ as well as $\mu_3\vec{d}_{10}^* = \mu_3\vec{b}_3$, where this coefficient μ_3 is the *same* for all keys. This produces keys which are properly distributed for either Game_{SFK+} or Game_{SFCT+}.

When $\mathcal{A}$ specifies $(ID_1^*, \ldots, ID_{j^*}^*)$ and M_0, M_1 for the challenge ciphertext, $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$. It first computes a normal ciphertext $C_0, C_1', \ldots, C_{j^*}'$ encrypting M_b using the normal encryption algorithm. It then chooses random values $\nu_7^i, \nu_8^i, \nu_{10}^i \in \mathbb{Z}_p$ for each i from 1 to j^*. For each i, it sets:

$$C_i = C_i' \cdot (T_1)^{\nu_{10}^i} \cdot (g^{\eta\vec{b}_1^*})^{\nu_7^i} \cdot (g^{\beta\vec{b}_2^*})^{\nu_8^i}.$$

If the exponent of T_1 is a linear combination of $\vec{b}_1^* = \vec{d}_7$ and $\vec{b}_2^* = \vec{d}_8$, then each C_i will have a random linear combination of $\vec{d}_7, \vec{d}_8$ in its exponent, making it properly distributed for

Game_{SFK+}. If the exponent of T_1 is a linear combination of $\vec{b}_1^* = \vec{d}_7$, $\vec{b}_2^* = \vec{d}_8$, and $\vec{b}_3^* = \vec{d}_{10}$, then each C_i will have a random linear combination of $\vec{d}_7$, $\vec{d}_8$, and $\vec{d}_{10}$, making it properly distributed for Game_{SFCT+}.

Therefore, $\mathcal{B}$ has properly simulated either Game_{SFK+} or Game_{SFCT+}, depending on the value of T_1. $\mathcal{B}$ can thus leverage the non-negligible difference in $\mathcal{A}$'s advantage between two games to achieve a non-negligible advantage against the subspace assumption.

□

Lemma 35. *If there exists a PPT algorithm $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{SKCT+} and Game_{Final}, then there exists a PPT algorithm $\mathcal{B}$ with non-negligible advantage against the subspace assumption, with $k = 1$ and $n = 10$.*

Proof. $\mathcal{B}$ is given

$$D = \left(g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_4}, \ldots, g^{\vec{b}_{10}}, g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, \ldots, g^{\vec{b}_{10}^*}, U_1, \mu_3\right),$$

along with T_1. It is $\mathcal{B}$'s task to decide whether T_1 is distributed as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*}$ or as $g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*}$.

$\mathcal{B}$ implicitly sets:

$$\vec{d}_1 = \eta\vec{b}_1^*,\ \vec{d}_2 = \beta\vec{d}_2^*,\ \vec{d}_3 = \vec{b}_4^*,\ \ldots,\ \vec{d}_9 = \vec{b}_{10}^*,\ \vec{d}_{10} = \vec{b}_3^*,$$

$$\vec{d}_1^* = \eta^{-1}\vec{b}_1,\ \vec{d}_2^* = \beta^{-1}\vec{b}_2,\ \vec{d}_3^* = \vec{b}_4,\ \ldots,\ \vec{d}_9^* = \vec{b}_{10},\ \vec{d}_{10}^* = \vec{b}_3.$$

We note that $\mathbb{D}, \mathbb{D}^*$ are properly distributed and that $\mathcal{B}$ can produce $g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}$ for the public parameters.

$\mathcal{B}$ implicitly sets $\alpha_1 = \eta\mu_1$ and $\alpha_2 = \beta\mu_2$. It can then compute $e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*}$ as:

$$e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*} = e_n(U_1, g^{\eta\vec{b}_1^*}).$$

It similarly computes $e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}_2^*} = e_n(U_1, g^{\beta\vec{b}_2^*})$. It provides $\mathcal{A}$ with the public parameters,

$$\text{PP} = \left\{G, p, e(g,g)^{\alpha_1\vec{d}_1\cdot\vec{d}_1^*}, e(g,g)^{\alpha_2\vec{d}_2\cdot\vec{d}_2^*}, g^{\vec{d}_1}, \ldots, g^{\vec{d}_6}\right\}.$$

$\mathcal{B}$ chooses random values $\theta, \sigma, \gamma', \xi' \in \mathbb{Z}_p$ and sets $\gamma = \eta^{-1}\gamma'$ and $\xi = \beta^{-1}\xi'$. It can then form $g^{\gamma\vec{d}_1^*} = (g^{\vec{b}_1})^{\gamma'}$, $g^{\xi\vec{d}_2^*} = (g^{\vec{b}_2})^{\xi'}$, $g^{\theta\vec{d}_3^*} = (g^{\vec{b}_4})^{\theta}$, $g^{\theta\vec{d}_4^*} = (g^{\vec{b}_5})^{\theta}$, $g^{\sigma\vec{d}_5^*} = (g^{\vec{b}_6})^{\sigma}$, and $g^{\sigma\vec{d}_6^*} = (g^{\vec{b}_7})^{\sigma}$.

When $\mathcal{A}$ requests a key for an identity vector $(ID_1, \ldots, ID_j)$, $\mathcal{B}$ computes the additional key elements as follows. It chooses random values $r_1^i, r_2^i \in \mathbb{Z}_p$ for each i from 1 to j, random values $z_7, z_8 \in \mathbb{Z}_p$, and random values $y_1', \ldots, y_j', w_1', \ldots, w_j' \in \mathbb{Z}_p$ subject to the constraint that $y_1' + \cdots + y_j' = w_1' + \cdots + w_j' = 0$. It implicitly sets $y_i = \eta y_i'$, $w_i = \beta w_i'$ for each i from 1 to $j-1$, and $y_j = \eta y_j' + \eta\mu_1$, $w_j = \beta w_j' + \beta\mu_2$. We note that $y_1, \ldots, y_j$ are randomly distributed up to the constraint that $y_1 + \cdots + y_j = \alpha_1$, and $w_1, \ldots, w_j$ are randomly distributed up to the constraint that $w_1 + \cdots + w_j = \alpha_2$. $\mathcal{B}$ can then compute:

$$\begin{aligned} K_i &= g^{y_i\vec{d}_1^* + w_i\vec{d}_2^* + r_1^i ID_i\theta\vec{d}_3^* - r_1^i\theta\vec{d}_4^* + r_2^i ID_i\sigma\vec{d}_5^* - r_2^i\sigma\vec{d}_6^*} \\ &= (g^{\vec{b}_1})^{y_i'} \cdot (g^{\vec{b}_2})^{w_i'} \cdot (g^{\vec{b}_4})^{r_1^i ID_i\theta} \cdot (g^{\vec{b}_5})^{-r_1^i\theta} \cdot (g^{\vec{b}_6})^{r_2^i ID_i\sigma} \cdot (g^{\vec{b}_7})^{-r_2^i\sigma} \end{aligned}$$

for all i from 1 to $j-1$, and

$$\begin{aligned} K_j &= g^{y_j\vec{d}_1^* + w_j\vec{d}_2^* + r_1^j ID_j\theta\vec{d}_3^* - r_1^j\theta\vec{d}_4^* + r_2^j ID_j\sigma\vec{d}_5^* - r_2^j\sigma\vec{d}_6^* + z_7\vec{d}_7^* + z_8\vec{d}_8^* + \mu_3\vec{d}_{10}^*} \\ &= U_1 \cdot (g^{\vec{b}_1})^{y_j'} \cdot (g^{\vec{b}_2})^{w_j'} \cdot (g^{\vec{b}_4})^{r_1^j ID_j\theta} \cdot (g^{\vec{b}_5})^{-r_1^j\theta} \cdot (g^{\vec{b}_6})^{r_2^j ID_j\sigma} \cdot (g^{\vec{b}_7})^{-r_2^j\sigma} \cdot (g^{\vec{b}_8})^{z_7} \cdot (g^{\vec{b}_9})^{z_8}. \end{aligned}$$

The keys that $\mathcal{B}$ produces are properly distributed for Game_{SFCT+} and Game_{Final}.

At some point, $\mathcal{A}$ declares an identity vector $(ID_1^*, \ldots, ID_{j^*}^*)$ and messages M_0, M_1 for the challenge ciphertext. $\mathcal{B}$ chooses a random bit $b \in \{0,1\}$ and creates a ciphertext which is either an encryption of M_b or an encryption of a random message, depending on the value of T_1. To do this, it implicitly sets $s_1 = \tau_1$ and $s_2 = \tau_2$. It chooses random values $t_1^i, t_2^i, v_7^i, v_8^i, \nu_{10}^i \in \mathbb{Z}_p$ for each i from 1 to j^*. It computes:

$$C_0 = M_b e_n(U_1, T_1),$$

and for each i from 1 to j^*:

$$\begin{aligned} C_i &= g^{s_1\vec{d}_1 + s_2\vec{d}_2 + t_1^i\vec{d}_3 + ID_i^* t_1^i \vec{d}_4 + t_2^i\vec{d}_5 + ID_i^* t_2^i \vec{d}_6 + v_7^i\vec{d}_7 + v_8^i\vec{d}_8 + v_{10}^i\vec{d}_{10}} \\ &= T_1 \cdot (g^{\vec{b}_4^*})^{t_1^i} \cdot (g^{\vec{b}_5^*})^{ID_i^* t_1^i} \cdot (g^{\vec{b}_6^*})^{t_2^i} \cdot (g^{\vec{b}_7^*})^{ID_i^* t_2^i} \cdot (g^{\vec{b}_8^*})^{v_7^i} \cdot (g^{\vec{b}_9^*})^{v_8^i} \cdot (g^{\vec{b}_3^*})^{\nu_{10}^i}. \end{aligned}$$

We note that the value of each v_{10}^i here depends on the nature of T_1.

Now, if $T_1 = g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*}$, then $C_0 = M_b e(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*}$. In this case, the ciphertext is a properly distributed encryption of M_b, as required in Game_{SFCT+}. However, if $T_1 = g^{\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*}$, then

$$C_0 = M_b e(g,g)^{\alpha_1 s_1 \vec{d}_1 \cdot \vec{d}_1^*} e(g,g)^{\alpha_2 s_2 \vec{d}_2 \cdot \vec{d}_2^*} e(g,g)^{\tau_3 \mu_3 \psi},$$

where $\psi = \vec{b}_i \cdot \vec{b}_i^*$. Here, τ_3 is uniformly random. To see this, observe that the coefficient of $\vec{d}_{10} = \vec{b}_3^*$ in each C_i is equal to $\nu_{10}^i + \tau_3$ in this case, which does not reveal τ_3 because each ν_{10}^i is random. Thus, the ciphertext is distributed as an encryption of a *random* message, independent of M_b, as required in Game_{Final}. Hence, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the subspace assumption.

□