Bilinear Entropy Expansion from the Decisional Linear Assumption

Lucas Kowalczyk
Columbia University
luke@cs.columbia.edu

Allison Bishop Lewko*
Columbia University
alewko@cs.columbia.edu

Abstract

We develop a technique inspired by pseudorandom functions that allows us to increase the entropy available for proving the security of dual system encryption schemes under the Decisional Linear Assumption. We show an application of the tool to Attribute-Based Encryption by presenting a Key-Policy ABE scheme that is fully-secure under DLIN which exhibits an exponential improvement over the state of the art schemes in terms of public parameter size.

1 Introduction

Since its conception in [34], attribute-based encryption (ABE) has served as a demonstrably fertile ground for exploring the possible tradeoffs between expressibility, security, and efficiency in cryptographically enforced access control. In addition to the potential applications it has in its own right, the primitive of attribute-based encryption has been a catalyst for the definitions and constructions of further cryptographic primitives, such as functional encryption for general circuits. The rich structure of secret keys demanded by expressive attribute-based encryption has promoted a continuing evolution of proof techniques designed to meet the challenges inherent in balancing large and complex structures on the pinhead of simple computational hardness assumptions.

The origins of attribute-based encryption can be traced back to identity-based encryption [11, 5], where users have identities that serve as public keys and secret keys are generated on demand by a master authority. A desirable notion of security for such schemes ensures resilience against arbitrary collusions among users by allowing an attacker to demand many secret keys for individual users and attack a ciphertext encrypted to any user not represented in the set of obtained keys. Proving this kind of security requires a reduction design that can satisfy the attacker's demands without fully knowing the master secret key. This challenge is exacerbated in the (key-policy) attribute-based setting, where user keys correspond to access policies expressed over attributes and ciphertexts are associated with subsets of these attributes. Decryption is allowed precisely when a single user's policy is satisfied by a ciphertext's attribute set. Thus, the structure of allowable keys that the attacker can request grows more complex as the scheme is equipped to express more complex policies.

As a consequence of this, the intuitive and elegant constructions of attribute-based encryption in bilinear groups in [18, 36] were only proven secure in the selective security model: a weakened model of security that requires the attacker to declare the target of attack in advance, before seeing the public parameters of the system. This limitation of the model allows

*Allison Lewko is supported in part by NSF CNS 1413971 and NSF CCF 1423306.

the security reduction to embed the computational challenge into its view of the public parameters of the scheme in a way that partitions the space of secret keys. Keys that do not satisfy the targeted ciphertext are able to be generated under the embedding, while keys that do satisfy the ciphertext cannot be generated. This approach does not extend well to the full security model, where this artificial limitation on the attacker is lifted.

The first fully secure ABE schemes appeared in [19], using the dual system encryption methodology [35] for designing the security reduction. In a dual system approach, there are typically multiple (computationally indistinguishable) forms of keys and ciphertexts. There are "normal" keys and ciphertexts that are employed in the real system, and then are various forms of "semi-functional" keys and ciphertexts. The core idea is to prove security via a hybrid argument, where the ciphertext is changed to semi-functional and keys are changed to semi-functional types one by one, until all the keys are of a semi-functional type incapable of decrypting the semi-functional ciphertext (it is important that they still decrypt normal ciphertexts, otherwise the hybrid transitions could be detected by the attacker who can create normal ciphertexts for itself using the public parameters). Once we reach a state where the key and ciphertexts distributions provided to the attacker are no longer bound by correct decrypt behavior, it is much easier for the reduction to produce these without knowing the master secret key.

The most critical step of these dual system arguments occurs when a particular key changes from a type that can decrypt the challenge ciphertext to a type that cannot - the fact that this change is not detected by the attacker is where the reduction must use the criterion that the access policy is not satisfied. The security reductions in [19] and many subsequent works (e.g. [30, 22]) used an information-theoretic argument for this step. However, this argument requires a great deal of entropy (specifically, fresh randomness for each attribute-use in a policy). This entropy was supplied by parameters in the semi-functional space that paralleled the published parameters of the normal space. This necessitated a blowup in public parameter and ciphertext sizes, specifically a multiplicative factor of the the number of attribute-uses allowed by the scheme within individual policies.

In [26], it was observed that the initial steps of a typical dual system encryption hybrid argument could be re-interpreted as providing a "shadow copy" of the system parameters in the semi-functional space that does not have to be committed to when the public parameters for the normal space are provided. This perspective suggests that one can embed a computational challenge into these semi-functional space parameters as semi-functional objects are produced. For instance, when a portion of these parameters affect a single semi-functional key that is queried after the semi-functional ciphertext, one can essentially embed the challenge in the same way as the original selective security arguments in [18]. In the reverse case, where the semi-functional key is queried before the challenge ciphertext, the embedding can be similar to a selective security proof for a ciphertext-policy ABE scheme, where keys are associated with attributes and ciphertexts are associated with access policies. In [26], state of the art selective techniques for KP-ABE and CP-ABE systems were combined into a full security proof, avoiding the blowup in parameters incurred by the information-theoretic dual system techniques.

However, even selective security for CP-ABE systems remains a rather challenging task, and the state of the art technique in [36] introduces an undesirable q-type assumption into the fully secure ABE scheme. In the CP-ABE setting, selectivity means that the attacker declares a target access policy up front. This can then be leveraged by the security reduction to design public parameters so that it can create keys precisely for sets of attributes that do not satisfy this target policy. The q-type assumption in [36] was a consequence of the need to encode a potentially large access policy into small public parameters. This leaves us still searching for an ideal KP-ABE scheme in the bilinear setting that has parameter sizes comparable to the

selectively secure scheme in [18] and a full security proof from a simple assumption such as the decisional linear assumption (DLIN). A security reduction for such a scheme must seemingly break outside the mold of using either a purely information-theoretic or purely computational argument for leveraging the fact that a requested key policy cannot be satisfied by the challenge ciphertext.

Our Results To demonstrate our approach, we present two KP-ABE constructions, one in the composite-order bilinear setting and one in the prime-order setting. Both schemes are proven fully secure from simple assumptions, and support LSSS/MSP access policies (like their bilinear predecessors). In the composite-order setting, we use a few specific instances of subgroup-decision assumptions and DLIN, and in the prime-order setting we rely only on DLIN. Our schemes greatly reduce the size of the public parameters as compared to [19, 30], as the number of group elements we need to include in the public parameters grows only logarithmically rather than than linearly in the bound on the number of attribute-uses in an access policy.

Our Techniques We intermix the computational and information-theoretic dual system encryption approaches, using computational steps to "boost" the entropy of a small set of (unpublished) semi-functional parameters to a level that suffices to make the prior information-theoretic argument work. Essentially, we use the fact that the semi-functional space parameters are never published to not only "delay" their definition as exploited in [26], but further to argue that they can (computationally) appear to provide more entropy than their size would information-theoretically allow. The gadget that allows us do this computational pre-processing before the running information-theoretic argument is presented as our "bilinear entropy expansion lemma."

The inspiration for the gadget construction comes from pseudorandom generators/pseudorandom functions. Naturally, if we want a small set of semi-functional generators to seemingly produce a large amount of entropy, we may want to view these parameters as the seed for a PRF, for example. Out-of-the-box PRF constructions like Naor-Reingold [27] and its DLIN-based extension [25] however are unsuitable in the bilinear setting (even though the DLIN version would remain secure) because they would require direct access to the seed for computation, and a secure bilinear construction will only provide indirect access to the seed as exponents of group elements.

To circumvent this difficulty, we use a subset-sum based construction that can be computed in a bilinear group with the seed elements in the exponents. Of course, using a naked linear structure would be detectable, but we are able to sprinkle in a rather minimal amount of additional random exponents to push the linear sub-structure out of reach of detection by regular group or pairing operations.

We build our construction in two steps. First, we present a construction for a *one-use* KP-ABE system which only supports access policies where each attribute is used at most once. This scheme achieves ciphertext and key sizes which rival those of selectively secure schemes (up to constants), while significantly reducing public parameter size. Then, we apply a standard transformation to get from a one-use system to a system which allows multiple uses of attributes in policies (the number of uses allowed per attribute is constant and fixed at setup). The overhead of this transformation is drastically mitigated by our scheme's small public parameters. The effect on ciphertext and key sizes compared to previous applications of this transformation remain the same up to constants.

1.1 Other Related Work

Additional work on ABE in the bilinear setting includes various constructions of KP-ABE and CP-ABE schemes (e.g. [4, 33, 17]), schemes supporting multiple authorities (e.g. [7, 8, 32, 22]), and schemes supporting large attribute universes (e.g. [23, 31]). Some of the structure for randomization in our schemes is inspired by [23].

There are also recent constructions of ABE schemes in the lattice setting. The construction of [16] allows access policies to be expressed as circuits, which makes it more expressive than any known bilinear scheme. It was proven selectively secure under the standard LWE assumption. Circuit policies are also supported by the construction in [13] based on multilinear maps. This scheme is also proven selectively secure, under a particular computational hardness assumption for multilinear groups. The very recent multilinear scheme in [14] achieves full security, relying on computational hardness assumptions in multilinear groups. The fully secure general functional encryption scheme in [37], which relies on indistinguishability obfuscation, can also be specialized to the ABE setting.

Some relationships between ABE and other cryptographic primitives have also been explored. The work of [2] derives schemes for verifiable computation from attribute-based encryption schemes, while [15] use attribute-based encryption as a tool in designing more general functional encryption and reusable garbling schemes. Dual system encryption proof techniques have also been further studied in the works of [21, 10, 37, 1], applied to achieve leakage resilience in [24, 20, 12], and applied directly to computational assumptions in [9].

1.2 Organization

In Section 2, we give the relevant background on KP-ABE systems and composite order bilinear groups, as well as formal statements of the complexity assumptions we rely on in the composite order setting. In Section 3, we present our KP-ABE system in composite order bilinear groups. In Section 4, we prove its full security. Section 5 gives the proof of our Bilinear Entropy Expansion lemma used in the proof of security for both the composite order and prime order variants of our construction. In Section 6, we give relevant background for prime order bilinear groups, state our complexity assumptions in this context, and present the prime order variant of our KP-ABE construction. We prove the full security of the prime order variant in Appendix A.

2 Preliminaries

2.1 Composite Order Bilinear Groups

We will first construct our system in composite order bilinear groups, which were introduced in [6]. We let $\mathcal{G}$ denote a group generator - an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group G. We define $\mathcal{G}$'s output as (N, G, G_T, e), where $N = pqw$ is a product of three distinct primes, G and G_T are cyclic groups of order N, and $e : G \times G \to G_T$ is a map with the following properties:

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

We refer to G as the *source group* and G_T as the *target group*. We assume that the group operations in G and G_T and the map e are computable in polynomial time with respect to λ, and the group descriptions of G and G_T include a generator of each group. We let G_p, G_q, and G_w denote the subgroups of order p, q, and w in G respectively. We note that these subgroups

are "orthogonal" to each other under the bilinear map e: if u, v are elements of two different subgroups out of $\{G_p, G_q, G_w\}$, then $e(u, v)$ is the identity element in G_T. If g_p generates G_p, g_q generates G_q, and g_w generates G_w, then every element f of G can be expressed as $f = g_p^{c_1} g_q^{c_2} g_w^{c_3}$ for some values $c_1, c_2, c_3 \in \mathbb{Z}_N$. We will refer to $g_p^{c_1}$ as the "G_p part of f", for example.

2.2 Complexity Assumptions

We now present the complexity assumptions we will use to prove the security of our system. We use the notation $x \leftarrow S$ to express that element x is chosen uniformly at random from the finite set S. We will first consider groups G whose orders are products of three distinct primes: p, q, w. We denote the subgroups of G with these orders by G_p, G_q, G_w respectively. We use the notation g_p, g_q, g_w to denote generators of these respective subgroups.

Subgroup Decision Assumption 1 The Subgroup Decision Problem 1 is stated as follows: on a group G of order pqw, given g_p, g_w and T where either $T = g_p^s \leftarrow G_p$ or $T = g_p^s g_q^{\tilde{s}} \leftarrow G_{pq}$ (where s is distributed uniformly in $\mathbb{Z}_p$ and $\tilde{s}$ is distributed uniformly in $\mathbb{Z}_q$), output "yes" if $T = g_p^s$, and "no" otherwise.

Definition 1. *Subgroup Decision Assumption 1 in G: no polynomial time algorithm can achieve non-negligible advantage in deciding the Subgroup Decision Problem 1 in G.*

Subgroup Decision Assumption 2 The Subgroup Decision Problem 2 is stated as follows: on a group G of order pqw, given $g_p, g_w, g_p^s g_q^{\tilde{s}}, g_q^{\tilde{r}} g_w^{r^*}$ and T where either $T = g_p^y g_w^{y^*}$ or $T = g_p^y g_q^{\tilde{y}} g_w^{y^*}$ (where s, y are distributed uniformly in $\mathbb{Z}_p$, $\tilde{s}, \tilde{r}$, and $\tilde{y}$ are distributed uniformly in $\mathbb{Z}_q$, and r^*, y^* are distributed uniformly in $\mathbb{Z}_w$), output "yes" if $T = g_p^y g_w^{y^*}$, and "no" otherwise.

Definition 2. *Subgroup Decision Assumption 2 in G: no polynomial time algorithm can achieve non-negligible advantage in deciding the Subgroup Decision Problem 2 in G.*

Subgroup Decision Assumption 3 The Subgroup Decision Problem 3 is stated as follows: given a group G of order pqw, given $g_p, g_q, g_w, g_p^{\alpha} g_q^{\tilde{\alpha}}, g_p^s g_q^{\tilde{s}}$, and T where either $T = e(g_p, g_p)^{\alpha s}$ or $T = X$ (where α, s are distributed uniformly in $\mathbb{Z}_p$, $\tilde{\alpha}, \tilde{s}$ are distributed uniformly in $\mathbb{Z}_q$, and X is distributed uniformly in G_T), output "yes" if $T = e(g_p, g_p)^{\alpha s}$, and "no" otherwise.

Definition 3. *Subgroup Decision Assumption 3 in G: no polynomial time algorithm can achieve non-negligible advantage in deciding the Subgroup Decision Problem 3 problem in G.*

We additionally use one assumption in the prime order setting, where G is a bilinear group of prime order q and g is a generator:

2-Linear Assumption (DLIN) The 2-Linear problem is stated as follows: given a cyclic group G of prime order p, $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}, g^{c_1 + c_2 + r} \in G$ (where y_1, y_2, c_1, c_2 are distributed uniformly in $\mathbb{Z}_p$ and r is either a uniform random element of $\mathbb{Z}_p$ or 0), output "yes" if r is a random element of $\mathbb{Z}_p$ and "no" otherwise.

Definition 4. *2-Linear Assumption in G: no polynomial time algorithm can achieve non-negligible advantage in deciding the 2-Linear problem in G.*

2.3 Background for ABE

We now give required background material on Linear Secret Sharing Schemes, the formal definition of a KP-ABE scheme, and the security definition we will use.

2.3.1 Linear Secret Sharing Schemes

Our construction uses linear secret-sharing schemes (LSSS). We use the following definition (adapted from [3]). In the context of ABE, attributes will play the role of parties and will be represented as nonempty subsets $K \subseteq [k]$ for a fixed k.

Definition 5. *(Linear Secret-Sharing Schemes (LSSS)) A secret sharing scheme* Π *over a set of attributes is called* linear *(over* $\mathbb{Z}_p$*) if*

1. *The shares belonging to all attributes form a vector over* $\mathbb{Z}_p$*.*
2. *There exists an* $\ell \times n$ *matrix* Λ *called the share-generating matrix for* Π*. The matrix* Λ *has* ℓ *rows and* n *columns. For all* $j = 1, \ldots, \ell$*, the* j^{th} *row of* Λ *is labeled by an attribute* K*. When we consider the column vector* $v = (s, r_2, \ldots, r_n)$*, where* $s \in \mathbb{Z}_p$ *is the secret to be shared and* $r_2, \ldots, r_n \in \mathbb{Z}_p$ *are randomly chosen, then* Λv *is the vector of* ℓ *shares of the secret* s *according to* Π*. The share* $(\Lambda v)_j = \lambda_K$ *belongs to attribute* K*.*

We note the *linear reconstruction* property: we suppose that Π is an LSSS. We let S denote an authorized set. Then there is a subset $S^* \subseteq S$ such that the vector $(1, 0, \ldots, 0)$ is in the span of rows of Λ indexed by S^*, and there exist constants $\{\omega_K \in \mathbb{Z}_p\}_{K \in S^*}$ such that, for any valid shares $\{\lambda_K\}$ of a secret s according to Π, we have: $\sum_{K \in S^*} \omega_K \lambda_K = s$. These constants $\{\omega_K\}$ can be found in time polynomial in the size of the share-generating matrix Λ [3]. For unauthorized sets, no such S^*, $\{\omega_K\}$ exist.

For our composite order group construction, we will employ LSSS matrices over $\mathbb{Z}_N$, where N is a product of three distinct primes p, q, w. As in the definition above over the prime order $\mathbb{Z}_p$, we say a set of attributes S is authorized if is a subset $S^* \subseteq S$ such that the rows of the access matrix A labeled by elements of S have the vector $(1, 0, \ldots, 0)$ in their span modulo N. However, in our security proof for our composite order system, we will further assume that for an unauthorized set, the corresponding rows of A do not include the vector $(1, 0, \ldots, 0)$ in their span modulo q. We may assume this because if an adversary can produce an access matrix A over $\mathbb{Z}_N$ and an unauthorized set over $\mathbb{Z}_N$ that is authorized over $\mathbb{Z}_q$, then this can be used to produce a non-trivial factor of the group order N, which would violate our subgroup decision assumptions.

2.3.2 KP-ABE Definition

A key-policy attribute-based encryption system consists of four algorithms: Setup, Encrypt, KeyGen, and Decrypt.

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

Encrypt$(\text{PP}, M, S) \rightarrow \text{CT}$ The encryption algorithm takes in the public parameters PP, the message M, and a set of attributes S. It will output a ciphertext CT. We assume that S is implicitly included in CT.

KeyGen$(\text{MSK}, \text{PP}, \mathbb{A}) \rightarrow \text{SK}$ The key generation algorithm takes in the master secret key MSK, the public parameters PP, and an access structure $\mathbb{A}$ over the universe of attributes. It outputs a private key SK which can be used to decrypt ciphertexts encrypted under a set of attributes which satisfies $\mathbb{A}$. We assume that $\mathbb{A}$ is implicitly included in SK.

Decrypt$(\mathrm{PP}, \mathrm{CT}, \mathrm{SK}) \to M$ The decryption algorithm takes in the public parameters PP, a ciphertext CT encrypted under a set of attributes S, and a private key SK for an access structure $\mathbb{A}$. If the set of attributes of the ciphertext satisfies the access structure of the private key, it outputs the message M.

2.3.3 Full Security for KP-ABE Systems

We define full security for KP-ABE Systems in terms of the following game:

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

Phase 1 The attacker queries the challenger for private keys corresponding to access structures.

Challenge The attacker declares two equal length messages M_0, M_1 and a set of attributes $A \subseteq \mathcal{U}$ where $\mathcal{U}$ is the attribute universe such that A does not satisfy the access structure of any of the keys requested in Phase 1. The challenger flips a random coin $\beta \in \{0, 1\}$, encrypts M_β under S to yield ciphertext CT_β and gives CT_β to the attacker.

Phase 2 The attacker queries the challenger for private keys corresponding to access structures that are not satisfied by S.

Guess The attacker outputs a guess β'.

Definition 6. *The advantage of an attacker $\mathcal{A}$ in this game is defined as* $Adv_{\mathcal{A}}^{KP-ABE}(\lambda) = \Pr[\beta = \beta'] - \frac{1}{2}$.

Definition 7. *A key-policy attribute based encryption scheme is fully secure if no polynomial time algorithm can achieve a non-negligible advantage in the above security game.*

2.3.4 Transformation from One-Use to Multiple Use KP-ABE

Given a KP-ABE scheme which is fully-secure when attributes are used at most once in access policies, we can obtain a KP-ABE scheme which is fully-secure when each attribute is used at most some constant number of times in access policies using a standard transformation. Essentially, multiple uses of an attribute are treated as new "attributes" in the one-use system. For example, if we want an attribute x to be able to be used up to k_x times in access policies, we will instantiate our one-use system with k_x "attributes" $x : 1, ..., x : k_x$. Each time we want to label a row of an access matrix Λ with x, we label it with $x : i$ for a new value of i. Each time we want to associate a subset S of attributes to a ciphertext, we instead use the set $S' = \{x : 1, ..., x : k_x \mid x \in S\}$. We can then employ the one-use KP-ABE scheme on this new larger set of "attributes" and retain its full security and functionality.

Clearly, this transformation comes at a cost. Typically, the ciphertext and public parameter size of the KP-ABE scheme resulting from the transformation now scale linearly with the number of attribute-*uses* allowed in access policies, not just the number of attributes. This blowup is seen in all previous fully secure KP-ABE schemes based on static assumptions and presents a problem if one desires policies which have high reuse of attributes. Our one-use KP-ABE scheme mitigates the problem with public parameter size by featuring public parameters that scale only logarithmically with the number of attributes supported by the system, compared to the linear scaling of all other known fully secure KP-ABE schemes based on static assumptions.

3 KP-ABE Construction

Our single-use KP-ABE construction assumes a polynomially sized attribute universe $\mathcal{U}$ where attributes are non-empty subsets $K \subseteq [k]$ for some fixed k. The prior fully secure single-use KP-ABE scheme in [19] required a fresh group element to appear in the public parameters for each attribute in the universe. After using the generic transformation discussed in section 2.3.4, this results in the scheme requiring a fresh group element for each attribute-*use* allowed in access policies. As a concrete example, if one wanted to allow 9 attributes to be used up to 7 times each, one needed to have $9 \times 7 = 63$ group elements in the public parameters corresponding to this attribute. In our composite order scheme, to allow the same $63 = 2^6 - 1$ attribute-uses, we only need 2×6 group elements in the public parameters corresponding to the attribute. The way we accomplish this dramatic "compression" of public parameters is to note that the encryptor can produce 63 group elements from 6 by taking products of all non-empty subsets (these correspond to subset-sums in the exponent). More generally, given k group elements $g^{a_1}, \ldots, g^{a_k}$, we can produce $2^k - 1$ group elements by enumerating over all non-empty subsets $K \subseteq [k]$ and computing $g^{\sum_{j \in K} a_j}$. We name the resulting collection of elements g^{A_K}, where $A_K := \sum_{j \in K} a_j$. Our composite order scheme uses two parallel such subset constructions (which is the reason for the factor of 2).

Obviously, these $2^k - 1$ group elements no longer look random - they have a linear relationships in their exponents by construction. However, since we are assuming the decisional linear assumption is hard, if we choose $2^k - 1$ additional random exponents $\{t_K\}$, then the $2(2^k - 1)$ group elements formed as $\{g^{t_K}, g^{t_K A_K}\}$ are computationally indistinguishable from $2(2^k - 1)$ uniformly random group elements (which lack any hidden linear structures in their exponents). The proof of this is the core of bilinear entropy expansion lemma, though the full statement of the lemma includes some additional structure that is useful for linking into a KP-ABE construction. The dual system encryption framework allows us to apply this argument to the parameters in the semi-functional space, where we do not need to publish the values $\{g^{a_j}\}$. (Note that publishing these would make the structure of $\{g^{t_K}, g^{t_K A_K}\}$ detectable through applications of the bilinear map.)

Setup$(\lambda, \mathcal{U}, k) \rightarrow PP, MSK$ The setup algorithm chooses a bilinear group G of order $N = pqw$ where p, q, w are primes. We let G_p, G_q, G_w represent the subgroup of order p, q, and w respectively in G. It then draws $\alpha \leftarrow \mathbb{Z}_N$ and random group element (generator) $g_p \in G_p$. For each $j \in [k]$, it chooses values $a_j, b_j \leftarrow \mathbb{Z}_N$.
The public parameters are $N, g_p, e(g_p, g_p)^\alpha, \{g_p^{a_j}, g_p^{b_j} : j \in [k]\}$. The MSK is α and a generator g_w of G_w. Such a construction is equipped to create keys for access policies which include attributes $K \subseteq [k]$ where K is not empty.

KeyGen$(MSK, \Lambda, PP) \rightarrow SK$ The key generation algorithm takes in the public parameters, master secret key, and LSSS access matrix Λ. First, the key generation algorithm generates $\{\lambda_K\}$: a linear sharing of α according to policy matrix Λ (the reader is referred to section 2.3.1 for details). For each attribute K corresponding to a row in the policy matrix Λ, it then raises generator g_w to random exponents to create $g_w^{z_K}, g_w^{z'_K}, g_w^{z''_K} \in G_w$, chooses exponent $y_K \leftarrow \mathbb{Z}_N$ and computes $g_p^{A_K} = \prod_{j \in K} g_p^{a_j}$ and $g_p^{B_K} = \prod_{j \in K} g_p^{b_j}$. Note that here and throughout the rest of the

description of our construction and its proof of security we will use the notation $A_K = \sum_{j \in K} a_j$ and $B_K = \sum_{j \in K} b_j$. It then outputs the secret key:

$$SK_\Lambda = \{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

(This implicitly includes Λ)

Encrypt$(M, S, PP) \to CT$ The encryption algorithm first draws $s \leftarrow \mathbb{Z}_N$. For each $K \in S$, the encryption algorithm draws $t_K \leftarrow \mathbb{Z}_N$ and computes $g_p^{A_K} = \prod_{j \in K} g_p^{a_j}$ and $g_p^{B_K} = \prod_{j \in K} g_p^{b_j}$. It then outputs the ciphertext:

$$CT = Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s, \quad g_p^{sA_K} g_p^{t_K B_K}, \quad g_p^{t_K} : (\forall K \in S)\}$$

(This implicitly includes S)

Decrypt$(CT, SK, PP) \to M$ We let S correspond to the set of attributes associated to ciphertext CT, and Λ be the policy matrix. If S satisfies Λ, the decryption algorithm computes suitable constants ω_K such that $\sum_{K \in S^*} \omega_K \lambda_K = \alpha$ (recall section 2.3.1). It then computes:

$$\begin{aligned}
&\prod_{K \in S^*} \left(e(g_p^s, \ g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}) \left(\frac{e(g_p^{y_K} g_w^{z'_K}, \ g_p^{sA_K} g_p^{t_K B_K})}{e(g_p^{t_K}, \ g_p^{y_K B_K} g_w^{z''_K})} \right)^{-1} \right)^{\omega_K} \\
&= \prod_{K \in S^*} \left(e(g_p, g_p)^{s\lambda_K} e(g_p, g_p)^{s y_K A_K} \left(\frac{e(g_p, g_p)^{s y_K A_K} e(g_p, g_p)^{t_K y_K B_K}}{e(g_p, g_p)^{t_K y_K B_K}} \right)^{-1} \right)^{\omega_K} \\
&= \prod_{K \in S^*} \left(\frac{e(g_p, g_p)^{s\lambda_K} e(g_p, g_p)^{s y_K A_K}}{e(g_p, g_p)^{s y_K A_K}} \right)^{\omega_K} \\
&= \prod_{K \in S^*} \left(e(g_p, g_p)^{s\lambda_K} \right)^{\omega_K} \\
&= e(g_p, g_p)^{\sum_{K \in S^*} s\omega_K \lambda_K} \\
&= e(g_p, g_p)^{\alpha s}
\end{aligned}$$

The message can then be recovered by computing: $Me(g_p, g_p)^{\alpha s} / e(g_p, g_p)^{\alpha s} = M$. This demonstrates correctness of the scheme.

4 Security Proof

Our security proof uses a hybrid argument over a sequence of games. We let Game_{real} denote the real security game. The rest of the games use semi-functional keys and ciphertexts, which we describe below. We let g_q denote a fixed generator of the subgroup G_q, which will serve as the "semi-functional space."

Like a typical dual system encryption proof, we will begin by transitioning from a normal ciphertext to a semi-functional ciphertext with semi-functional components that mimic the

structure of their normal counterparts. This kind of transition can be done with a basic subgroup decision assumption. We will then perform a hybrid over keys, gradually changing each one to a semi-functional form that does not properly decrypt the semi-functional ciphertext. To start, we can bring in semi-functional components for a particular key that mimic the structure of normal components, up to the constraint that the shared valued in the semi-functional space will be 0 (modulo q). Technically, this constraint arises because we will be taking a challenge term from a subgroup decision assumption that has an unknown exponent in the normal space and raising it to a share - so we have to make this a share of 0 and separately share the α value in the normal space so that the unknown exponent does not affect the correctness of the sharing in the normal space. At a higher level, this constraint explains why the simulator at this stage of the hybrid cannot solve the challenge problem for itself by test decrypting against a semi-functional ciphertext. Since the structure in the semi-functional space parallels the normal structure and the shared value here is zero, the semi-functional components will cancel out upon decryption.

So we can arrive at a stage where a key and ciphertext have semi-functional components structured just like the normal space, but with fresh parameters modulo q that are independent of the published parameters modulo p. This is a consequence of the Chinese Remainder Theorem, that ensures when we sample an exponent uniformly at random modulo N, its modulo p and modulo q reductions are independent and uniformly random in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively. Since these implicit parameters in the semi-functional space are never published, we can use our bilinear entropy expansion lemma to argue that their subset-sum structure is hidden under the decisional linear assumption. This allows us to replace them with higher entropy parameters (lacking the subset-sum structure of the normal space), and then argue that the shared value in the semi-functional space is information-theoretically hidden (this is where we use that the access policy is not satisfied and that attributes are used at most once in the policy). This enables us to switch the semi-functional shares in the key to shares of a random value, now destroying correct decryption of a semi-functional ciphertext. We then remove some of the other (now unnecessary) semi-functional components of the key, to reclaim the entropy of those parameters to use in processing the next key in the hybrid. Finally, once we have reached a game where all keys are semi-functional with shares of a random secret modulo q, we can use our our Subgroup Decision 3 to create such keys without knowing the master secret and can hence complete the proof.

We now formally present our definitions of semi-functional ciphertexts and keys and our hybrid proof:

Semi-functional Ciphertext We will use 3 types of semi-functional ciphertexts. To produce a semi-functional ciphertext for an attribute set S, one first calls the normal encryption algorithm to produce a normal ciphertext consisting of:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s, \quad g_p^{sA_K} g_p^{t_K B_K}, \quad g_p^{t_K} : (\forall K \in S)\}$$

One then draws $\tilde{s} \leftarrow \mathbb{Z}_N$. For each $K \in S$, an exponent $\tilde{t}_K \leftarrow \mathbb{Z}_N$ is chosen. The remaining composition of the semifunctional ciphertext depends on the type of ciphertext desired:

Type 1 The semi-functional ciphertext of Type 1 is formed as:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s g_q^{\tilde{s}}, \quad g_p^{sA_K} g_p^{t_K B_K} g_q^{\tilde{s}A_K} g_q^{\tilde{t}_K B_K}, \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

(again, here $A_K = \sum_{j \in K} a_j$ and $B_K = \sum_{j \in K} b_j$)

Type 2 The semi-functional ciphertext of Type 2 is formed as:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s g_q^{\tilde{s}}, \quad g_p^{sA_K} g_p^{t_K B_K} g_q^{\tilde{s}A_K} g_q^{\tilde{t}_K \tilde{b}_K}, \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

for fixed $\tilde{b}_K \in \mathbb{Z}_N$ which are chosen uniformly at random and fixed if they do not already exist (in a semi-functional key, for instance).

Type 3 The semi-functional ciphertext of Type 3 is formed as:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s g_q^{\tilde{s}}, \quad g_p^{sA_K} g_p^{t_K B_K} g_q^{\tilde{s}\tilde{a}_K} g_q^{\tilde{t}_K \tilde{b}_K}, \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

for fixed $\tilde{a}_K, \tilde{b}_K \in \mathbb{Z}_N$ which are chosen uniformly at random and fixed if they do not already exist.

Semi-functional Keys We will use 7 types of semi-functional keys. To produce a semi-functional key for an access policy Λ, one first calls the normal key generation algorithm to produce a normal key consisting of:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

The first 6 types of keys fall under 3 classes which have two variants each: a "Z" variant and an "R" variant. For Z-type keys one computes a linear sharing *of 0* under access policy Λ, creating shares $\tilde{\lambda}_K$. For R-type keys one computes a linear sharing of *a random element* u of $\mathbb{Z}_q$ which is fixed once it is created and used for all R-type keys. u is shared under access policy Λ, to create shares $\tilde{\lambda}_K$. The next steps depend on the class of the key:

Class 1 First compute $g_q^{A_K}$ and $g_q^{B_K}$ (where, again, A_K and B_K represent the subset-sums of a_j and b_j). For each K label in the honest key, one then draws $\tilde{y}_K \leftarrow \mathbb{Z}_N$ and forms the semi-functional key of type 1Z or 1R (depending on the sharing $\tilde{\lambda}_K$) as:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} g_q^{\tilde{y}_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_q^{\tilde{y}_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_q^{\tilde{y}_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Class 2 First compute $g_q^{A_K}$. *Random values* $\tilde{b}_K \in \mathbb{Z}_N$ are chosen if they do not already exist (in a semi-functional ciphertext, for instance) and fixed. For each K label in the honest key, one then draws $\tilde{y}_K \leftarrow \mathbb{Z}_N$ and forms the semi-functional key of type 2Z or 2R as:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} g_q^{\tilde{y}_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_q^{\tilde{y}_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_q^{\tilde{y}_K \tilde{b}_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Class 3 *Random values* $\tilde{a}_K, \tilde{b}_K \in \mathbb{Z}_N$ are chosen if they do not already exist and fixed. For each K label in the honest key, one then draws $\tilde{y}_K \leftarrow \mathbb{Z}_N$ and forms the semi-functional key of type 3Z or 3R as:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} g_q^{\tilde{y}_K \tilde{a}_K} g_w^{z_K}, \quad g_p^{y_K} g_q^{\tilde{y}_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_q^{\tilde{y}_K \tilde{b}_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Note that we now have defined 6 types of keys: 1Z, 1R, 2Z, 2R, 3Z, and 3R, where the letter (Z/R) describes whether the $\tilde{\lambda}_K$ share zero or a random element of $\mathbb{Z}_q$ respectively, and the number (1/2/3) describes whether the semi-functional analogues of the $g_p^{A_K}$ and $g_p^{B_K}$ in the G_q group are structured as subset-sums or as random elements of G_q (Class 1 keys have both $g_q^{A_K}$ and $g_q^{B_K}$. Class 2 keys have just $g_q^{A_K}$ structured, with a random element $g_q^{\tilde{b}_K}$. Class 3 keys have both replaced by random elements $g_q^{\tilde{a}_K}, g_q^{\tilde{b}_K}$ of G_q). There is one final type of key, type 4R, which does not contain any of these elements:

Type 4R Using shares $\tilde{\lambda}_K$ of u (which is randomly chosen from $\mathbb{Z}_p$ and fixed if it has not already been fixed), one forms the semi-functional key of type 4R as:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Proof Structure Our hybrid proof takes place over a series of games defined as follows: Letting Q denote the total number of key queries that the attacker makes, we define Game_{ℓ_1}, Game_{ℓ_2}, Game_{ℓ_3}, Game_{ℓ_4}, Game_{ℓ_5}, Game_{ℓ_6}, and Game_{ℓ_7} for $\ell = 1, ..., Q$. In each game, the first $\ell - 1$ keys are semi-functional of type 4R, and all keys after the ℓth request are normal. They differ in the construction of the ℓth key and the ciphertext as follows:

Game$_{\ell_1}$ In this game, the ℓth key is type 1Z and the ciphertext is type 1.

Game$_{\ell_2}$ In this game, the ℓth key is type 2Z and the ciphertext is type 2.

Game$_{\ell_3}$ In this game, the ℓth key is type 3Z and the ciphertext is type 3.

Game$_{\ell_4}$ In this game, the ℓth key is type 3R and the ciphertext is type 3.

Game$_{\ell_5}$ In this game, the ℓth key is type 2R and the ciphertext is type 2.

Game$_{\ell_6}$ In this game, the ℓth key is type 1R and the ciphertext is type 1.

Game$_{\ell_7}$ In this game, the ℓth key is type 4R and the ciphertext is type 1.

Note that under this definition, we have that in **Game**$_{0_7}$, the ciphertext given to the attacker is type 1 and the keys are all normal.

The outer structure of our hybrid argument will progress as follows. First, we transition from Game_{real} to Game_{0_7}, then to Game_{1_1}, next to Game_{1_2}, next to Game_{1_3}, next to Game_{1_4}, next to Game_{1_5}, next to Game_{1_6}, next to Game_{1_7} and then to Game_{2_1} and so on. We then arrive at Game_{Q_7}, where the ciphertext is semifunctional of type 1 and *all* of the keys given to the attacker are semi-functional of type 4R.

We then transition to one last game named Game_{final} which will complete our proof. Game_{final} uses a semi-functional ciphertext of a new type: type X, which we will now define:

Type X The semi-functional ciphertext of Type X is formed as:

$$MX, \quad \{g_p^s g_q^{\tilde{s}}, \quad g_p^{sA_K} g_p^{t_K B_K} g_q^{\tilde{s}A_K} g_q^{\tilde{t}_K B_K}, \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

where X is a uniform random element of G_T.(again, here $A_K = \sum_{j \in K} a_j$ and $B_K = \sum_{j \in K} b_j$)

Game$_{final}$ In this game, all keys are semi-functional of type 4R and the ciphertext is semi-functional of type X.

Note that a ciphertext of type X information-theoretically hides its message M because the message is multiplied by the uniform random X which is unused anywhere else. So, in Game_{final}, no polynomial time adversary will be able to achieve advantage in the security game, completing our proof.

Our hybrid argument is accomplished in the following lemmas:

Lemma 8. *Under the Subgroup Decision Assumption 1, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{real}$ and $Game_{0_7}$.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between Game_{real} and Game_{0_7}, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the Subgroup Decision Problem 1 as follows:

Given g_p, g_w and T where either $T = g_p^s \leftarrow G_p$ or $T = g_p^s g_q^{\tilde{s}} \leftarrow G_{pq}$ (where $s \leftarrow \mathbb{Z}_p$ and $\tilde{s} \leftarrow \mathbb{Z}_q$), consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g_p, choosing α, a_j, b_j randomly from $\mathbb{Z}_N$ and giving the constructed public parameters to $\mathcal{A}$. $\mathcal{B}$ can respond to key requests by running the usual key generation algorithm to make normal keys because it knows the MSK.

To return the challenge ciphertext for a set of attributes S, for each $K \in S$, a $t'_K \leftarrow \mathbb{Z}_N$ is drawn, then the following ciphertext is constructed and provided:

$$Me(g_p, T)^{\alpha}, \quad \left\{T, \quad T^{A_K} T^{t'_K B_K}, \quad T^{t'_K} : (\forall K \in S)\right\}$$

The simulator is able to honestly follow the rest of the scheme using generators g_p, g_w, so the only difference between its execution distributions depends on the different ciphertexts formed dependent on T.

Notice that if $T = g_p^s$ for $s \leftarrow \mathbb{Z}_p$, then the distribution of ciphertexts formed is identical to the honest case (where $t_K = t'_K s$), and so the simulator's behavior is exactly that of Game_{real}.

If $T = g_p^s g_q^{\tilde{s}}$ for $s \leftarrow \mathbb{Z}_p$, $\tilde{s} \leftarrow \mathbb{Z}_q$, the ciphertext formed is a semi-functional ciphertext of type 1 (where $t_K = t'_K s$ and $\tilde{t}_K = t'_K \tilde{s}$ are independent since the values of t'_K modulo p and modulo q are indpendent and uniform in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively by the Chinese Remainder Theorem), so the simulator's behavior is exactly that of Game_{0_7}.

Therefore, any adversary with non-negligible difference in advantage between Game_{real} and Game_{0_7} could be used to achieve the same non-negligible advantage in deciding the Subgroup Decision Problem 1. By assumption this is not possible, so such an adversary cannot exist.

□

Lemma 9. *Under the Subgroup Decision Assumption 2, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{(\ell-1)_7}$ and $Game_{\ell_1}$ for any ℓ from 1 to Q*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between $\text{Game}_{(\ell-1)_7}$ and Game_{ℓ_1} for some ℓ in $[1, Q]$, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the Subgroup Decision Problem 2 as follows:

Given $g_p, g_w, g_p^s g_q^{\tilde{s}}, g_q^{\tilde{r}} g_w^{r^*}$ and T where either $T = g_p^y g_w^{y^*}$ or $T = g_p^y g_q^{\tilde{y}} g_w^{y^*}$ (where $s, y \leftarrow \mathbb{Z}_p$, $\tilde{s}, \tilde{r}, \tilde{y} \leftarrow \mathbb{Z}_q$, and $r^*, y^* \leftarrow \mathbb{Z}_w$), consider the simulator in the security game which acts as follows:

The public parameters are formed by using the provided g_p, choosing α, a_j, b_j randomly from $\mathbb{Z}_n$ and giving the constructed public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S, for each $K \in S$, $t'_K \leftarrow \mathbb{Z}_N$ is drawn, then the following semi-functional ciphertext of type 1 is constructed and provided:

$$Me(g_p, g_p^s g_q^{\tilde{s}})^{\alpha}, \quad \{g_p^s g_q^{\tilde{s}}, \quad (g_p^s g_q^{\tilde{s}})^{A_K} (g_p^s g_q^{\tilde{s}})^{t'_K B_K}, \quad (g_p^s g_q^{\tilde{s}})^{t'_K} : (\forall K \in S)\}$$

Here the semi-functional ciphertext has $t_K = \tilde{s} t'_K$ and $\tilde{t}_K = \tilde{s} t'_K$ which are independent since the values of t'_K modulo p and modulo q are indpendent and uniform in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively by the Chinese Remainder Theorem.

For each requested key for policy Λ, since the simulator knows the MSK α, it can generate an honest key:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

using the key generation algorithm with g_p and g_w. For honest keys after the ℓth request, the simulator returns this key.

The simulator additionally chooses and fixes $u' \leftarrow \mathbb{Z}_N$. For key requests up to the $(\ell-1)$th request, the simulator generates shares $\tilde{\lambda}'_K$ of u'. The first $\ell - 1$ semi-functional keys of type 4R can be created by returning:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} (g_q^{\tilde{r}} g_w^{r^*})^{\tilde{\lambda}'_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Note that these are valid semi-functional keys of type 4R where the $\tilde{\lambda}_K = \tilde{r}\tilde{\lambda}'_K$ ($\tilde{\lambda}'_K$ are a sharing of u' so the $\tilde{r}\tilde{\lambda}'_K$ are a sharing of $u = \tilde{r}u'$, which is uniformly distributed in $\mathbb{Z}_q$ and fixed across all R-type keys). Also, notice that the extra G_w component $g_w^{r^*\tilde{\lambda}'_K}$ is absorbed by the uniform random $g_w^{z_K}$.

For the ℓth key request, the simulator generates λ'_K and λ''_K, shares of α and 0 respectively. It then draws $y'_K \leftarrow \mathbb{Z}_N$ and produces the key:

$$\{g_p^{\lambda'_K} T^{\lambda''_K} (T^{y'_K})^{A_K} g_w^{z_K}, \quad T^{y'_K} g_w^{z'_K}, \quad (T^{y'_K})^{B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

If $T = g_p^y g_w^{y^*}$, then the key is an honest key, where $\lambda_K = \lambda'_K + y\lambda''_K$, which is a valid sharing of α (λ''_K is a sharing of zero which multiplying it by y does not change, and adding this to λ'_K (the sharing of α) results in another sharing of α), and $y_K = y'_K y$. The extra G_w components are all absorbed by the uniform random $g_w^{z_K}, g_w^{z'_K}$, and $g_w^{z''_K}$.

If $T = g_p^y g_q^{\tilde{y}} g_w^{y^*}$, then the key is semi-functional of type 1Z, where $\lambda_K = \lambda'_K + y\lambda''_K$, which is again a valid sharing of α, $\tilde{y}_K = y'_K \tilde{y}$, and $y_K = y'_K y$ which are independent since the values of y'_K modulo p and modulo q are indpendent and uniform in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively by the Chinese Remainder Theorem. The extra G_w components are all absorbed by the uniform random $g_w^{z_K}, g_w^{z'_K}$, and $g_w^{z''_K}$.

Therefore, any adversary able to achieve a non-negligible difference in advantage between $\text{Game}_{(\ell-1)_7}$ and Game_{ℓ_1} for some ℓ in $[1, Q]$ could be used to achieve the same non-negligible advantage in deciding the Subgroup Decision Problem 2. By assumption this is not possible, so such an adversary cannot exist. □

Bilinear Entropy Expansion Before the next step of the hybrid proof, we will need a result about the indistinguishability between two distributions, based on the 2-Linear Computational Hardness Assumption.

Definition 10. *Given G, a group of prime order q, and g a generator of that group, let $\mathcal{D}_1(m)$ be the distribution of:*

$$\begin{aligned}
&g^{\tilde{s}}, \\
&g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}}, \\
&g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}}, \\
&g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, \\
&g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}}, \\
&g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}},
\end{aligned}$$

where the $\tilde{y}_i, \tilde{t}_i, b_i, r_i, \tilde{s} \leftarrow \mathbb{Z}_q$ *and* $M = 2^m$.

Definition 11. *Given* G*, a group of prime order* q*,* g *a generator of that group, and* $C = \{c_1, ..., c_m\}$ *a set of* m *elements drawn uniformly at random from* $\mathbb{Z}_q$*, let* $\mathcal{D}_2(m)$ *be the same distribution as above where the* $\tilde{y}_i, \tilde{t}_i, b_i, \tilde{s} \leftarrow \mathbb{Z}_q$ *but each* $r_i = \sum_{j \in C_i} c_j$ *where* C_i *denotes the* i*th indexed nonempty subset of* C *(*$|C| = m$ *and there are* $M - 1 = 2^m - 1$ *nonempty subsets).*

The Bilinear Entropy Expansion Lemma states that these two distributions are indistinguishable:

Lemma 12. *The distributions* $\mathcal{D}_1(k)$ *and* $\mathcal{D}_2(k)$ *are computationally indistinguishable under the 2-Linear computational hardness assumption if* $k = O(\lg poly(\lambda))$.

We defer the proof of this lemma to the end of this security proof, and return to the next step in our hybrid: where the B_K in the semi-functional space change to random elements $g_q^{\tilde{b}_K}$.

Lemma 13. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between* $Game_{\ell_1}$ *and* $Game_{\ell_2}$ *for any* ℓ *from* 1 *to* Q.

Proof. From Lemma 12 we have that the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ are computationally indistinguishable under the 2-Linear computational hardness assumption if $k = O(\lg poly(\lambda))$. However, if an algorithm $\mathcal{A}$ is able to achieve a non-negligible difference in advantage δ between Game_{ℓ_1} and Game_{ℓ_2} for some ℓ in $[1, Q]$, then we could use $\mathcal{A}$ to create a distinguisher $\mathcal{B}$ for $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ where k is the same value that defines our attribute universe $\mathcal{U}$. (This presents a contradiction since $|\mathcal{U}| = O(poly(\lambda))$ and $\mathcal{U}$ is defined as containing all subsets of $[k]$, so $k = O(\lg poly(\lambda))$). We do this as follows:

Let the generator of G_q used in $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ be g_q. Rename the set C as $C = \{b'_1, ..., b'_k\}$. Now, given the challenge instance

$$
\begin{aligned}
& g_q^{\tilde{s}}, \\
& g_q^{\tilde{y}_1}, ..., g_q^{\tilde{y}_{K-1}}, \\
& g_q^{\tilde{y}_1 r_1}, ..., g_q^{\tilde{y}_{K-1} r_{K-1}}, \\
& g_q^{\tilde{y}_1 b_1}, ..., g_q^{\tilde{y}_{K-1} b_{K-1}}, \\
& g_q^{\tilde{t}_1}, ..., g_q^{\tilde{t}_{K-1}}, \\
& g_q^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g_q^{\tilde{s} r_{K-1} + \tilde{t}_{K-1} b_{K-1}}
\end{aligned}
$$

relabel each index as the subset $K \subseteq [k]$ associated to it, so in particular we have:

$$g_q^{\tilde{y}_K}, g_q^{\tilde{y}_K r_K}$$

for all nonempty subsets $K \subseteq [k]$. (Note that here we do not even use all of the challenge instance, just the sections of $g_q^{\tilde{y}_K}, g_q^{\tilde{y}_K r_K}$. The full set is needed later when we reuse the lemma in the next hybrid proof).

Now, $\mathcal{B}$ forms the public parameters using g_p, choosing α, a_j, b_j randomly and giving the appropriately constructed public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S, first choose an $s \leftarrow \mathbb{Z}_N$. The simulator then redefines $\tilde{s}$ by sampling it uniformly at random from $\mathbb{Z}_N$. Then, $t_K, t_K^* \leftarrow \mathbb{Z}_N$

are chosen for each $K \in S$. The following semi-functional ciphertext can then be constructed and provided:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s g_q^{\tilde{s}}, \quad g_p^{sA_K} g_p^{t_K B_K} g_q^{\tilde{s}A_K} (g_q^{\tilde{y}_K r_K})^{t_K^*}, \quad g_p^{t_K} (g_q^{\tilde{y}_K})^{t_K^*} : (\forall K \in S)\}$$

Here, $\tilde{t}_K = t_K^* \tilde{y}_K$ for all $K \in S$.

For each requested key S for policy Λ, since the simulator knows the MSK α, it can generate an honest key:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

using the key generation algorithm with g_p and g_w. For honest keys after the ℓth request, the simulator returns this key.

The simulator additionally chooses and fixes $u \leftarrow \mathbb{Z}_N$. For key requests up to the $(\ell - 1)$th request, the simulator generates shares $\tilde{\lambda}_K$ of u.The first $\ell - 1$ semi-functional keys of type 4R can be created by returning:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

For the ℓth key request, $\mathcal{B}$ can compute $\tilde{\lambda}_K$: a sharing of 0 and return the key:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} (g_q^{\tilde{y}_K})^{A_K} g_w^{z_K}, \quad g_p^{y_K} (g_q^{\tilde{y}_K}) g_w^{z'_K}, \quad g_p^{y_K B_K} (g_q^{\tilde{y}_K r_K}) g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Notice that if $r_K = \sum_{j \in K} b'_j$, then $g_q^{r_K} = \prod_{j \in K} g_q^{b'_j}$ is distributed identically to $g_q^{B_K} = \prod_{j \in K} g_q^{b_j}$ by the Chinese Remainder Theorem. Each b_j is a random element of $\mathbb{Z}_N$ so its value taken mod q is independent of its value mod p. So, the distributions of $(\prod_{j \in K} g_p^{b_j}, \prod_{j \in K} g_q^{b_j}) = (g_p^{B_K}, g_q^{B_K})$ and $(\prod_{j \in K} g_p^{b_j}, \prod_{j \in K} g_q^{b'_j}) = (g_p^{B_K}, g_q^{r_K})$ are identical.

Constructing the ℓth key and challenge ciphertext this way gives them alternatively structured or random components depending on whether the $r_K = \sum_{j \in K} b'_j$ or are uniformly random in $\mathbb{Z}_q$. If the challenge set is drawn from $\mathcal{D}_2(k)$, then $\mathcal{B}$ behaves as expected in $\mathbf{Game}_{\ell_1}$ (where the challenge ciphertext and ℓth key are semi-functional of type 1 and 1Z respectively). If the challenge set is drawn from $\mathcal{D}_1(k)$, then $\mathcal{B}$ behaves as expected in $\mathbf{Game}_{\ell_2}$ (where the challenge ciphertext and ℓth key are semi-functional of type 2 and 2Z respectively).

Therefore, any adversary with non-negligible difference in advantage δ between Game_{ℓ_1} and Game_{ℓ_2} could be used to achieve the same non-negligible difference in advantage in distinguishing the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$. By Lemma 12 this is not possible, so such an adversary cannot exist and therefore we have proven that no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_1} and Game_{ℓ_2} for any ℓ from 1 to Q. □

The proof for the next step in our hybrid is similar, as we transition from structured $g_q^{A_K}$ to random elements $g_q^{\tilde{a}_K}$ (except this uses the full version of our Bilinear Entropy Expansion lemma):

Lemma 14. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_2} and Game_{ℓ_3} for any ℓ from 1 to Q.*

Proof. From Lemma 12 we have that the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ are computationally indistinguishable under the 2-Linear computational hardness assumption if $k = O(\lg poly(\lambda))$. However, if an algorithm $\mathcal{A}$ is able to achieve non-negligible difference in advantage δ between Game_{ℓ_2} and Game_{ℓ_3} for some ℓ in $[1, Q]$, then again we could use $\mathcal{A}$ to create a distinguisher $\mathcal{B}$ for $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ (where again we use the same k which defines our attribute universe $\mathcal{U}$ and satisfies $k = O(\lg poly(\lambda))$). We do this as follows:

Let the generator of G_q used in $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ be g_q. Relabel the set C as $C = \{a'_1, ..., a'_k\}$. Now, given the challenge instance

$$
\begin{aligned}
&g_q^{\tilde{s}},\\
&g_q^{\tilde{y}_1}, ..., g_q^{\tilde{y}_{K-1}},\\
&g_q^{\tilde{y}_1 r_1}, ..., g_q^{\tilde{y}_{K-1} r_{K-1}},\\
&g_q^{\tilde{y}_1 b_1}, ..., g_q^{\tilde{y}_{K-1} b_{K-1}},\\
&g_q^{\tilde{t}_1}, ..., g_q^{\tilde{t}_{K-1}},\\
&g_q^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g_q^{\tilde{s} r_{K-1} + \tilde{t}_{K-1} b_{K-1}}
\end{aligned}
$$

relabel each index as the subset $K \subseteq [k]$ associated to it, so we have:

$$g_q^{\tilde{s}}, g_q^{\tilde{y}_K}, g_q^{\tilde{y}_K r_K}, g_q^{\tilde{y}_K b_K}, g_q^{\tilde{t}_K}, g_q^{\tilde{s} r_K + \tilde{t}_K b_K}$$

for all nonempty subsets $K \subseteq [k]$.

Now, $\mathcal{B}$ forms the public parameters using g_p, choosing α, a_j, b_j randomly and giving the appropriately constructed public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S, first choose an $s \leftarrow \mathbb{Z}_N$. Then, $t_K \leftarrow \mathbb{Z}_N$ are chosen for all $K \in S$. The following semi-functional ciphertext can then be constructed and provided:

$$Me(g_p, g_p)^{\alpha s}, \quad \{g_p^s (g_q^{\tilde{s}}), \quad g_p^{s A_K} g_p^{t_K B_K} (g_q^{\tilde{s} r_K + \tilde{t}_K b_K}), \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

Note that here we implicitly set $\tilde{b}_K = b_K$.

For each requested key for policy Λ, since the simulator knows the MSK α, it can generate an honest key:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

using the key generation algorithm with g_p and g_w. For honest keys after the ℓth request, the simulator returns this key.

The simulator additionally chooses and fixes $u \leftarrow \mathbb{Z}_N$. For key requests up to the $(\ell - 1)$th request, the simulator generates shares $\tilde{\lambda}_K$ of u.The first $\ell - 1$ semi-functional keys of type 4R can be created by returning:

$$SK = \{g_p^{\lambda_K} A_K^{y_K} g_q^{\tilde{\lambda}_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

For the ℓth key request, $\mathcal{B}$ can compute $\tilde{\lambda}_K$: a sharing of 0, and return the key:

$$\{g_p^{\lambda_K} g_p^{y_K A_K} g_q^{\tilde{\lambda}_K} (g_q^{\tilde{y}_K r_K}) g_w^{z_K}, \quad g_p^{y_K} (g_q^{\tilde{y}_K}) g_w^{z'_K}, \quad g_p^{y_K B_K} (g_q^{\tilde{y}_K b_K}) g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Again, notice that if $r_K = \sum_{j \in K} a'_j$, then $g_q^{A_K} = \prod_{j \in K} g_q^{a_j}$ is distributed identically to $\prod_{j \in K} g_q^{a'_j} = g_q^{r_K}$ by the Chinese Remainder Theorem. Each a_j is a random element of $\mathbb{Z}_N$ so its value taken mod q is independent of its value mod p. So, the distributions of $(\prod_{j \in K} g_p^{a_j}, \prod_{j \in K} g_q^{a_j}) = (g_p^{A_K}, g_q^{A_K})$ and $(\prod_{j \in K} g_p^{a_j}, \prod_{j \in K} g_q^{a'_j}) = (g_p^{A_K}, g_q^{r_K})$ are identical.

So, constructing the challenge ciphertext and ℓth key this way causes them to have alternatively structured or random components depending on whether the $r_K = \sum_{j \in K} a'_j$ or are uniformly random in $\mathbb{Z}_q$. If the challenge set is drawn from $\mathcal{D}_2(k)$, then $\mathcal{B}$ behaves as expected in $\mathbf{Game}_{\ell_2}$ (where the challenge ciphertext and ℓth key are semi-functional of type 2 and 2Z respectively). If the challenge set is drawn from $\mathcal{D}_1(k)$, then $\mathcal{B}$ behaves as expected in $\mathbf{Game}_{\ell_3}$ (where the challenge ciphertext and ℓth key are semi-functional of type 3 and 3Z respectively).

Therefore, any adversary with non-negligible difference in advantage δ between Game_{ℓ_2} and Game_{ℓ_3}. could be used to achieve the same non-negligible difference in advantage in distinguishing the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$. By Lemma 12 this is not possible, so such an adversary cannot exist and therefore we have proven that no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_2} and Game_{ℓ_3} for any ℓ from 1 to Q. □

We continue to the next step in our hybrid proof:

Lemma 15. *No polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_3} and Game_{ℓ_4} for any ℓ from 1 to Q*

Proof. Recall that in both Game_{ℓ_3} and Game_{ℓ_4}, the ciphertext is semi-functional of type 3, all keys after the ℓth key request are normal, and the first $\ell - 1$ keys are semi-functional of type 4R. The only difference is the ℓth key (either semi-functional of type 3Z or 3R). In Game_{ℓ_3}, the shares $\tilde{\lambda}_K$ are a sharing of 0, while in Game_{ℓ_4}, they are a sharing of the u used in R-type keys. The transition between these two modes of operation is accomplished using an information-theoretic argument. Namely, the distribution of elements seen by an attacker in both games is the same.

To see this, note that for any attribute K not included in the ciphertext, then the distributions of the elements of the key indexed by K are identical in both cases, since the $\tilde{\lambda}_K$ are masked by a $g_q^{\tilde{a}_K \tilde{y}_K}$ where the $\tilde{a}_K$ are chosen uniformly at random and used nowhere else (they do not appear in keys other than the ℓth key and only appear at most once in the ℓth key because of our single-use restriction), therefore hiding the value of all $\tilde{\lambda}_K$ information-theoretically.

For attributes K used in the ciphertext, this argument does not apply (since these $\tilde{a}_K$ do appear elsewhere - in the ciphertext). For these, note that in the security game the attacker is not allowed to request a key that is able to decrypt the challenge ciphertext. So, for this key with policy Λ, the rows of the policy matrix Λ corresponding to the attributes the challenge ciphertext is encrypted under do not contain $(1, 0, ..., 0)$ in their span (modulo N). We can further assume that the corresponding rows of Λ do not include $(1, 0, ..., 0)$ in their span modulo q (because if an adversary can produce an unauthorized set over $\mathbb{Z}_N$ that is authorized over $\mathbb{Z}_q$, then this can be used to produce a non-trivial factor of the group order N, which would violate our subgroup decision assumptions). Therefore, there exists some vector $\vec{w} \in \mathbb{Z}_q^n$ orthogonal to the span of these rows which is not orthogonal to $(1, 0, ..., 0)$ modulo q (since $\mathbb{Z}_q$ is a finite field). By scaling this vector we can have $\vec{w} = (1, w_2, ..., w_n)$ for some collection of w_i. Now notice that whether the shares $\tilde{\lambda}_K$ which comprise $\vec{\tilde{\lambda}}$ were generated by taking $\Lambda \vec{r} = \vec{\tilde{\lambda}}$ where $\vec{r} = (0, r_2, ..., r_n)$ (that is, a valid sharing of zero) or taking $\Lambda(\vec{r} + u\vec{w})$ where (that is, a valid

sharing of the u used in R-type keys), then the distributions of shares $\tilde{\lambda}_K$ modulo q (they are exponents of g_q) that are not information-theoretically hidden by the previous argument are the same. All that is seen are $\tilde{\lambda}_K$ created by taking dot products with rows of Λ - for the shares that are not information-theoretically hidden, we have that the $u\vec{w}$ contributes 0 modulo q to the share so the distribution of these shares is the same as those produced by an honest sharing of zero.

Since the distributions of elements of the key given the ciphertext are identical whether the $\tilde{\lambda}_K$ are formed by taking a sharing of zero or a random element of $\mathbb{Z}_N$, then no algorithm can tell the difference between the two games, and so we have proven the lemma.

□

Now that we have changed the $\tilde{\lambda}_K$ from sharing 0 to sharing u, we proceed to restructure the A_K, B_K:

Lemma 16. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_4}$ and $Game_{\ell_5}$ for any ℓ from 1 to Q.*

Proof. Note that this proof is identical to Lemma 14, (which uses the Bilinear Entropy Expansion Lemma 12), with the only difference being that now the $\tilde{\lambda}_K$ are a sharing of a random element of $\mathbb{Z}_N$ instead of 0. So, the only change in the proof is that instead of generating $\tilde{\lambda}_K$ as a sharing of 0, the $\tilde{\lambda}_K$ are generated as a sharing of u: a fixed random element of $\mathbb{Z}_N$. The simulator is equally capable of making either set of shares, and using the sharing of a random element does not affect the simulator's ability to perform any of the other actions detailed in the remainder of Lemma 14. □

Lemma 17. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_5}$ and $Game_{\ell_6}$ for any ℓ from 1 to Q.*

Proof. Note that this proof is again identical to Lemma 13, with the only difference being that now the $\tilde{\lambda}_K$ are a sharing of a random element of $\mathbb{Z}_N$ instead of 0. So, again, the only change in the proof is that instead of generating $\tilde{\lambda}_K$ as a sharing of 0, the $\tilde{\lambda}_K$ are generated as a sharing of u: a fixed random element of $\mathbb{Z}_N$. The simulator is equally capable of making either set of shares, and using the sharing of a random element does not affect the simulator's ability to perform any of the other actions detailed in the remainder of Lemma 13. □

Lemma 18. *Under the Subgroup Decision Assumption 2, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_6}$ and $Game_{\ell_7}$ for any ℓ from 1 to Q*

Proof. If an algorithm $\mathcal{A}$ achieves non-negligible difference in advantage between Game_{ℓ_6} and Game_{ℓ_7} for some ℓ in $[1, Q]$, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the Subgroup Decision Problem 2 as follows:

Given $g_p, g_w, g_p^s g_q^{\tilde{s}}, g_q^{\tilde{r}} g_w^{r^*}$ and T where either $T = g_p^y g_w^{y^*}$ or $T = g_p^y g_q^{\tilde{y}} g_w^{y^*}$ (where $s, y \leftarrow \mathbb{Z}_p$, $\tilde{s}, \tilde{r}, \tilde{y} \leftarrow \mathbb{Z}_q$, and $r^*, y^* \leftarrow \mathbb{Z}_w$), consider the simulator in the security game which acts as follows:

The public parameters are formed by using the provided g_p, choosing α, a_j, b_j randomly and giving the constructed public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S, for each $K \in S$, a $t'_K \leftarrow \mathbb{Z}_N$ is chosen, then the following semi-functional ciphertext of type 1 is constructed and provided:

$$Me(g_p, g_p^s g_q^{\tilde{s}})^\alpha, \quad \{g_p^s g_q^{\tilde{s}}, \quad (g_p^s g_q^{\tilde{s}})^{A_K} (g_p^s g_q^{\tilde{s}})^{t'_K B_K}, \quad (g_p^s g_q^{\tilde{s}})^{t'_K} : (\forall K \in S)\}$$

Here the semi-functional ciphertext has $t_K = \tilde{s}t'_K$ and $\tilde{t}_K = \tilde{s}t'_K$ which are independent since the values of t'_K modulo p and modulo q are independent and uniform in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively by the Chinese Remainder Theorem.

For each requested key for policy Λ, since the simulator knows the MSK α, it can generate an honest key:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

using the key generation algorithm with g_p and g_w. For honest keys after the ℓth request, the simulator returns this key. The first $\ell - 1$ semi-functional keys of type 4R can be created by generating shares $\tilde{\lambda}'_K$ of a fixed $u' \leftarrow \mathbb{Z}_q$ and returning:

$$SK = \{g_p^{\lambda_K} g_p^{y_K A_K} (g_q^{\tilde{r}} g_w^{r^*})^{\tilde{\lambda}'_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Note that these are valid semi-functional keys of type 4R where the $\tilde{\lambda}_K = \tilde{r}\tilde{\lambda}'_K$ and the extra G_w component is absorbed by the uniform random $g_w^{z_K}$.

For the ℓth key request, the simulator generates λ_K and $\tilde{\lambda}'_K$, shares of α and u' respectively. It then draws y'_K uniformly at random from $\mathbb{Z}_N$ and produces the key:

$$\{g_p^{\lambda_K} (g_q^{\tilde{r}} g_w^{r^*})^{\tilde{\lambda}'_K} (T^{y'_K})^{A_K} g_w^{z_K}, \quad T^{y'_K} g_w^{z'_K}, \quad (T^{y'_K})^{B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Again, if $\tilde{\lambda}'_K$ are a sharing of u', then $\tilde{r}\tilde{\lambda}'_K$ is the sharing of $\tilde{r}u'$, which is fixed across all R-type keys.

If $T = g_p^y g_q^{\tilde{y}} g_w^{y^*}$, then the key is semi-functional of type 1R, where $\tilde{y}_K = y'_K \tilde{y}$, and $y_K = y'_K y$ which are independent since the values of y'_K modulo p and modulo q are independent and uniform in $\mathbb{Z}_p, \mathbb{Z}_q$ respectively by the Chinese Remainder Theorem. The extra G_w components are all absorbed by the uniform random $g_w^{z_K}, g_w^{z'_K}$, and $g_w^{z''_K}$.

If $T = g_p^y g_w^{y^*}$, then the key is a semi-functional key of type 4R, where $y_K = y'_K y$. The extra G_w components are all absorbed by the uniform random $g_w^{z_K}, g_w^{z'_K}$, and $g_w^{z''_K}$.

Therefore, any adversary able to achieve a non-negligible difference in advantage between Game_{ℓ_6} and Game_{ℓ_7} for some ℓ in $[1, Q]$ could be used to achieve the same non-negligible advantage in deciding the Subgroup Decision Problem 2. By assumption this is not possible, so such an adversary cannot exist.

□

This set of hybrids takes us to Game_{Q_7}: where the semi-functional ciphertext is of type 1 and all keys are semi-functional of type 4R. We take one more step to Game_{final} where the distribution of the ciphertext is independent of the message - a game in which the adversary cannot have any advantage:

Lemma 19. *Under the Subgroup Decision Assumption 3, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{Q_7} and Game_{final}.*

Proof. If an algorithm $\mathcal{A}$ is able to achieve a non-negligible difference in advantage between Game_{Q_7} and Game_{final}, then we could use $\mathcal{A}$ to break the Subgroup Decision Assumption as follows:

Given $g_p, g_q, g_w, g_p^{\alpha} g_q^{\tilde{\alpha}}, g_p^s g_q^{\tilde{s}}$, and T where either $T = e(g_p, g_p)^{\alpha s}$ or X (where $\alpha, s \leftarrow \mathbb{Z}_p$, $\tilde{\alpha}, \tilde{s} \leftarrow \mathbb{Z}_q$, and $X \leftarrow G_T$), consider the simulator in the security game which acts as follows:

The public parameters are formed by using the provided g_p, computing $e(g_p, g_p^{\alpha} g_q^{\tilde{\alpha}}) = e(g_p, g_p)^{\alpha}$, choosing a_j, b_j randomly and giving the constructed public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S, for each $K \in S$, $t_K, \tilde{t}_K \leftarrow \mathbb{Z}_N$ are chosen, then the following ciphertext is constructed and provided:

$$MT, \quad \{(g_p^s g_q^{\tilde{s}}), \quad (g_p^s g_q^{\tilde{s}})^{A_K} g_p^{t_K B_K} g_q^{\tilde{t}_K B_K}, \quad g_p^{t_K} g_q^{\tilde{t}_K} : (\forall K \in S)\}$$

For all key requests, note that the simulator no longer has the value α (only $g_p^\alpha g_q^{\tilde{\alpha}}$). However, note that it can still generate $g_p^{\lambda_K} g_q^{\tilde{\lambda}_K}$ where the λ_K are a sharing of α and $\tilde{\lambda}_K$ are a sharing of $\tilde{\alpha}$ by performing the same share-generation procedure starting with $g_p^\alpha g_q^{\tilde{\alpha}}$ in parallel for both subgroups, just in the exponent of $g_p g_q$ (each of which it has, separately), by choosing random $r_i, \tilde{r}_i$, computing $g_p^{r_i} g_q^{\tilde{r}_i}$, and raising to appropriate exponents from the policy matrix / computing products to get the desired dot product in the exponent. It then draws $y_K \leftarrow \mathbb{Z}_N$ and $g_w^{z_K}, g_w^{z'_K}, g_w^{z''_K}$ uniformly at random from G_w (by choosing exponents of g_w uniformly at random from $\mathbb{Z}_n$), finally producing the semi-functional key of type 4R (where $u = \tilde{\alpha}$):

$$\{g_p^{\lambda_K} g_q^{\tilde{\lambda}_K} g_p^{y_K A_K} g_w^{z_K}, \quad g_p^{y_K} g_w^{z'_K}, \quad g_p^{y_K B_K} g_w^{z''_K} : (\forall K \text{ labels} \in \Lambda)\}$$

Note that when $T = e(g_p, g_p)^{\alpha s}$, then the simulator is acting exactly as in Game_{Q_7} (where the ciphertext is distributed as a semi-functional ciphertext of type 1). When $T = X$ (a completely random element of G_T), then the ciphertext is distributed as a semi-functional ciphertext of type X (as in Game_{final}) and the message is information-theoretically masked.

So, any adversary able to achieve a non-negligible difference in advantage between Game_{Q_7} and Game_{final} could be used to achieve the same non-negligible advantage in deciding the Subgroup Decision Problem 3. By assumption this is not possible, so such an adversary cannot exist.

□

We have now proven the following theorem

Theorem 20. *Under the 2-Linear Computational Hardness Assumption, Subgroup Decision Assumption 1, Subgroup Decision Assumption 2, and Subgroup Decision Assumption 3, our single-use KP-ABE scheme is fully secure.*

Proof. If the 2-Linear Computational Hardness Assumption, Subgroup Decision Assumption 1, Subgroup Decision Assumption 2, and Subgroup Decision Assumption 3 hold, then by the previous lemmas, we have shown that the real security game is computationally indistinguishable from Game_{final}, in which the challenge ciphertext's message is information-theoretically hidden from the attacker. Hence, no attacker can achieve a non-negligible advantage in breaking the KP-ABE scheme. □

5 Bilinear Entropy Expansion Lemma

We return to the deferred distribution indistinguishability proof. Recall the definitions of the two distributions:

Definition 21. *Given G, a group of prime order q, and g a generator of that group, let $\mathcal{D}_1(m)$*

be the distribution of:

$$
\begin{aligned}
&g^{\tilde{s}},\\
&g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}},\\
&g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}},\\
&g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}},\\
&g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}},\\
&g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}},
\end{aligned}
$$

where the $\tilde{y}_i, \tilde{t}_i, b_i, r_i, \tilde{s} \leftarrow \mathbb{Z}_q$ *and* $M = 2^m$.

Definition 22. *Given* G*, a group of prime order* q*,* g *a generator of that group, and* $C = \{c_1, ..., c_m\}$ *a set of* m *elements drawn uniformly at random from* $\mathbb{Z}_q$*, let* $\mathcal{D}_2(m)$ *be the distribution of:*

$$
\begin{aligned}
&g^{\tilde{s}},\\
&g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}},\\
&g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}},\\
&g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}},\\
&g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}},\\
&g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}},
\end{aligned}
$$

where the $\tilde{y}_i, \tilde{t}_i, b_i, \tilde{s} \leftarrow \mathbb{Z}_q$ *and* $M = 2^m$ *but each* $r_i = \sum_{j \in C_i} c_j$ *where* C_i *denotes the* i*th indexed nonempty subset of* C *(*$|C| = m$ *and there are* $M - 1 = 2^m - 1$ *nonempty subsets).*

We show that the distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$ are computationally indistinguishable if $m = O(\lg poly(\lambda))$ through an inductive proof, beginning with the base case of $m = 2$, where a distinguisher for $\mathcal{D}_1(2)$ and $\mathcal{D}_2(2)$ ($C = \{c_1, c_2\}$) can be used to achieve the same advantage in the 2-Linear Problem.

Lemma 23. *If there exists a polynomial-time algorithm able to achieve advantage* $2^2\delta$ *in distinguishing between the distributions* $\mathcal{D}_1(2)$ *and* $\mathcal{D}_2(2)$*, then there exists a polynomial-time algorithm able to achieve advantage* δ *in the 2-Linear Problem.*

Proof. If there exists a polynomial time algorithm $\mathcal{A}$ which distinguishes between $\mathcal{D}_1(2)$ and $\mathcal{D}_2(2)$ with advantage $2^2\delta$, we can construct a distinguisher for the 2-Linear problem: $\mathcal{B}$. $\mathcal{B}$, upon receiving $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}, g^{c_1 + c_2 + r}$, draws uniform random $\tilde{s}, b_3, \tilde{y}_3, \tilde{t}_1, \tilde{t}_2, \tilde{t}_3, \gamma_1, \gamma_2 \leftarrow \mathbb{Z}_q$,

then creates the set:

$$
\begin{array}{l}
g^{\tilde{s}}, \\
g^{y_1}, \quad g^{y_2}, \quad g^{\tilde{y}_3}, \\
g^{y_1 c_1}, \quad g^{y_2 c_2}, \quad (g^{c_1+c_2+r})^{\tilde{y}_3}, \\
(g^{y_1 c_1})^{-\frac{\tilde{s}}{\tilde{t}_1}} (g^{y_1})^{\gamma_1}, \quad (g^{y_2 r_c})^{-\frac{\tilde{s}}{\tilde{t}_2}} (g^{y_2})^{\gamma_2}, \quad g^{\tilde{y}_3 b_3}, \\
g^{\tilde{t}_1}, \quad g^{\tilde{t}_2}, \quad g^{\tilde{t}_3}, \\
g^{\tilde{t}_1 \gamma_1}, \quad g^{\tilde{t}_2 \gamma_2}, \quad (g^{c_1+c_2+r})^{\tilde{s}} g^{\tilde{t}_3 b_3}
\end{array}
$$

then runs $\mathcal{A}$ on this input and returns the output of $\mathcal{A}$.

Notice that if $r = 0$, this distribution is exactly $\mathcal{D}_2(2)$ (with $C = \{c_1, c_2\}$, $\tilde{y}_1 = y_1, \tilde{y}_2 = y_2, b_1 = -\frac{c_1 s}{\tilde{t}_1} + \gamma_1$, and $b_2 = -\frac{c_2 s}{\tilde{t}_2} + \gamma_2$). If r is instead random, this distribution is exactly $\mathcal{D}_1(2)$. Therefore, $\mathcal{B}$ will achieve the same advantage $2^2\delta$ as $\mathcal{A}$ (which is greater than δ) in deciding the 2-Linear problem. □

Lemma 24. *For all integers $m \geq 2$, if there exists a polynomial-time algorithm able to achieve an advantage of $2^{m+1}\delta$ deciding between distributions $\mathcal{D}_1(m+1)$ and $\mathcal{D}_2(m+1)$, then either there exists a polynomial-time algorithm able to achieve an advantage of $2^m\delta$ in deciding between distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$ or there exists a polynomial time algorithm able to achieve an advantage of δ in the 2-Linear Problem.*

Proof. If there exists a polynomial time algorithm $\mathcal{A}$ which distinguishes between $\mathcal{D}_1(m+1)$ and $\mathcal{D}_2(m+1)$ with non-negligible advantage $2^{m+1}\delta$, we construct $\mathcal{B}$: a distinguisher for $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$.

$\mathcal{B}$, upon receiving

$$
\begin{array}{l}
g^{\tilde{s}}, \\
g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}}, \\
g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}}, \\
g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, \\
g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}}, \\
g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}},
\end{array}
$$

where $M = 2^m$, first draws $y_1^*, ..., y_{M-1}^*, \sigma_1, ..., \sigma_{M-1}, \gamma_1, ..., \gamma_{M-1}, \tilde{y}_M, \tilde{t}_M, b_M, c_{m+1} \leftarrow \mathbb{Z}_q$, and constructs the following set:

$g^{\tilde{s}}$,
$g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}}, g^{\tilde{y}_M}$,
$(g^{\tilde{y}_1})^{y_1^*}, ..., (g^{\tilde{y}_{M-1}})^{y_{M-1}^*}$,
$g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}}, g^{\tilde{y}_M c_{m+1}}$,
$(g^{\tilde{y}_1})^{y_1^* c_{m+1}} (g^{\tilde{y}_1 r_1})^{y_1^*}, ..., (g^{\tilde{y}_{M-1}})^{y_{M-1}^* c_{m+1}} (g^{\tilde{y}_{M-1} r_{M-1}})^{y_{M-1}^*}$,
$g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, g^{\tilde{y}_M b_M}$,
$(g^{\tilde{y}_1 b_1})^{y_1^*} (g^{\tilde{y}_1})^{\sigma_1 y_1^*}, ..., (g^{\tilde{y}_{M-1} b_{M-1}})^{y_{M-1}^*} (g^{\tilde{y}_{M-1}})^{\sigma_{M-1} y_{M-1}^*}$,
$g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}}, g^{\tilde{t}_M}$,
$g^{\tilde{t}_1} (g^{\tilde{y}_1})^{\gamma_1}, ..., g^{\tilde{t}_{M-1}} (g^{\tilde{y}_{M-1}})^{\gamma_{M-1}}$,
$g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}}, (g^{\tilde{s}})^{c_{m+1}} g^{\tilde{t}_M b_M}$,
$(g^{\tilde{s}})^{c_{m+1}} g^{\tilde{s} r_1 + \tilde{t}_1 b_1} (g^{\tilde{t}_1})^{\sigma_1} (g^{b_1 \tilde{y}_1})^{\gamma_1} (g^{\tilde{y}_1})^{\sigma_1 \gamma_1}, ..., (g^{\tilde{s}})^{c_{m+1}} g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}} (g^{\tilde{t}_{M-1}})^{\sigma_{M-1}} (g^{b_{M-1} \tilde{y}_{M-1}})^{\gamma_{M-1}} (g^{\tilde{y}_{M-1}})^{\sigma_{M-1} \gamma_{M-1}}$
$=$
$g^{\tilde{s}}$,
$g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}}, g^{\tilde{y}_M}$,
$g^{\tilde{y}_1 y_1^*}, ..., g^{\tilde{y}_{M-1} y_{M-1}^*}$,
$g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}}, g^{\tilde{y}_M c_{m+1}}$,
$g^{\tilde{y}_1 y_1^* (r_1 + c_{m+1})}, ..., g^{\tilde{y}_{M-1} y_{M-1}^* (r_{M-1} + c_{m+1})}$,
$g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, g^{\tilde{y}_M b_M}$,
$g^{\tilde{y}_1 y_1^* (b_1 + \sigma_1)}, ..., g^{\tilde{y}_{M-1} y_{M-1}^* (b_{M-1} + \sigma_{M-1})}$,
$g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}}, g^{\tilde{t}_M}$,
$g^{\tilde{t}_1 + \tilde{y}_1 \gamma_1}, ..., g^{\tilde{t}_{M-1} + \tilde{y}_{M-1} \gamma_{M-1}}$,
$g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}}, g^{\tilde{s} c_{m+1} + \tilde{t}_M b_M}$,
$g^{\tilde{s}(r_1 + c_{m+1}) + (\tilde{t}_1 + \tilde{y}_1 \gamma_1)(b_1 + \sigma_1)}, ..., g^{\tilde{s}(r_{M-1} + c_{m+1}) + (\tilde{t}_{M-1} + \tilde{y}_{M-1} \gamma_{M-1})(b_{M-1} + \sigma_{M-1})}$

Notice that if $\mathcal{B}$'s input is $\mathcal{D}_2(m)$, then the distribution of sets constructed by $\mathcal{B}$ is exactly $\mathcal{D}_2(m+1)$, where a new c_{m+1} element is drawn and added to form the subsets of the new augmented set C, $\tilde{y}_{M+i} = \tilde{y}_i y_i^*$, $b_{M+i} = b_i + \sigma_i$, and $\tilde{t}_{M+i} = \tilde{t}_i + \tilde{y}_i \gamma_i$ for $i = 1, ..., M-1$ which are all uniformly distributed at random. However, if $\mathcal{B}$'s input is $\mathcal{D}_1(m)$, then the distribution of sets constructed by $\mathcal{B}$ is not exactly $\mathcal{D}_1(m+1)$.

Definition 25. *Let $\mathcal{D}'_1(m+1)$ be the distribution of sets created by $\mathcal{B}$ given input sets from $\mathcal{D}_1(m)$.*

We have therefore only proved that if an algorithm is able to achieve advantage in distinguishing $\mathcal{D}'_1(m+1)$ and $\mathcal{D}_2(m+1)$, then it can be used to achieve that same advantage in deciding between $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$.

Fortunately, we can transition between $\mathcal{D}'_1(m+1)$ and $\mathcal{D}_1(m+1)$ using a hybrid proof:

First we define $M = 2^m$ hybrid distributions indexed by (j):

Definition 26. *Let* $\mathcal{D}_1'^{(j)}(m+1)$ *be the distribution of:*

$g^{\tilde{s}}$,
$g^{\tilde{y}_1}, ..., g^{\tilde{y}_{M-1}}, g^{\tilde{y}_M}$,
$g^{\tilde{y}_{M+1}}, ..., g^{\tilde{y}_{2M-1}}$,
$g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_{M-1} r_{M-1}}, g^{\tilde{y}_M c_{m+1}}$,
$g^{\tilde{y}_{M+1}(r_1+c_{m+1})}, ..., g^{\tilde{y}_{M+j}(r_j+c_{m+1})}, g^{\tilde{y}_{M+j+1} r_{M+j+1}}, ..., g^{\tilde{y}_{2M-1} r_{2M-1}}$,
$g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, g^{\tilde{y}_M b_M}$,
$g^{\tilde{y}_{M+1} b_{M+1}}, ..., g^{\tilde{y}_{2M-1} b_{2M-1}}$,
$g^{\tilde{t}_1}, ..., g^{\tilde{t}_{M-1}}, g^{\tilde{t}_M}$,
$g^{\tilde{t}_{M+1}}, ..., g^{\tilde{t}_{2M-1}}$,
$g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_{M-1} + \tilde{t}_{M-1} b_{M-1}}, g^{\tilde{s} c_{m+1} + \tilde{t}_M b_M}$,
$g^{\tilde{s}(r_1+c_{m+1}) + \tilde{t}_{M+1} b_{M+1}}, ..., g^{\tilde{s}(r_j+c_{m+1}) + \tilde{t}_{M+j} b_{M+j}}, g^{\tilde{s} r_{M+j+1} + \tilde{t}_{M+j+1} b_{M+j+1}}, ..., g^{\tilde{s} r_{2M-1} + \tilde{t}_{2M-1} b_{2M-1}}$

for $j = 0, ..., M-1$ *where the* r_{M+j+i} *are distributed uniformly at random in* $\mathbb{Z}_p$ *for* $i = 1, ...M-j-1$

Notice that $\mathcal{D}_1'^{(0)}(m+1) = \mathcal{D}_1(m+1)$ and $\mathcal{D}_1'^{(M-1)}(m+1) = \mathcal{D}_1'(m+1)$. So, if some adversary $\mathcal{A}$ could distinguish between $\mathcal{D}_1(m+1)$ and $\mathcal{D}_1'(m+1)$ with non-negligible advantage $2^m\delta$, then:

$$|\Pr[\mathcal{A} = 1|\mathcal{D}_1'^{(0)}(m+1)] - \Pr[\mathcal{A} = 1|\mathcal{D}_1'^{(M-1)}(m+1)]| = 2^m\delta \tag{1}$$

So, by the triangle inequality, there must exists some j such that:

$$\left|\Pr[\mathcal{A} = 1|\mathcal{D}_1'^{(j+1)}(m+1)] - \Pr[\mathcal{A} = 1|\mathcal{D}_1'^{(j)}(m+1)]\right| \geq \frac{2^m\delta}{M} = \delta \tag{2}$$

Such an $\mathcal{A}$ can be used to construct a distinguisher for the 2-Linear Problem: $\mathcal{B}$ that achieves advantage δ:

$\mathcal{B}$, upon receiving $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}, g^{c_1+c_2+r}$, relabels the elements as:

$$g, g^{y_1}, g^{y_2}, g^{y_1 r^*}, g^{y_2 c_{m+1}}, g^x$$

(defining $y_1 = y_1, y_2 = y_2, r^* = c_1, c_{m+1} = c_2$, and $x = c_1 + c_2 + r$) $\mathcal{B}$ then draws

$$\begin{array}{l} \tilde{s}, \gamma \\ \tilde{y}_1, ..., \tilde{y}_j, \tilde{y}_{j+2}, ..., \tilde{y}_M \\ y^*_{M+1}, ..., y^*_{M+j}, \\ \tilde{y}_{M+j+1}, ..., \tilde{y}_{2M-1}, \\ t_1, ..., t_{2M-1} \\ b_1, ..., b_j, b_{j+2}, ..., b_{2M-1} \\ r_1, ..., r_j, r_{j+2}, ..., r_{2M-1} \end{array}$$

uniformly at random from $\mathbb{Z}_q$ and constructs:

$g^{\tilde{s}}$,

$g^{\tilde{y}_1}, ..., g^{\tilde{y}_j}, g^{y_1}, g^{\tilde{y}_{j+2}}, ..., g^{\tilde{y}_M}$,

$(g^{y_2})^{y^*_{M+1}}, ..., (g^{y_2})^{y^*_{M+j}}, g^{\tilde{y}_{M+j+1}}, ..., g^{\tilde{y}_{2M-1}}$,

$g^{\tilde{y}_1 r_1}, ..., g^{\tilde{y}_j r_j}, g^{y_1 r^*}, g^{\tilde{y}_{j+2} r_{j+2}}, ..., g^{\tilde{y}_M r_M}$,

$((g^{y_2})^{y^*_{M+1}})^{r_1}(g^{y_2 c_{m+1}})^{y^*_{M+1}}, ..., ((g^{y_2})^{y^*_{M+j}})^{r_j}(g^{y_2 c_{m+1}})^{y^*_{M+j}}, (g^x)^{\tilde{y}_{M+j+1}}, g^{\tilde{y}_{M+j+2} r_{M+j+2}}, ..., g^{\tilde{y}_{2M-1} r_{2M-1}}$

$g^{\tilde{y}_1 b_1}, ..., g^{\tilde{y}_j b_j}, (g^{y_1 r^*})^{-\frac{\tilde{s}}{\tilde{t}_j}}(g^{y_1})^{\gamma}, g^{\tilde{y}_{j+2} b_{j+2}}, ..., g^{\tilde{y}_{M-1} b_{M-1}}, g^{\tilde{y}_M b_M}$,

$(g^{y_2})^{y^*_{M+1} b_{M+1}}, ..., (g^{y_2})^{y^*_{M+j} b_{M+j}}, g^{\tilde{y}_{M+j+1} b_{M+j+1}} ..., g^{\tilde{y}_{2M-1} b_{2M-1}}$,

$g^{\tilde{t}_1}, ..., g^{\tilde{t}_M}$,

$g^{\tilde{t}_{M+1}}, ..., g^{\tilde{t}_{2M-1}}$,

$g^{\tilde{s} r_1 + \tilde{t}_1 b_1}, ..., g^{\tilde{s} r_j + \tilde{t}_j b_j}, g^{\tilde{t}_{j+1}\gamma}, g^{\tilde{s} r_{j+1} + \tilde{t}_{j+1} b_{j+1}}, ..., g^{\tilde{s} r_M + \tilde{t}_M b_M}$,

$g^{\tilde{s}(r_1 + c_{m+1}) + \tilde{t}_{M+1} b_{M+1}}, ..., g^{\tilde{s}(r_j + c_{m+1}) + \tilde{t}_{M+j} b_{M+j}}, (g^x)^{\tilde{s}} g^{\tilde{t}_{M+j+1} b_{M+j+1}}, g^{\tilde{s} r_{M+j+2} + \tilde{t}_{M+j+2} b_{M+j+2}}, ..., g^{\tilde{s} r_{2M-1} + \tilde{t}_{2M-1} b_{2M-1}}$

where $b_{j+1} = -\frac{sr^*}{\tilde{t}_{j+1}} + \gamma$, $\tilde{y}_{j+1} = y_1$, $r_{j+1} = r^*$, and the $\tilde{y}_{M+i} = y_2 \tilde{y}^*_{M+i}$ for $i = 1, ..., j$ are distributed uniformly at random from $\mathbb{Z}_p$.

$\mathcal{B}$ then runs $\mathcal{A}$ on this input and outputs the same.

Note that if $x = r^* + c_{m+1} + 0$, then $\mathcal{B}$ has sampled an instance of $\mathcal{D}_1'^{(j+1)}(m+1)$. Otherwise, if $x = r^* + c_{m+1} + r$ for a uniform random r it has sampled an instance of $\mathcal{D}_1'^{(j)}(m+1)$. So, $\mathcal{B}$ will enjoy the same advantage δ of $\mathcal{A}$ but in deciding the 2-Linear Problem.

We assumed there is a polynomial time algorithm $\mathcal{A}$ which distinguishes between $\mathcal{D}_1(m+1)$ and $\mathcal{D}_2(m+1)$ with advantage $2^{m+1}\delta$. By the triangle inequality, then $\mathcal{A}$ must be able to be used to either achieve advantage $2^m\delta$ in distinguishing between instances of $\mathcal{D}_1(m+1)$ and $\mathcal{D}_1'(m+1)$ or achieve advantage $2^m\delta$ in distinguishing between instances of between $\mathcal{D}_1'(m+1)$ and $\mathcal{D}_2(m+1)$.

In the first case, if $\mathcal{A}$ can be used to achieve advantage $2^m\delta$ in distinguishing between instances of $\mathcal{D}_1(m+1)$ and $\mathcal{D}_1'(m+1)$, then we showed in the first proof how such an algorithm could be used to distinguish between $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$ with the same advantage $(2^m\delta)$.

In the second case, if $\mathcal{A}$ can be used to achieve advantage $2^m\delta$ in distinguishing between instances of $\mathcal{D}_1'(m+1)$ and $\mathcal{D}_2(m+1)$, then we showed in the second proof how such an algorithm could be used to break the 2-Linear problem with advantage $\frac{2^m\delta}{M} = \delta$.

Therefore, if there is a polynomial time algorithm $\mathcal{A}$ which distinguishes between $\mathcal{D}_1(m+1)$ and $\mathcal{D}_2(m+1)$ with advantage $2^{m+1}\delta$, then either there exists a polynomial-time algorithm able to achieve an advantage of $2^m\delta$ in deciding between distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$ or there exists a polynomial time algorithm able to achieve an advantage of δ in the 2-Linear Problem. □

Now we have all the ingredients necessary for the main statement:

Lemma 27. *The distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ are computationally indistinguishable under the 2-Linear computational hardness assumption if $k = O(\lg poly(\lambda))$.*

Proof. We have shown that for all integers $m \geq 2$, if there exists a polynomial-time algorithm able to achieve an advantage of $2^{m+1}\delta$ deciding between distributions $\mathcal{D}_1(m+1)$ and $\mathcal{D}_2(m+1)$, then either there exists a polynomial-time algorithm able to achieve an advantage of $2^m\delta$ in

deciding between distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$ or there exists a polynomial time algorithm able to achieve an advantage of δ in the 2-Linear Problem. We have also shown that if there exists a polynomial-time algorithm able to achieve advantage $2^2\delta$ in distinguishing between the distributions $\mathcal{D}_1(2)$ and $\mathcal{D}_2(2)$, then there exists a polynomial-time algorithm able to achieve advantage δ in the 2-Linear Problem. By induction, it follows that for all m, if an algorithm is able to achieve an advantage of $2^m\delta$ in distinguishing between distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$, then that algorithm can be used to achieve advantage δ in the 2-Linear problem. Equivalently, if an algorithm is able to achieve an advantage of δ' in distinguishing between distributions $\mathcal{D}_1(m)$ and $\mathcal{D}_2(m)$, then that algorithm can be used to achieve advantage $\frac{\delta'}{2^m}$ in the 2-Linear problem.

If $k = O(\lg \text{poly}(\lambda))$, then any algorithm $\mathcal{A}$ able to achieve non-negligible advantage δ in distinguishing between $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ can be used to achieve non-negligible advantage $O(\frac{\delta}{\text{poly}(\lambda)})$ in the 2-Linear problem. This violates our 2-Linear Assumption, so no such algorithm $\mathcal{A}$ can exist. □

6 Prime Order KP-ABE

We will now show a version of our construction translated into the prime order setting. The full security of this variant will be proven solely using the 2-Linear Assumption described in section 2.2. We again present a one-use KP-ABE scheme and remind the reader that the same generic transformation from section 2.3.4 can be used to obtain a version of the scheme that works for multiple uses of attributes in access policies. First we present some additional tools used:

6.1 Prime Order Bilinear Groups

We now let G denote a bilinear group of prime order p, with bilinear map $e : G \times G \to G_T$. In addition to referring to individual elements of G, we will also consider "vectors" of group elements. For $\vec{v} = (v_1, ..., v_n) \in \mathbb{Z}_p^n$ and $g \in G$, we write $g^{\vec{v}}$ to denote the n-tuple of elements of G:

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

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

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

We define e_n to denote the product of the component wise 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.

We will use the notation $\vec{\mathbf{b}}_{1,j} = g^{\vec{b}_{1,j}}, ..., \vec{\mathbf{b}}_{9,j} = g^{\vec{b}_{9,j}}$ (and similarly for the starred vectors), where scalar multiplication of bold vectors denotes exponentiation and addition denotes the normal component-wise group operation; i.e: $a\vec{\mathbf{b}}_{1,j} + b\vec{\mathbf{b}}_{1,j} = g^{(a+b)\vec{b}_{1,j}}$. This notation allows us to avoid having to write large sums in exponents.

Dual Pairing Vector Spaces We will employ the concept of dual pairing vector spaces from [28, 29]. We will choose two random sets of vectors:

$$\begin{aligned}\mathbb{B} := \{&\vec{b}_1, \vec{b}_{2,1}, ..., \vec{b}_{2,k}, \vec{b}_3,\\ &\vec{b}_4, \vec{b}_{5,1}, ..., \vec{b}_{5,k}, \vec{b}_6,\\ &\vec{b}_7, \vec{b}_{8,1}, ..., \vec{b}_{8,k}, \vec{b}_9\}\end{aligned}$$

and

$$\begin{aligned}\mathbb{B}^* := \{&\vec{b}^*_{1,1}, ..., \vec{b}^*_{1,k}, \vec{b}^*_2, \vec{b}^*_{3,1}, ..., \vec{b}^*_{3,k},\\ &\vec{b}^*_{4,1}, ..., \vec{b}^*_{4,k}, \vec{b}^*_5, \vec{b}^*_{6,1}, ..., \vec{b}^*_{6,k},\\ &\vec{b}^*_{7,1}, ..., \vec{b}^*_{7,k}, \vec{b}^*_8, \vec{b}^*_{9,1}, ..., \vec{b}^*_{9,k}\}\end{aligned}$$

of $\mathbb{Z}_p^{3k+6}$ subject to the constraint that they are "dual orthonormal" in the following sense:

$$\begin{aligned}
&\vec{b}_1 \cdot \vec{b}^*_{1,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_1 \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_{2,j} \cdot \vec{b}^*_2 = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \text{each } \vec{b}_{2,j} \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_3 \cdot \vec{b}^*_{3,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_3 \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_4 \cdot \vec{b}^*_{4,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_4 \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_{5,j} \cdot \vec{b}^*_5 = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \text{each } \vec{b}_{5,j} \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_6 \cdot \vec{b}^*_{6,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_6 \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_7 \cdot \vec{b}^*_{7,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_7 \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_{8,j} \cdot \vec{b}^*_8 = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \text{each } \vec{b}_{8,j} \text{ is orthogonal to all other vectors in } \mathcal{B}^*\\
&\vec{b}_9 \cdot \vec{b}^*_{9,j} = 1 \,(\mathrm{mod}\, p) \text{ for all } j,\ \vec{b}_9 \text{ is orthogonal to all other vectors in } \mathcal{B}^*
\end{aligned}$$

We note that choosing sets $(\mathcal{B}, \mathcal{B}^*)$ at random from sets satisfying these dual orthonormality constraints can be realized by choosing a set of $3k+6$ vectors $\mathcal{B}$ uniformly at random from $\mathbb{Z}_p^{3k+6}$ (these vectors will be linearly independent with high probability), then determine each vector of $\mathcal{B}^*$ from its orthonormality constraints. We will denote choosing random dual orthonormal sets this way as: $(\mathcal{B}, \mathcal{B}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$

6.2 Prime Order KP-ABE Construction

In a typical execution of a dual system encryption proof strategy in the prime-order setting, we use orthogonal subspaces in the exponents to play the role of normal and semi-functional components. Since the semi-functional vectors are never published, they can serve as "hidden parameters" that supply fresh entropy, even conditioned on the public parameters. The prior prime order variants of fully secure dual system ABE schemes have used a fresh pair of vectors for each attribute use to supply enough entropy to make an information-theoretic switch from a nominal semi-functional key (one that has semi-functional components but still manages to correctly decrypt the semi-functional ciphertext) to a real semi-functional one (a key that no longer decrypts the semi-functional ciphertext correctly). This makes the public parameters allocated for each attribute grow linearly with the bound on the number of attribute uses allowed in access policies (analogously to the composite-order counterparts of these schemes).

We replace this linear dependence with a poly-logarithmic one. Essentially, for a bound of at most $2^k - 1$ attribute uses, we publish $O(k)$ vectors in a space of dimension $O(k)$. These

vectors will satisfy the previously mentioned orthonormality constraints and allow us to form $2^k - 1$ sets of well-behaved vectors by taking subset-sum combinations in the exponent. The semi-functional space vectors are not fixed conditioned on the published parameters, and the additional $O(k)$ degrees of freedom this provides allow us to argue that the $\{a_j, b_j\}$ parameters (in the construction below) in the semi-functional space are decorrelated from their normal space counterparts. (This is something that comes for free in the composite-order setting, where we can use the Chinese Remainder Theorem. In the prime-order setting, the fresh entropy is supplied by a hidden linear transformation.)

Having subset-sums of decorellated $\{a_j, b_j\}$ values in the semi-functional space allows us to apply our bilinear entropy expansion lemma in the exponent of the semi-functional space. This provides sufficient entropy to execute the information-theoretic step of a traditional dual system encryption strategy.

Again, we identify the attribute universe $\mathcal{U}$ for this single-use KP-ABE with the set of all nonempty subsets of $[k]$.

Setup$(\lambda, \mathcal{U}, k) \to PP, MSK$ The setup algorithm chooses a bilinear group G of prime order p. It then chooses random group element $g \in G$ and exponents $\alpha, \alpha' \leftarrow \mathbb{Z}_p$. For $j \in [k]$ it chooses values $a_j, b_j \leftarrow \mathbb{Z}_p$ and generates a random dual orthonormal set:

$$\begin{aligned}(\mathbb{B}, \mathbb{B}^*) = (\{&\vec{b}_1, \vec{b}_{2,1}, ..., \vec{b}_{2,k}, \vec{b}_3,\\ &\vec{b}_4, \vec{b}_{5,1}, ..., \vec{b}_{5,k}, \vec{b}_6,\\ &\vec{b}_7, \vec{b}_{8,1}, ..., \vec{b}_{8,k}, \vec{b}_9\},\\ \{&\vec{b}^*_{1,1}, ..., \vec{b}^*_{1,k}, \vec{b}^*_2, \vec{b}^*_{3,1}, ..., \vec{b}^*_{3,k},\\ &\vec{b}^*_{4,1}, ..., \vec{b}^*_{4,k}, \vec{b}^*_5, \vec{b}^*_{6,1}, ..., \vec{b}^*_{6,k},\\ &\vec{b}^*_{7,1}, ..., \vec{b}^*_{7,k}, \vec{b}^*_8, \vec{b}^*_{9,1}, ..., \vec{b}^*_{9,k}\}) \leftarrow Dual(\mathbb{Z}_p^{3k+6})\end{aligned}$$

The public parameters PP are:

$$\begin{aligned}&k, p, g, e(g,g)^{\alpha}, e(g,g)^{\alpha'},\\ &\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\}\\ &\{a_j\vec{\mathbf{b}}^*_2, b_j\vec{\mathbf{b}}^*_2, a_j\vec{\mathbf{b}}^*_5, b_j\vec{\mathbf{b}}^*_5 : j \in [k]\}\end{aligned}$$

(We use the bolded vector notation to denote the vector in the exponent, as detailed in subsection 6.1.) The MSK is:

$$\begin{aligned}&\alpha, \alpha', \{\vec{\mathbf{b}}_1, \vec{\mathbf{b}}_{2,j}, \vec{\mathbf{b}}_4, \vec{\mathbf{b}}_{5,j} : j \in [k]\},\\ &\{a_j\vec{\mathbf{b}}_1, b_j\vec{\mathbf{b}}_3, a_j\vec{\mathbf{b}}_4, b_j\vec{\mathbf{b}}_6 : j \in [k]\}\end{aligned}$$

Such a construction is equipped to create keys for access policies which include attributes $K \subseteq [k]$ where K is not empty.

KeyGen$(MSK, \Lambda, PP) \to SK$ The key generation algorithm takes in the public parameters, master secret key, and LSSS access matrix Λ. First, the key generation algorithm generates $\{\lambda_K, \lambda'_K\}$: linear sharings of α and α' according to policy matrix Λ (the reader is referred to

section 2.3.1 for details). For each attribute K labeling a row in the policy matrix Λ, it then chooses exponents $y_K, y'_K \leftarrow \mathbb{Z}_p$ and computes:

$$A_K\vec{\mathbf{b}}_1 = \sum_{j\in K} a_j\vec{\mathbf{b}}_1$$
$$B_K\vec{\mathbf{b}}_3 = \sum_{j\in K} b_j\vec{\mathbf{b}}_3$$
$$A_K\vec{\mathbf{b}}_4 = \sum_{j\in K} a_j\vec{\mathbf{b}}_4$$
$$B_K\vec{\mathbf{b}}_6 = \sum_{j\in K} b_j\vec{\mathbf{b}}_6$$

It then outputs the secret key:

$$SK_\Lambda = \{\lambda_K\vec{\mathbf{b}}_1 + y_K A_K\vec{\mathbf{b}}_1 + y_K\left(\sum_{j\in K}\vec{\mathbf{b}}_{2,j}\right) + y_K B_K\vec{\mathbf{b}}_3$$
$$+\lambda'_K\vec{\mathbf{b}}_4 + y'_K A_K\vec{\mathbf{b}}_4 + y'_K\left(\sum_{j\in K}\vec{\mathbf{b}}_{5,j}\right) + y'_K B_K\vec{\mathbf{b}}_6$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

Again, note that here and throughout the rest of the description of our construction and its proof of security we will use the notation $A_K = \sum_{j\in K} a_j$ and $B_K = \sum_{j\in K} b_j$.

Encrypt$(M, S, PP) \rightarrow CT$ The encryption algorithm first draws $s, s' \leftarrow \mathbb{Z}_p$. For each $K \in S$, the encryption algorithm draws $t_K, t'_K \leftarrow \mathbb{Z}_p$ and computes:

$$A_K\vec{\mathbf{b}}^*_2 = \sum_{j\in K} a_j\vec{\mathbf{b}}^*_2$$
$$B_K\vec{\mathbf{b}}^*_2 = \sum_{j\in K} b_j\vec{\mathbf{b}}^*_2$$
$$A_K\vec{\mathbf{b}}^*_5 = \sum_{j\in K} a_j\vec{\mathbf{b}}^*_5$$
$$B_K\vec{\mathbf{b}}^*_5 = \sum_{j\in K} b_j\vec{\mathbf{b}}^*_5$$

It then outputs the ciphertext:

$$CT = Me(g,g)^{\alpha s+\alpha' s'}, \{s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$:(\forall K \in S)\}$$

(This implicitly includes S)

Decrypt$(CT, SK, PP) \rightarrow M$ We let S correspond to the set of attributes associated to ciphertext CT, and Λ be the policy matrix. If S satisfies Λ, the decryption algorithm computes constants ω_K such that $\sum_{K \in S^*} \omega_K \lambda_K = \alpha$ and $\sum_{K \in S^*} \omega_K \lambda'_K = \alpha'$ (recall section 2.3.1). It then computes:

$$\prod_{K \in S^*} e_n \Bigg(s \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{1,j} \right) - s A_K \vec{\mathbf{b}}^*_2 - t_K B_K \vec{\mathbf{b}}^*_2 + t_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{3,j} \right) + s' \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{4,j} \right) - s' A_K \vec{\mathbf{b}}^*_5 - t'_K B_K \vec{\mathbf{b}}^*_5 + t'_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{6,j} \right), \lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3 + \lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6 \Bigg)^{\frac{\omega_K}{|K|}}$$

$$= \prod_{K \in S^*} \Bigg(\prod_{j \in K} e(g,g)^{s \lambda_K \vec{b}_1 \cdot \vec{b}^*_{1,j}} \prod_{j \in K} e(g,g)^{s y_K A_K \vec{b}_1 \cdot \vec{b}^*_{1,j}} \prod_{j \in K} e(g,g)^{-s y_K A_K \vec{b}_{2,j} \cdot \vec{b}^*_2} \prod_{j \in K} e(g,g)^{-y_K t_K B_K \vec{b}_{2,j} \cdot \vec{b}^*_2} \prod_{j \in K} e(g,g)^{t_K y_K B_K \vec{b}_3 \cdot \vec{b}^*_{3,j}} \prod_{j \in K} e(g,g)^{s' \lambda'_K \vec{b}_4 \cdot \vec{b}^*_{4,j}} \prod_{j \in K} e(g,g)^{s' y'_K A_K \vec{b}_4 \cdot \vec{b}^*_{4,j}} \prod_{j \in K} e(g,g)^{-s' y'_K A_K \vec{b}_{5,j} \cdot \vec{b}^*_5} \prod_{j \in K} e(g,g)^{-y'_K t'_K B_K \vec{b}_{5,j} \cdot \vec{b}^*_5} \prod_{j \in K} e(g,g)^{t'_K y'_K B_K \vec{b}_6 \cdot \vec{b}^*_{6,j}} \Bigg)^{\frac{\omega_K}{|K|}}$$

$$= \prod_{K \in S^*} \Big(e(g,g)^{s \lambda_K |K|} e(g,g)^{s y_K A_K |K|} e(g,g)^{-s y_K A_K |K|} e(g,g)^{-y_K t_K B_K |K|} e(g,g)^{t_K y_K B_K |K|} e(g,g)^{s' \lambda'_K |K|} e(g,g)^{s' y'_K A_K |K|} e(g,g)^{-s' y'_K A_K |K|} e(g,g)^{-y'_K t'_K B_K |K|} e(g,g)^{t'_K y'_K B_K |K|} \Big)^{\frac{\omega_K}{|K|}}$$

$$= \prod_{K \in S^*} (e(g,g)^{s\lambda_K} e(g,g)^{s'\lambda'_K})^{\omega_K}$$
$$= e(g,g)^{s \sum_{K \in S^*} \omega_K \lambda_K + s' \sum_{K \in S^*} \omega_K \lambda'_K}$$
$$= e(g,g)^{s\alpha + s'\alpha'}$$

The message can then be recovered by computing: $Me(g,g)^{\alpha s + \alpha' s'} / e(g,g)^{\alpha s + \alpha' s'} = M$. This also shows correctness of the scheme.

Security The proof of security for this prime order variant (which holds under the 2-Linear Assumption) is presented in appendix A.

7 Concluding Remarks

We have presented a prime order KP-ABE scheme fully secure under the DLIN assumption and an analogous composite order scheme fully secure under additional subgroup decision type assumptions. Both schemes allow a bound of $2^k - 1$ attribute-uses in an access policy, where the number of group elements required in the public parameters per attribute-use grows polynomially with k. Previous fully secure schemes required either a q-type assumption or public parameters growing linearly with $2^k - 1$. An interesting question for future work is whether the ciphertext sizes can be significantly reduced (our schemes have ciphertexts still growing linearly in size with $2^k - 1$).

We have chosen to demonstrate our techniques on KP-ABE schemes, though we note that they are equally applicable to the CP-ABE setting. The core of CP-ABE schemes often mirror the structure of KP-ABE schemes, and would benefit similarly from the reduced public parameter size our lemma enables.

Finally, our bilinear entropy expansion lemma is not restricted to the ABE setting, and we suspect it may have applications to other cryptographic primitives. Primitive structure can be built around the lemma's core components of $\{g^{t_K}, g^{t_K A_K}\}$, which can be plugged in to replace a need for independent random group elements. Our prime order and composite order KP-ABE schemes demonstrate this usage.

References

[1] N. Attrapadung. Dual system encryption via doubly selective security: Framework, fully secure functional encryption for regular languages, and more. In *EUROCRYPT*, pages 557–577, 2014.

[2] M. Raykova B. Parno and V. Vaikuntanathan. How to delegate and verify in public: Verifiable computation from attribute-based encryption. In *TCC*, pages 422–439, 2012.

[3] A. Beimel. Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

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

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

[7] M. Chase. Multi-authority attribute based encryption. In *Theory of Cryptography, 4th Theory of Cryptography Conference, TCC 2007, Amsterdam, The Netherlands, February 21-24, 2007, Proceedings*, pages 515–534, 2007.

[8] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security*, pages 121–130, 2009.

[9] M. Chase and S. Meiklejohn. Déjà Q: using dual systems to revisit q-type assumptions. In *EUROCRYPT*, pages 622–639, 2014.

[10] J. Chen and H. Wee. Fully, (almost) tightly secure IBE and dual system groups. In *CRYPTO*, pages 435–460, 2013.

[11] 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.

[12] Y. Dodis, A. B. Lewko, B. Waters, and D. Wichs. Storing secrets on continually leaky devices. In *FOCS*, pages 688–697, 2011.

[13] S. Garg, C. Gentry, S. Halevi, A. Sahai, and B. Waters. Attribute-based encryption for circuits from multilinear maps. In *CRYPTO*, pages 479–499, 2013.

[14] S. Garg, C. Gentry, S. Halevi, and M. Zhandry. Fully secure attribute based encryption from multilinear maps. *IACR Cryptology ePrint Archive*, 2014:622, 2014.

[15] S. Goldwasser, Y. T. Kalai, R. A. Popa, V. Vaikuntanathan, and N. Zeldovich. How to run turing machines on encrypted data. In *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, pages 536–553, 2013.

[16] S. Gorbunov, V. Vaikuntanathan, and H. Wee. Attribute-based encryption for circuits. In *STOC*, pages 545–554, 2013.

[17] V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP*, 2008.

[18] 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.

[19] 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.

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

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

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

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

[24] A. B. Lewko, M. Lewko, and B. Waters. How to leak on key updates. In *STOC*, pages 725–734, 2011.

[25] A. B. Lewko and B. Waters. Efficient pseudorandom functions from the decisional linear assumption and weaker variants. In *Proceedings of the 2009 ACM Conference on Computer and Communications Security*, pages 112–120, 2009.

[26] A. B. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques. In *CRYPTO*, pages 180–198, 2012.

[27] M. Naor and O. Reingold. Number-theoretic constructions of efficient pseudo-random functions. In *FOCS*, pages 458–467, 1997.

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

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

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

[31] T. Okamoto and K. Takashima. Fully secure unbounded inner-product and attribute-based encryption. In *ASIACRYPT*, pages 349–366, 2012.

[32] T. Okamoto and K. Takashima. Decentralized attribute-based signatures. In *PKC*, pages 125–142, 2013.

[33] 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.

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

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

[36] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC*, pages 53–70, 2011.

[37] H. Wee. Dual system encryption via predicate encodings. In *TCC*, pages 616–637, 2014.

A Prime Order KP-ABE Security Proof

Our security proof for the prime order variant again uses a hybrid argument over a sequence of games. We let Game_{real} denote the real security game. The rest of the games use semi-functional keys and ciphertexts, which we will now describe.

Semi-functional Ciphertext We will use 4 types of semi-functional ciphertexts. To produce a semi-functional ciphertext for an attribute set S, one first calls the normal encryption algorithm to produce a normal ciphertext consisting of:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$:(\forall K\in S)\}$$

One then chooses $s''\leftarrow\mathbb{Z}_p$. For each attribute K, an exponent $t''_K\leftarrow\mathbb{Z}_p$ is drawn. For all $j\in[k]$, $a''_j, b''_j\leftarrow\mathbb{Z}_p$ are drawn and fixed if they do not already exist (in a semi-functional key, for instance). We will use the notation $\tilde{A}_K=\sum_{j\in K}a''_j$ and $\tilde{B}_K=\sum_{j\in K}b''_j$. The remaining composition of the semifunctional ciphertext depends on the type of ciphertext desired:

Type 0 The semi-functional ciphertext of Type 0 is formed as:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''A_K\vec{\mathbf{b}}^*_8 - t''_K B_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)$$
$$:(\forall K\in S)\}$$

Notice that in this type, the a''_j, b''_j are unused.

Type 1 The semi-functional ciphertext of Type 1 is formed as:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''\tilde{A}_K\vec{\mathbf{b}}^*_8 - t''_K \tilde{B}_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)$$
$$:(\forall K\in S)\}$$

Notice that in this type, the subset sums in the extra semifunctional space have been decoupled from the sums in the normal space (by being replaced with subset sums of the independently chosen a''_j and b''_j).

Type 2 The semi-functional ciphertext of Type 2 is formed as:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{ s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''\tilde{A}_K\vec{\mathbf{b}}^*_8 - t''_K\tilde{b}_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)$$
$$:(\forall K\in S)\}$$

for fixed $\tilde{b}_K \in \mathbb{Z}_p$ which are chosen uniformly at random and fixed if they do not already exist.

Type 3 The semi-functional ciphertext of Type 3 is formed as:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{ s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''\tilde{a}_K\vec{\mathbf{b}}^*_8 - t''_K\tilde{b}_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)$$
$$:(\forall K\in S)\}$$

for fixed $\tilde{a}_K, \tilde{b}_K \in \mathbb{Z}_p$ which are chosen uniformly at random and fixed if they do not already exist.

Semi-functional Keys We will use 9 types of semi-functional keys. To produce a semi-functional key for an access policy Λ, one first calls the normal key generation algorithm to produce a normal key consisting of:

$$\{\lambda_K\vec{\mathbf{b}}_1 + y_K A_K\vec{\mathbf{b}}_1 + y_K\left(\sum_{j\in K}\vec{\mathbf{b}}_{2,j}\right) + y_K B_K\vec{\mathbf{b}}_3$$
$$+\lambda'_K\vec{\mathbf{b}}_4 + y'_K A_K\vec{\mathbf{b}}_4 + y'_K\left(\sum_{j\in K}\vec{\mathbf{b}}_{5,j}\right) + y'_K B_K\vec{\mathbf{b}}_6$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

The first 8 types of keys fall under 4 classes which have two variants each: a "Z" variant and an "R" variant. For Z-type keys one computes a linear sharing *of 0* under access policy Λ, creating shares λ''_K. For R-type keys one computes a linear sharing of *a random element* u of $\mathbb{Z}_p$ which is fixed and used in all R type keys. u is shared under access policy Λ, creating shares λ''_K. For each $j \in [k]$, $a''_j, b''_j \leftarrow \mathbb{Z}_p$ are drawn and fixed if they do not already exist (in a semifunctional ciphertext, for instance). Recall that we use the notation $\tilde{A}_K = \sum_{j\in K} a''_j$ and $\tilde{B}_K = \sum_{j\in K} b''_j$. The next steps depend on the class of the key:

Class 0 For each K in the honest key, one then chooses a new $y''_K \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 0Z or 0R (depending on the sharing λ''_K) as:

$$\{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3$$

$$+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6$$

$$+\lambda''_K \vec{\mathbf{b}}_7 + y''_K A_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + y''_K B_K \vec{\mathbf{b}}_9$$

$$:(\forall K \text{ labels} \in \Lambda)\}$$

Notice that in this type, the a''_j and b''_j are unused.

Class 1 For each K in the honest key, one then chooses a new $y''_K \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 1Z or 1R (depending on the sharing λ''_K) as:

$$\{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3$$

$$+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6$$

$$+\lambda''_K \vec{\mathbf{b}}_7 + y''_K \tilde{A}_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + y''_K \tilde{B}_K \vec{\mathbf{b}}_9$$

$$:(\forall K \text{ labels} \in \Lambda)\}$$

Notice that in this type, the subset sums in the extra semifunctional space have been decoupled from the sums in the normal space (by being replaced with subset sums of the independently chosen a''_j and b''_j).

Class 2 *Random values* $\tilde{b}_K \leftarrow \mathbb{Z}_p$ are chosen if they do not already exist (in a semi-functional ciphertext, for instance) and fixed. For each K in the honest key, one then chooses a new $y''_K \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 2Z or 2R as:

$$\{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3$$

$$+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6$$

$$+\lambda''_K \vec{\mathbf{b}}_7 + y''_K \tilde{A}_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + y''_K \tilde{b}_K \vec{\mathbf{b}}_9$$

$$:(\forall K \text{ labels} \in \Lambda)\}$$

Class 3 *Random values* $\tilde{a}_K, \tilde{b}_K \leftarrow \mathbb{Z}_p$ are chosen if they do not already exist (in a semi-functional ciphertext, for instance) and fixed. For each K in the honest key, one then chooses a new $y''_K \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 3Z or 3R as:

$$\begin{aligned}
&\{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3 \\
&+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6 \\
&+\lambda''_K \vec{\mathbf{b}}_7 + y''_K \tilde{a}_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + y''_K \tilde{b}_K \vec{\mathbf{b}}_9 \\
&:(\forall K \text{ labels} \in \Lambda)\}
\end{aligned}$$

Note that we now have defined 8 types of keys: 0Z, 0R, 1Z, 1R, 2Z, 2R, 3Z, and 3R, where the number denotes the class and the letter (Z/R) describes whether the λ''_K share zero or a random element of $\mathbb{Z}_p$ respectively. There is one final type of key: type 4R:

Type 4R Using shares λ''_K of u (which is randomly chosen from $\mathbb{Z}_p$ and fixed if it has not already been fixed), one forms the semi-functional key of type 4R as:

$$\begin{aligned}
&\{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3 \\
&+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6 \\
&+\lambda''_K \vec{\mathbf{b}}_7 \\
&:(\forall K \text{ labels} \in \Lambda)\}
\end{aligned}$$

Proof Structure Our hybrid proof takes place over a series of games defined as follows: Letting Q denote the total number of key queries that the attacker makes, we define Game_{ℓ_0}, Game_{ℓ_1}, Game_{ℓ_2}, Game_{ℓ_3}, Game_{ℓ_4}, Game_{ℓ_5}, Game_{ℓ_6}, Game_{ℓ_7}, and Game_{ℓ_8} for $\ell = 0, ..., Q$. In each game, the first $\ell - 1$ keys are semi-functional of type 4R, and all keys after the ℓth request are normal. They differ in the construction of the ℓth key and the ciphertext as follows:

Game_{ℓ_0} In this game, the ℓth key is type 0Z and the ciphertext is type 0.

Game_{ℓ_1} In this game, the ℓth key is type 1Z and the ciphertext is type 1.

Game_{ℓ_2} In this game, the ℓth key is type 2Z and the ciphertext is type 2.

Game_{ℓ_3} In this game, the ℓth key is type 3Z and the ciphertext is type 3.

Game_{ℓ_4} In this game, the ℓth key is type 3R and the ciphertext is type 3.

Game_{ℓ_5} In this game, the ℓth key is type 2R and the ciphertext is type 2.

Game$_{\ell_6}$ In this game, the ℓth key is type 1R and the ciphertext is type 1.

Game$_{\ell_7}$ In this game, the ℓth key is type 0R and the ciphertext is type 0.

Game$_{\ell_8}$ In this game, the ℓth key is type 4R and the ciphertext is type 0.

Note that under this definition, we have that in **Game**$_{0_8}$, the ciphertext given to the attacker is type 0 and the keys are all normal.

The outer structure of our hybrid argument will progress as follows. First, we transition from Game$_{real}$ to Game$_{0_8}$, then to Game$_{1_0}$, next to Game$_{1_1}$, next to Game$_{1_2}$, next to Game$_{1_3}$, next to Game$_{1_4}$, next to Game$_{1_5}$, next to Game$_{1_6}$, next to Game$_{1_7}$, next to Game$_{1_8}$ and then to Game$_{2_0}$ and so on. We then arrive at Game$_{Q_8}$, where the ciphertext is semifunctional of type 0 and *all* of the keys given to the attacker are semi-functional of type 4R.

There are two more games in the security proof: Game$_{penultimate}$ and Game$_{final}$. We transition from Game$_{Q_8}$ to Game$_{penultimate}$ and lastly to Game$_{final}$ which will complete our proof. The games are defined as follows:

Game$_{penultimate}$ In this game, all keys are semi-functional of type 0R and the ciphertext is semi-functional of type 0.

Game$_{final}$ uses a semi-functional ciphertext of a new type: type X, which we will now define:

Type X The semi-functional ciphertext of Type X is formed as:

$$\begin{aligned} Me(g,g)^{\alpha s+\alpha' s'}, \{ & x\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - xA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right) \\ & + s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right) \\ & + s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''A_K\vec{\mathbf{b}}^*_8 - t''_K B_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right) \\ & : (\forall K \in S)\} \end{aligned}$$

for an $x \leftarrow \mathbb{Z}_p$.

Game$_{final}$ In this game, all keys are semi-functional of type 0R and the ciphertext is semi-functional of type X.

Note that a ciphertext of type X information-theoretically hides its message M because the $e(g,g)$ blinding factor is raised to an exponent s which is unused anywhere else. So, in Game$_{final}$, no polynomial time adversary will be able to achieve advantage in the security game, completing our proof.

Our hybrid argument is accomplished in the following lemmas:

Lemma 28. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game$_{real}$ and Game$_{0_8}$.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between Game$_{real}$ and Game$_{0_8}$, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the 2-Linear Problem as follows:

Given $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ and $T = g^{c_1+c_2+r} \in G$, where either $r = 0$ or is a uniform random element of $\mathbb{Z}_p$, consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g, choosing $\tilde{\alpha}, \tilde{\alpha}', a_j, b_j \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

The simulator then implicitly defines the sets $(\mathbb{B}, \mathbb{B}^*)$ as:

$$\begin{aligned}
&\vec{b}^*_{1,j} = y_1 \vec{d}^*_{1,j} + y_1 c_1 \vec{d}^*_{7,j} && \vec{b}^*_2 = y_1 \vec{d}^*_2 + y_1 c_1 \vec{d}^*_8 && \vec{b}^*_{3,j} = y_1 \vec{d}^*_{3,j} + y_1 c_1 \vec{d}^*_{9,j} \\
&\vec{b}^*_{4,j} = y_2 \vec{d}^*_{4,j} + y_2 c_2 \vec{d}^*_{7,j} && \vec{b}^*_5 = y_2 \vec{d}^*_5 + y_2 c_2 \vec{d}^*_8 && \vec{b}^*_{6,j} = y_2 \vec{d}^*_{6,j} + y_2 c_2 \vec{d}^*_{9,j} \\
&\vec{b}^*_{7,j} = \vec{d}^*_{7,j} && \vec{b}^*_8 = \vec{d}^*_8 && \vec{b}^*_{9,j} = \vec{d}^*_{9,j}
\end{aligned}$$

$$\begin{aligned}
&\vec{b}_1 = y_1^{-1} \vec{d}_1 && \vec{b}_{2,j} = y_1^{-1} \vec{d}_{2,j} && \vec{b}_3 = y_1^{-1} \vec{d}_3 \\
&\vec{b}_4 = y_2^{-1} \vec{d}_4 && \vec{b}_{5,j} = y_2^{-1} \vec{d}_{5,j} && \vec{b}_6 = y_2^{-1} \vec{d}_6 \\
&\vec{b}_7 = \vec{d}_7 - c_1 \vec{d}_1 - c_2 \vec{d}_4 && \vec{b}_{8,j} = \vec{d}_{8,j} - c_1 \vec{d}_{2,j} - c_2 \vec{d}_{5,j} && \vec{b}_9 = \vec{d}_9 - c_1 \vec{d}_3 - c_2 \vec{d}_6
\end{aligned}$$

(The distribution of the sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to that produced by $Dual(\mathbb{Z}_p^{3k+6})$, since there is a one to one mapping between any sets produced this way and the sets produced by the $Dual(\mathbb{Z}_p^{3k+6})$ procedure.)

The public parameters are constructed as:

$$\begin{aligned}
&p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1 \tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2 \tilde{\alpha}'}, \\
&\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\} \\
&\{a_j \vec{\mathbf{b}}^*_2, b_j \vec{\mathbf{b}}^*_2, a_j \vec{\mathbf{b}}^*_5, b_j \vec{\mathbf{b}}^*_5 : j \in [k]\}
\end{aligned}$$

implicitly defining $\alpha = y_1 \tilde{\alpha}$ and $\alpha' = y_2 \tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be made by the simulator by taking combinations of $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ raised to the appropriate vectors.)

The simulator then gives the public parameters to $\mathcal{A}$. To respond to key requests for policies Λ (all keys are honest in both games), the simulator first computes $\tilde{\lambda}_K, \tilde{\lambda}'_K$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively. For each K attribute label in Λ, it generates $\tilde{y}_K, \tilde{y}'_K \leftarrow \mathbb{Z}_p$ and outputs:

$$\begin{aligned}
SK_\Lambda = \{&\tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K A_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + \tilde{y}_K B_K y_1 \vec{\mathbf{b}}_3 \\
&+ \tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K A_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + \tilde{y}'_K B_K y_2 \vec{\mathbf{b}}_6 \\
&: (\forall K \text{ labels} \in \Lambda)\}
\end{aligned}$$

Note that the simulator cannot make any of the $\vec{\mathbf{b}}_{x,j}$ in this honest key alone (because of the y_1^{-1}, y_2^{-1} terms). However, the simulator is able to make $y_1 \vec{\mathbf{b}}_{1,j} = \vec{\mathbf{d}}_{1,j}$, for example, so it can construct keys as described above. The distribution of keys constructed this way is identical to the distribution of normal honest keys where $y_K = \tilde{y}_K y_1$ and $y'_K = \tilde{y}'_K y_2$, which are uniformly distributed elements of $\mathbb{Z}_p$. The shares $\lambda_K = \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}'_K y_2$ are then shares of $\tilde{\alpha} y_1$ and $\tilde{\alpha}' y_2$ respectively, which are uniformly distributed elements of $\mathbb{Z}_p$ and appropriately matching in the public parameters.

To return the challenge ciphertext for a set of attributes S, first, for each $K \in S$, $\tilde{t}_K, \tilde{t}'_K, \tilde{t}''_K \leftarrow \mathbb{Z}_p$ are drawn. The following ciphertext is then constructed and provided:

$$
\begin{aligned}
&Me(g,g)^{\tilde{\alpha}} e(g,g)^{\tilde{\alpha}'},\\
&\{\prod_{j\in K}\left[g^{\vec{d}^*_{1,j}} g^{\vec{d}^*_{4,j}} T^{\vec{d}^*_{7,j}}\right]\\
&(g^{\vec{d}^*_2} g^{\vec{d}^*_5} T^{\vec{d}^*_8})^{-A_K}\\
&\left(g^{\vec{d}^*_2} g^{\vec{d}^*_5} T^{\vec{d}^*_8}\right)^{-\tilde{t}''_K B_K} \left((g^{y_1})^{\vec{d}^*_2} (g^{y_1 c_1})^{\vec{d}^*_8}\right)^{-\tilde{t}_K B_K} \left((g^{y_2})^{\vec{d}^*_5} (g^{y_2 c_2})^{\vec{d}^*_8}\right)^{-\tilde{t}'_K B_K}\\
&\prod_{j\in K}\left[\left(g^{\vec{d}^*_{3,j}} g^{\vec{d}^*_{6,j}} T^{\vec{d}^*_{9,j}}\right)^{\tilde{t}''_K} \left((g^{y_1})^{\vec{d}^*_{3,j}} (g^{y_1 c_1})^{\vec{d}^*_{9,j}}\right)^{\tilde{t}_K} \left((g^{y_2})^{\vec{d}^*_{6,j}} (g^{y_2 c_2})^{\vec{d}^*_{9,j}}\right)^{\tilde{t}'_K}\right]\\
&:(\forall K \in S)\}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$s equal to:

$$
\begin{aligned}
&Me(g,g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'},\\
&\{y_1^{-1}\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - y_1^{-1}A_K\vec{\mathbf{b}}^*_2 - (\tilde{t}_K + y_1^{-1}\tilde{t}''_K)B_K\vec{\mathbf{b}}^*_2 + (\tilde{t}_K + y_1^{-1}\tilde{t}''_K)\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)\\
&+y_2^{-1}\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - y_2^{-1}A_K\vec{\mathbf{b}}^*_5 - (\tilde{t}'_K + y_2^{-1}\tilde{t}''_K)B_K\vec{\mathbf{b}}^*_5 + (\tilde{t}'_K + y_2^{-1}\tilde{t}''_K)\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)\\
&+r\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - rA_K\vec{\mathbf{b}}^*_8 - \tilde{t}''_K rB_K\vec{\mathbf{b}}^*_8 + \tilde{t}''_K r\ \left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)\\
&:(\forall K \in S)\}
\end{aligned}
$$

Notice that for $T = g^{c_1+c_2+r}$, if $r = 0$, then the distribution of ciphertexts formed is identical to the honest case where $s = y_1^{-1}$, $s' = y_2^{-1}$, $t_K = \tilde{t}_K + y_1^{-1}\tilde{t}''_K$ and $t'_K = \tilde{t}'_K + y_2^{-1}\tilde{t}''_K$, which are distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{real}.

If r is a uniform randomly chosen element of $\mathbb{Z}_p$, the ciphertext formed is distributed exactly like a semi-functional ciphertext of type 0 where $s = y_1^{-1}$, $s' = y_2^{-1}$, $s'' = r$, $t_K = \tilde{t}_K + y_1^{-1}\tilde{t}''_K, t'_K = \tilde{t}'_K + y_2^{-1}\tilde{t}''_K$, and $t''_K = \tilde{t}''_K r$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{0_8} .

Therefore, any adversary with non-negligible difference in advantage between Game_{real} and Game_{0_8} could be used to achieve the same non-negligible advantage in deciding the 2-Linear Problem. By assumption this is not possible, so such an adversary cannot exist.

□

Lemma 29. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $\text{Game}_{(\ell-1)_8}$ and Game_{ℓ_0} for any ℓ from 1 to Q.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between $\text{Game}_{(\ell-1)_8}$ and Game_{ℓ_0} for some ℓ in $\{1, ..., Q\}$, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the 2-Linear Problem as follows:

Given $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ and $T = g^{c_1+c_2+r} \in G$, where either $r = 0$ or is a uniform random element of $\mathbb{Z}_p$, consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g, choosing $\tilde{\alpha}, \tilde{\alpha}', a_j, b_j \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

The simulator then implicitly defines the sets $(\mathbb{B}, \mathbb{B}^*)$ as:

$$\begin{array}{lll}
\vec{b}^*_{1,j} = \vec{d}^*_{1,j} & \vec{b}^*_2 = \vec{d}^*_2 & \vec{b}^*_{3,j} = \vec{d}^*_{3,j} \\
\vec{b}^*_{4,j} = \vec{d}^*_{4,j} & \vec{b}^*_5 = \vec{d}^*_5 & \vec{b}^*_{6,j} = \vec{d}^*_{6,j} \\
\vec{b}^*_{7,j} = \vec{d}^*_{7,j} - c_1\vec{d}^*_{1,j} - c_2\vec{d}^*_{4,j} & \vec{b}^*_8 = \vec{d}^*_8 - c_1\vec{d}^*_2 - c_2\vec{d}^*_5 & \vec{b}^*_{9,j} = \vec{d}^*_{9,j} - c_1\vec{d}^*_{3,j} - c_2\vec{d}^*_{6,j}
\end{array}$$

$$\begin{array}{lll}
\vec{b}_1 = \vec{d}_1 + c_1\vec{d}_7 & \vec{b}_{2,j} = \vec{d}_{2,j} + c_1\vec{d}_{8,j} & \vec{b}_3 = \vec{d}_3 + c_1\vec{d}_9 \\
\vec{b}_4 = \vec{d}_4 + c_2\vec{d}_7 & \vec{b}_{5,j} = \vec{d}_{5,j} + c_2\vec{d}_{8,j} & \vec{b}_6 = \vec{d}_6 + c_2\vec{d}_9 \\
\vec{b}_7 = \vec{d}_7 & \vec{b}_{8,j} = \vec{d}_{8,j} & \vec{b}_9 = \vec{d}_9
\end{array}$$

(The distribution of sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to that produced by $Dual(\mathbb{Z}_p^{3k+6})$, since there is a one to one mapping between any sets produced this way and the sets produced by the $Dual(\mathbb{Z}_p^{3k+6})$ procedure.)

The public parameters are constructed as:

$$\begin{array}{l}
p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\} \\
\{a_j\vec{\mathbf{b}}^*_2, b_j\vec{\mathbf{b}}^*_2, a_j\vec{\mathbf{b}}^*_5, b_j\vec{\mathbf{b}}^*_5 : j \in [k]\}
\end{array}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be easily made by the simulator from the $\vec{d}^*$ vectors)

The simulator then gives the public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S when it is requested, first, $s'' \leftarrow \mathbb{Z}_p$ is chosen. Then, for each $K \in S$, $\tilde{t}_K, \tilde{t}'_K, \tilde{t}''_K \leftarrow \mathbb{Z}_p$ are chosen. The following ciphertext is then constructed and provided:

$$\begin{array}{l}
Me(g, g^{y_1c_1})^{\tilde{\alpha}s''} e(g, g^{y_2c_2})^{\tilde{\alpha}'s''}, \\
\{ \prod_{j\in K} (g^{\vec{d}^*_{7,j}})^{s''} \\
(g^{\vec{d}^*_8})^{-s''A_K} \\
(g^{\vec{d}^*_2})^{-\tilde{t}_K B_K} (g^{\vec{d}^*_5})^{-\tilde{t}'_K B_K} (g^{\vec{d}^*_8})^{-t''_K B_K} \\
\prod_{j\in K} \left[(g^{\vec{d}^*_{3,j}})^{\tilde{t}_K} (g^{\vec{d}^*_{6,j}})^{\tilde{t}'_K} (g^{\vec{d}^*_{9,j}})^{t''_K} \right] \\
: (\forall K \in S)\}
\end{array}$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
& Me(g,g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'}, \\
& \{c_1 s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{1,j}\right) - c_1 s'' A_K \vec{\mathbf{b}}^*_2 - (\tilde{t}_K + c_1 t''_K) B_K \vec{\mathbf{b}}^*_2 + (\tilde{t}_K + c_1 t''_K) \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{3,j}\right) \\
& + c_2 s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{4,j}\right) - c_2 s'' A_K \vec{\mathbf{b}}^*_5 - (\tilde{t}'_K + c_2 t''_K) B_K \vec{\mathbf{b}}^*_5 + (\tilde{t}'_K + c_2 t''_K) \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{6,j}\right) \\
& + s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{7,j}\right) - s'' A_K \vec{\mathbf{b}}^*_8 - t''_K B_K \vec{\mathbf{b}}^*_8 + t''_K \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{9,j}\right) \\
& : (\forall K \in S)\}
\end{aligned}
$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$, $t_K = \tilde{t}_K + c_1 t''_K$, and $t'_K = \tilde{t}'_K + c_2 t''_K$ are all independently and uniformly randomly distributed in $\mathbb{Z}_p$.

To respond to key requests for policies Λ, the simulator first computes an honest key by creating $\tilde{\lambda}_K, \tilde{\lambda}'_K$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ under policy Λ respectively.

$$
\begin{aligned}
SK_\Lambda = \{ & \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{\lambda}_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{\lambda}'_K} \\
& \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{y}_K A_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{y}'_K A_K} \\
& \prod_{j\in K} \left[\left((g^{y_1})^{\vec{d}_{2,j}} (g^{y_1 c_1})^{\vec{d}_{8,j}}\right)^{\tilde{y}_K} \left((g^{y_2})^{\vec{d}_{5,j}} (g^{y_2 c_2})^{\vec{d}_{8,j}}\right)^{\tilde{y}'_K}\right] \\
& \left((g^{y_1})^{\vec{d}_3} (g^{y_1 c_1})^{\vec{d}_9}\right)^{\tilde{y}_K B_K} \left((g^{y_2})^{\vec{d}_5} (g^{y_2 c_2})^{\vec{d}_9}\right)^{\tilde{y}'_K B_K} \\
& : (\forall K \text{ labels} \in \Lambda)\}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
SK_\Lambda = \{ & \tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 A_K \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 \left(\sum_{j\in K} \vec{\mathbf{b}}_{2,j}\right) + \tilde{y}_K y_1 B_K \vec{\mathbf{b}}_3 \\
& + \tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 A_K \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 \left(\sum_{j\in K} \vec{\mathbf{b}}_{5,j}\right) + \tilde{y}'_K y_2 B_K \vec{\mathbf{b}}_6 \\
& : (\forall K \text{ labels} \in \Lambda)\}
\end{aligned}
$$

The distribution of keys constructed this way is identical to the distribution of normal honest keys where $y_K = \tilde{y}_K y_1$ and $y'_K = \tilde{y}'_K y_2$, which are uniformly distributed. The shares $\lambda_K = \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}'_K y_2$ are then shares of $\alpha = \tilde{\alpha} y_1$ and $\alpha' = \tilde{\alpha}' y_2$ respectively, which are appropriately matching in the public parameters. The simulator responds to all honest key requests (after the ℓth request) in this way.

The simulator additionally chooses and fixes $u \leftarrow \mathbb{Z}_p$. For key requests up to the $(\ell - 1)$th request, the simulator generates shares λ''_K of u. It then creates a semi-functional key of type 4R by adding $\lambda''_K \vec{\mathbf{b}}_7$ to each term in the secret key set (it can create these using its knowledge

of $\vec{b}_7 = \vec{d}_7$):

$$SK_\Lambda = \{\tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 A_K \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j}\right) + \tilde{y}_K y_1 B_K \vec{\mathbf{b}}_3$$
$$+\tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 A_K \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j}\right) + \tilde{y}'_K y_2 B_K \vec{\mathbf{b}}_6$$
$$+\lambda''_K \vec{\mathbf{b}}_7$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

On the ℓth key request, for each K attribute label in policy Λ, the simulator first creates sharings $\tilde{\lambda}_K$, $\tilde{\lambda}'_K$ of $\tilde{\alpha}, \tilde{\alpha}'$ respectively. It then creates a sharing $\tilde{\lambda}''_K$ of 0 under Λ, then generates $\tilde{y}_K, \tilde{y}'_K, \tilde{y}''_K \leftarrow \mathbb{Z}_p$ and outputs:

$$SK_\Lambda = \{ \left(g^{\vec{d}_1} g^{\vec{d}_4} T^{\vec{d}_7}\right)^{\tilde{\lambda}''_K}$$
$$\left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{\lambda}_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{\lambda}'_K}$$
$$\left(g^{\vec{d}_1} g^{\vec{d}_4} T^{\vec{d}_7}\right)^{\tilde{y}''_K A_K} \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{y}_K A_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{y}'_K A_K}$$
$$\prod_{j \in K} \left[\left(g^{\vec{d}_{2,j}} g^{\vec{d}_{5,j}} T^{\vec{d}_8}\right)^{\tilde{y}''_K} \left((g^{y_1})^{\vec{d}_{2,j}} (g^{y_1 c_1})^{\vec{d}_{8,j}}\right)^{\tilde{y}_K} \left((g^{y_2})^{\vec{d}_{5,j}} (g^{y_2 c_2})^{\vec{d}_{8,j}}\right)^{\tilde{y}'_K}\right]$$
$$\left(g^{\vec{d}_3} g^{\vec{d}_6} T^{\vec{d}_9}\right)^{\tilde{y}''_K B_K} \left((g^{y_1})^{\vec{d}_3} (g^{y_1 c_1})^{\vec{d}_9}\right)^{\tilde{y}_K B_K} \left((g^{y_2})^{\vec{d}_6} (g^{y_2 c_2})^{\vec{d}_9}\right)^{\tilde{y}'_K B_K}$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$\{(\tilde{\lambda}''_K + \tilde{\lambda}_K y_1)\vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j}\right) + (\tilde{y}_K y_1 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_3$$
$$+(\tilde{\lambda}''_K + \tilde{\lambda}'_K y_2)\vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j}\right) + (\tilde{y}'_K y_2 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_{6,j}$$
$$+r\tilde{\lambda}''_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r A_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j}\right) + \tilde{y}''_K r B_K \vec{\mathbf{b}}_9$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

Since the $\tilde{\lambda}''_K$ are a sharing of zero and the $\tilde{\lambda}_K, \tilde{\lambda}'_K$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_K = \tilde{\lambda}''_K + \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}''_K + \tilde{\lambda}'_K y_2$ are sharings of $\alpha = \tilde{\alpha} y_1$ and $\alpha' = \tilde{\alpha}' y_2$ respectively, which are appropriately matching in the public parameters.

Notice that for $T = g^{c_1+c_2+r}$, if $r = 0$, then this ℓth key is distributed exactly like an honest key where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, and $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$ which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of $\text{Game}_{(\ell-1)_8}$.

If r is a uniform randomly chosen element of $\mathbb{Z}_p$, the ℓth key is distributed exactly like a semi-functional key of type 0Z where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$, and $y''_K = \tilde{y}''_K r$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$. Since the $\tilde{\lambda}''_K$ are a sharing of zero, the

shares $\lambda''_K = r\tilde{\lambda}''_K$ are also a sharing of zero, as is appropriate for a semi-functional key of type 0Z. So, if r is a uniform randomly chosen element of $\mathbb{Z}_p$, the simulator's behavior is exactly that of Game_{ℓ_0} .

Therefore, any adversary with non-negligible difference in advantage between $\text{Game}_{(\ell-1)_8}$ and Game_{ℓ_0} could be used to achieve the same non-negligible advantage in deciding the 2-Linear Problem. By assumption this is not possible, so such an adversary cannot exist.

□

Lemma 30. *No polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_0} and Game_{ℓ_1} for any ℓ from 1 to Q.*

Proof. The distributions of Game_{ℓ_0} and Game_{ℓ_1} are actually identical, due to the way the sets $(\mathbb{B}, \mathbb{B}^*)$ are chosen. Namely, since not all of the vectors are used in the public parameters, there is a invertible linear function that can be used to produce an identical distribution of sets, but decorrelates the a_j, b_j in the semi-functional space.

Consider the simulation of Game_{ℓ_0} which first uses the normal $Dual(\mathbb{Z}_p^{3k+6})$ procedure to generate orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

Consider two scenarios: in the first, the sets $(\mathbb{B}, \mathbb{B}^*)$ used in the scheme's simulation are defined via the identity transformation on $(\mathbb{D}, \mathbb{D}^*)$. That is, each $\vec{b} = \vec{d}$ and $\vec{b}^* = \vec{d}^*$. A simulator can clearly simulate Game_{ℓ_0} exactly by following the procedures using these $(\mathbb{D}, \mathbb{D}^*) = (\mathbb{B}, \mathbb{B}^*)$ to create public parameters, semi-functional ciphertexts of type 0 and keys which are semi-functional of type 4R up to the ℓth key, which is made semi-functional of type 0Z, after which all keys are made honestly.

In the second scenario, the sets $(\mathbb{B}, \mathbb{B}^*)$ are implicitly defined as follows:

$$\begin{array}{lll}
\vec{b}^*_{1,j} = \vec{d}^*_{1,j} & \vec{b}^*_2 = \vec{d}^*_2 & \vec{b}^*_{3,j} = \vec{d}^*_{3,j} \\
\vec{b}^*_{4,j} = \vec{d}^*_{4,j} & \vec{b}^*_5 = \vec{d}^*_5 & \vec{b}^*_{6,j} = \vec{d}^*_{6,j} \\
\vec{b}^*_{7,j} = \vec{d}^*_{7,j} + \tilde{a}_j \vec{d}^*_8 & \vec{b}^*_8 = \vec{d}^*_8 & \vec{b}^*_{9,j} = \vec{d}^*_{9,j} + \tilde{b}_j \vec{d}^*_8
\end{array}$$

$$\begin{array}{lll}
\vec{b}_1 = \vec{d}_1 & \vec{b}_{2,j} = \vec{d}_{2,j} & \vec{b}_3 = \vec{d}_3 \\
\vec{b}_4 = \vec{d}_4 & \vec{b}_{5,j} = \vec{d}_{5,j} & \vec{b}_6 = \vec{d}_6 \\
\vec{b}_7 = \vec{d}_7 & \vec{b}_{8,j} = \vec{d}_{8,j} - \tilde{a}_j \vec{d}_7 - \tilde{b}_j \vec{d}_9 & \vec{b}_9 = \vec{d}_9
\end{array}$$

(The distribution of all sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to the set created using the identity transformation since there is a one to one mapping between both sets and our orthogonality constraints are satisfied by both sets).

So, the $(\mathbb{B}, \mathbb{B}^*)$ formed using this alternative transformation have the same distribution as the sets straight from the $Dual(\mathbb{Z}_p^{3k+6})$ procedure. Furthermore, since this transformation only causes differences in the $\vec{d}^*_{7,j}, \vec{d}^*_{9,j}$, and $\vec{b}_{d,j}$ vectors, using this set of $(\mathbb{D}, \mathbb{D}^*)$ in the same simulation described above will result in the same of public parameters, the first $(\ell - 1)$ keys of type 4R, and all honest keys (since no $\vec{d}^*_{7,j}, \vec{d}^*_{9,j}$, and $\vec{b}_{d,j}$ are used in these objects). The only difference is seen in the ℓth key and the challenge ciphertext, where the transformation causes the semi-functional subset-sums to lose their correlation with the sums in the normal space.

The ℓth key, which is now semi-functional of type 1Z:

$$\{\lambda_K \vec{\mathbf{d}}_1 + y_K A_K \vec{\mathbf{d}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{d}}_{2,j} \right) + y_K B_K \vec{\mathbf{d}}_3$$
$$+\lambda'_K \vec{\mathbf{d}}_4 + y'_K A_K \vec{\mathbf{d}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{d}}_{5,j} \right) + y'_K B_K \vec{\mathbf{d}}_6$$
$$+\lambda''_K \vec{\mathbf{d}}_7 + y''_K A_K \vec{\mathbf{d}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{d}}_{8,j} \right) + y''_K B_K \vec{\mathbf{d}}_9$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

$$= \{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3$$
$$+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6$$
$$+\lambda''_K \vec{\mathbf{b}}_7 + y''_K A_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} [\vec{\mathbf{b}}_{8,j} + \tilde{a}_j \vec{\mathbf{b}}_7 + \tilde{b}_j \vec{\mathbf{b}}_9] \right) + y''_K B_K \vec{\mathbf{b}}_9$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

$$= \{\lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + y_K B_K \vec{\mathbf{b}}_3$$
$$+\lambda'_K \vec{\mathbf{b}}_4 + y'_K A_K \vec{\mathbf{b}}_4 + y'_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + y'_K B_K \vec{\mathbf{b}}_6$$
$$+\lambda''_K \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} (a_j + \tilde{a}_j) \right) \vec{\mathbf{b}}_7 + y''_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + y''_K \left(\sum_{j \in K} (b_j + \tilde{b}_j) \right) \vec{\mathbf{b}}_9$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

where here, the decoupled $a''_j = a_j + \tilde{a}_j$ and $b''_j = b_j + \tilde{b}_j$.

The challenge ciphertext, which is now semi-functional of type 1:

$$Me(g,g)^{\alpha s + \alpha' s'}, \{s \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{1,j} \right) - s A_K \vec{\mathbf{d}}^*_2 - t_K B_K \vec{\mathbf{d}}^*_2 + t_K \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{3,j} \right)$$
$$+s' \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{4,j} \right) - s' A_K \vec{\mathbf{d}}^*_5 - t'_K B_K \vec{\mathbf{d}}^*_5 + t'_K \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{6,j} \right)$$
$$+s'' \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{7,j} \right) - s'' A_K \vec{\mathbf{d}}^*_8 - t''_K B_K \vec{\mathbf{d}}^*_8 + t''_K \left(\sum_{j \in K} \vec{\mathbf{d}}^*_{9,j} \right)$$
$$:(\forall K \in S)\}$$

$$Me(g,g)^{\alpha s+\alpha' s'}, \{ s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}[\vec{\mathbf{b}}^*_{7,j} - \tilde{a}_j\vec{\mathbf{b}}^*_8]\right) - s''A_K\vec{\mathbf{b}}^*_8 - t''_K B_K\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}[\vec{\mathbf{b}}^*_{9,j} - \tilde{b}_j\vec{\mathbf{b}}^*_8]\right)$$
$$:(\forall K \in S)\}$$

$$= Me(g,g)^{\alpha s+\alpha' s'}, \{ s\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{1,j}\right) - sA_K\vec{\mathbf{b}}^*_2 - t_K B_K\vec{\mathbf{b}}^*_2 + t_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{3,j}\right)$$
$$+s'\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{4,j}\right) - s'A_K\vec{\mathbf{b}}^*_5 - t'_K B_K\vec{\mathbf{b}}^*_5 + t'_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{6,j}\right)$$
$$+s''\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{7,j}\right) - s''\left(\sum_{j\in K}(a_j+\tilde{a}_j)\right)\vec{\mathbf{b}}^*_8 - t''_K\left(\sum_{j\in K}(b_j+\tilde{b}_j)\right)\vec{\mathbf{b}}^*_8 + t''_K\left(\sum_{j\in K}\vec{\mathbf{b}}^*_{9,j}\right)$$
$$:(\forall K \in S)\}$$

where again, the decoupled $a''_j = a_j + \tilde{a}_j$ and $b''_j = b_j + \tilde{b}_j$ and appropriately match with the elements used in the ℓth key. So the game simulated in this scenario is exactly that of Game_{ℓ_1}.

Since the only difference between these two scenarios is the definition of the sets $(\mathbb{B}, \mathbb{B}^*)$, and we showed that both definitions result in the same distribution of sets upon generation, then we have shown that Game_{ℓ_0} and Game_{ℓ_1} are actually identical, and therefore no polynomial time attacker can achieve a non-negligible difference in advantage between them. □

We move on to the next lemma, in which the newly uncorrelated $\tilde{B}_K$ in the semi-functional space lose their structure and become independently chosen fixed random values $\tilde{b}_K \in \mathbb{Z}_p$ via the use of Lemma 12.

Lemma 31. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_1} and Game_{ℓ_2} for any ℓ from 1 to Q.*

Proof. This can be proved using the same idea as in Lemma 13 of the composite order proof. From Lemma 12, we have that the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$ are computationally indistinguishable under the 2-Linear computational hardness assumption if $k = O(\lg poly(\lambda))$. A polynomial time attacker $\mathcal{A}$ able to achieve a non-negligible difference in advantage between Game_{ℓ_1} and Game_{ℓ_2} for some ℓ from 1 to Q can be used to achieve non-negligible difference in advantage deciding between the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$, violating the 2-Linear Assumption.

Specifically, just as in Lemma 13, setting k to be the same value that defines our attribute universe $\mathcal{U}$ (so $k = O(\lg poly(\lambda))$), the

$$g^{\tilde{y}_K}, g^{\tilde{y}_K r_K}$$

obtained from the challenge 2-Linear Problem instance where $|G| = p$ (where either $r_K = \sum_{j\in K} b''_j$ and b''_j are uniform random elements of $\mathbb{Z}_p$ or each r_K is a uniform random element of $\mathbb{Z}_p$) can

be used to replace the b''_j (so depending on whether the r_K are structured or independently random, the resulting element is either $\tilde{B}_K$ or $\tilde{b}_K$). All other values are chosen by the simulator. Constructing the appropriate public parameters, ciphertext, and keys as in Lemma 13 (but modified to match the new form of the prime-order construction), allows the $\tilde{B}_K$ to lose its structure and become independent $\tilde{b}_K$, transitioning from Game_{ℓ_1} to Game_{ℓ_2}. So, an adversary able to achieve a non-negligible difference in advantage δ between Game_{ℓ_1} and Game_{ℓ_2} for some ℓ from 1 to Q can be used to achieve the same non-negligible difference in advantage deciding between the distributions $\mathcal{D}_1(k)$ and $\mathcal{D}_2(k)$, violating the 2-Linear Assumption. □

The next step in the hybrid proof sees the uncorrelated $\tilde{A}_K$ in the semi-functional space lose their structure and become independently chosen fixed random values $\tilde{a}_K \in \mathbb{Z}_p$ again via the use of Lemma 12.

Lemma 32. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_2} and Game_{ℓ_3} for any ℓ from 1 to Q .*

Proof. This can be proved using the same idea as in Lemma 14 of the composite order proof, in the same way that the previous lemma used the idea of Lemma 13. Specifically, the challenge instance is used to replace the $\tilde{A}_K$, and all other elements are chosen by the simulator appropriately. This allows us to transition from Game_{ℓ_2} to Game_{ℓ_3}. □

Now that the coefficients in the semi-functional space have become decorrelated and lost their subset structure, we can apply a nearly identical argument to that of Lemma 15 of the composite order proof to move from the λ''_K in the semi-functional space sharing zero to sharing a random element of $\mathbb{Z}_p$.

Lemma 33. *No polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_3} and Game_{ℓ_4} for any ℓ from 1 to Q.*

Proof. The argument of Lemma 15 proving that both games have the same distribution holds here. The only difference is that the in the information theoretic argument for attributes not in the challenge ciphertext, the $\tilde{a}_K$ that are masking the value of the share λ''_K are coefficients of $g^{\vec{b}_7}$ (instead of just exponents of g). Regardless, the argument holds. □

We get the next two lemmas for free, as in the composite order proof:

Lemma 34. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_4} and Game_{ℓ_5} for any ℓ from 1 to Q.*

Lemma 35. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_5} and Game_{ℓ_6} for any ℓ from 1 to Q .*

Proof. Just as in the composite order proof, we can just perform the same hybrid as Lemma 31 and Lemma 32 in reverse, with the only difference being that the λ''_K are generated as shares of a random element of $\mathbb{Z}_p$ instead of zero. This doesn't affect the argument since the simulator is able to generate the λ''_K both ways. □

Lemma 36. *No polynomial time attacker can achieve a non-negligible difference in advantage between Game_{ℓ_6} and Game_{ℓ_7} for any ℓ from 1 to Q.*

Proof. This information-theoretic argument is the same as Lemma 30 in reverse, with the only difference being that the λ''_K are generated as shares of a random element of $\mathbb{Z}_p$ instead of zero. This doesn't affect the argument since again the simulator is able to generate the λ''_K both ways. □

The last step in the hybrid is to change the ℓth key from semi-functional of type 0R to 4R (losing its $\tilde{A}_K, \tilde{B}_K$):

Lemma 37. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_7}$ and $Game_{\ell_8}$ for any ℓ from 1 to Q.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between Game_{ℓ_7} and Game_{ℓ_8} for some ℓ in $\{1, ..., Q\}$, then we could use $\mathcal{A}$ to achieve non-negligible advantage in the 2-Linear Problem as follows:

Given $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ and $T = g^{c_1+c_2+r} \in G$, where either $r = 0$ or is a uniform random element of $\mathbb{Z}_p$, consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g, choosing $\tilde{\alpha}, \tilde{\alpha}', a_j, b_j \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

The simulator then implicitly defines the sets $(\mathbb{B}, \mathbb{B}^*)$ as:

$$\begin{array}{lll}
\vec{b}^*_{1,j} = \vec{d}^*_{1,j} & \vec{b}^*_2 = \vec{d}^*_2 & \vec{b}^*_{3,j} = \vec{d}^*_{3,j} \\
\vec{b}^*_{4,j} = \vec{d}^*_{4,j} & \vec{b}^*_5 = \vec{d}^*_5 & \vec{b}^*_{6,j} = \vec{d}^*_{6,j} \\
\vec{b}^*_{7,j} = \vec{d}^*_{7,j} - c_1\vec{d}^*_{1,j} - c_2\vec{d}^*_{4,j} & \vec{b}^*_8 = \vec{d}^*_8 - c_1\vec{d}^*_2 - c_2\vec{d}^*_5 & \vec{b}^*_{9,j} = \vec{d}^*_{9,j} - c_1\vec{d}^*_{3,j} - c_2\vec{d}^*_{6,j}
\end{array}$$

$$\begin{array}{lll}
\vec{b}_1 = \vec{d}_1 + c_1\vec{d}_7 & \vec{b}_{2,j} = \vec{d}_{2,j} + c_1\vec{d}_{8,j} & \vec{b}_3 = \vec{d}_3 + c_1\vec{d}_9 \\
\vec{b}_4 = \vec{d}_4 + c_2\vec{d}_7 & \vec{b}_{5,j} = \vec{d}_{5,j} + c_2\vec{d}_{8,j} & \vec{b}_6 = \vec{d}_6 + c_2\vec{d}_9 \\
\vec{b}_7 = \vec{d}_7 & \vec{b}_{8,j} = \vec{d}_{8,j} & \vec{b}_9 = \vec{d}_9
\end{array}$$

(The distribution of all sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to that produced by $Dual(\mathbb{Z}_p^{3k+6})$, since there is a one to one mapping between any sets produced this way and the sets produced by the $Dual(\mathbb{Z}_p^{3k+6})$ procedure.)

The public parameters are constructed as:

$$\begin{array}{l}
p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\} \\
\{a_j\vec{\mathbf{b}}^*_2, b_j\vec{\mathbf{b}}^*_2, a_j\vec{\mathbf{b}}^*_5, b_j\vec{\mathbf{b}}^*_5 : j \in [k]\}
\end{array}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be easily made by the simulator from the $\vec{d}^*$ vectors)

The simulator then gives the public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S when it is requested, first, $s'' \leftarrow \mathbb{Z}_p$ is chosen. Then, for each $K \in S$, $\tilde{t}_K, \tilde{t}'_K, \tilde{t}''_K \leftarrow \mathbb{Z}_p$ are chosen. The following ciphertext is then constructed and provided:

$$\begin{array}{l}
Me(g, g^{y_1 c_1})^{\tilde{\alpha}s''} e(g, g^{y_2 c_2})^{\tilde{\alpha}'s''}, \\
\{ \prod_{j\in K} (g^{\vec{d}^*_{7,j}})^{s''} \\
(g^{\vec{d}^*_8})^{-s''A_K} \\
(g^{\vec{d}^*_2})^{-\tilde{t}_K B_K} (g^{\vec{d}^*_5})^{-\tilde{t}'_K B_K} (g^{\vec{d}^*_8})^{-t''_K B_K} \\
\prod_{j\in K} \left[(g^{\vec{d}^*_{3,j}})^{\tilde{t}_K} (g^{\vec{d}^*_{6,j}})^{\tilde{t}'_K} (g^{\vec{d}^*_{9,j}})^{t''_K} \right] \\
:(\forall K \in S)\}
\end{array}$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
& Me(g,g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'}, \\
& \{c_1 s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{1,j}\right) - c_1 s'' A_K \vec{\mathbf{b}}^*_2 - (\tilde{t}_K + c_1 t''_K) B_K \vec{\mathbf{b}}^*_2 + (\tilde{t}_K + c_1 t''_K) \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{3,j}\right) \\
& + c_2 s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{4,j}\right) - c_2 s'' A_K \vec{\mathbf{b}}^*_5 - (\tilde{t}'_K + c_2 t''_K) B_K \vec{\mathbf{b}}^*_5 + (\tilde{t}'_K + c_2 t''_K) \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{6,j}\right) \\
& + s'' \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{7,j}\right) - s'' A_K \vec{\mathbf{b}}^*_8 - t''_K B_K \vec{\mathbf{b}}^*_8 + t''_K \left(\sum_{j\in K} \vec{\mathbf{b}}^*_{9,j}\right) \\
& : (\forall K \in S)\}
\end{aligned}
$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$, $t_K = \tilde{t}_K + c_1 t''_K$, and $t'_K = \tilde{t}'_K + c_2 t''_K$ are all independently and uniformly randomly distributed in $\mathbb{Z}_p$.

To respond to key requests for policies Λ, the simulator first computes an honest key by creating $\tilde{\lambda}_K, \tilde{\lambda}'_K$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ under policy Λ respectively.

$$
\begin{aligned}
SK_\Lambda = \{ & \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{\lambda}_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{\lambda}'_K} \\
& \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{y}_K A_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{y}'_K A_K} \\
& \prod_{j\in K} \left[\left((g^{y_1})^{\vec{d}_{2,j}} (g^{y_1 c_1})^{\vec{d}_{8,j}}\right)^{\tilde{y}_K} \left((g^{y_2})^{\vec{d}_{5,j}} (g^{y_2 c_2})^{\vec{d}_{8,j}}\right)^{\tilde{y}'_K}\right] \\
& \left((g^{y_1})^{\vec{d}_3} (g^{y_1 c_1})^{\vec{d}_9}\right)^{\tilde{y}_K B_K} \left((g^{y_2})^{\vec{d}_5} (g^{y_2 c_2})^{\vec{d}_9}\right)^{\tilde{y}'_K B_K} \\
& : (\forall K \text{ labels} \in \Lambda)\}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
SK_\Lambda = \{ & \tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 A_K \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 \left(\sum_{j\in K} \vec{\mathbf{b}}_{2,j}\right) + \tilde{y}_K y_1 B_K \vec{\mathbf{b}}_3 \\
& + \tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 A_K \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 \left(\sum_{j\in K} \vec{\mathbf{b}}_{5,j}\right) + \tilde{y}'_K y_2 B_K \vec{\mathbf{b}}_6 \\
& : (\forall K \text{ labels} \in \Lambda)\}
\end{aligned}
$$

The distribution of keys constructed this way is identical to the distribution of normal honest keys where $y_K = \tilde{y}_K y_1$ and $y'_K = \tilde{y}'_K y_2$, which are uniformly distributed. The shares $\lambda_K = \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}'_K y_2$ are then shares of $\alpha = \tilde{\alpha} y_1$ and $\alpha' = \tilde{\alpha}' y_2$ respectively, which are appropriately matching in the public parameters. The simulator responds to all honest key requests (after the ℓth request) in this way.

The simulator additionally chooses and fixes $u \leftarrow \mathbb{Z}_p$. For key requests up to the $(\ell - 1)$th request, the simulator generates shares λ''_K of u. It then creates a semi-functional key of type 4R by adding $\lambda''_K \vec{\mathbf{b}}_7$ to each term in the secret key set (it can create these using its knowledge

of $\vec{b}_7 = \vec{d}_7$):

$$SK_\Lambda = \{\tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 A_K \vec{\mathbf{b}}_1 + \tilde{y}_K y_1 \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j}\right) + \tilde{y}_K y_1 B_K \vec{\mathbf{b}}_3$$
$$+\tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 A_K \vec{\mathbf{b}}_4 + \tilde{y}'_K y_2 \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j}\right) + \tilde{y}'_K y_2 B_K \vec{\mathbf{b}}_6$$
$$+\lambda''_K \vec{\mathbf{b}}_7$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

On the ℓth key request, for each K attribute label in Λ, the simulator first creates sharings $\tilde{\lambda}_K$, $\tilde{\lambda}'_K$ of $\tilde{\alpha}, \tilde{\alpha}'$ respectively. It then creates a sharing λ''_K of u under Λ, then generates uniformly random $\tilde{y}_K, \tilde{y}'_K, \tilde{y}''_K \in \mathbb{Z}_p$ and outputs:

$$SK_\Lambda = \{ \left(g^{\vec{d}_7}\right)^{\lambda''_K}$$
$$\left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{\lambda}_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{\lambda}'_K}$$
$$\left(g^{\vec{d}_1} g^{\vec{d}_4} T^{\vec{d}_7}\right)^{\tilde{y}''_K A_K} \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7}\right)^{\tilde{y}_K A_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7}\right)^{\tilde{y}'_K A_K}$$
$$\prod_{j \in K} \left[\left(g^{\vec{d}_{2,j}} g^{\vec{d}_{5,j}} T^{\vec{d}_8}\right)^{\tilde{y}''_K} \left((g^{y_1})^{\vec{d}_{2,j}} (g^{y_1 c_1})^{\vec{d}_{8,j}}\right)^{\tilde{y}_K} \left((g^{y_2})^{\vec{d}_{5,j}} (g^{y_2 c_2})^{\vec{d}_{8,j}}\right)^{\tilde{y}'_K}\right]$$
$$\left(g^{\vec{d}_3} g^{\vec{d}_6} T^{\vec{d}_9}\right)^{\tilde{y}''_K B_K} \left((g^{y_1})^{\vec{d}_3} (g^{y_1 c_1})^{\vec{d}_9}\right)^{\tilde{y}_K B_K} \left((g^{y_2})^{\vec{d}_6} (g^{y_2 c_2})^{\vec{d}_9}\right)^{\tilde{y}'_K B_K}$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$\{\tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j}\right) + (\tilde{y}_K y_1 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_3$$
$$+\tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j}\right) + (\tilde{y}'_K y_2 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_{6,j}$$
$$+\lambda''_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r A_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j}\right) + \tilde{y}''_K r B_K \vec{\mathbf{b}}_9$$
$$:(\forall K \text{ labels} \in \Lambda)\}$$

Since the $\tilde{\lambda}_K, \tilde{\lambda}'_K$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_K = \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}'_K y_2$ are sharings of $\alpha = \tilde{\alpha} y_1$ and $\alpha' = \tilde{\alpha}' y_2$ respectively, which are appropriately matching in the public parameters.

Notice that for $T = g^{c_1 + c_2 + r}$, if $r = 0$, then this ℓth key is distributed exactly like a semi-functional key of type 4R where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, and $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{ℓ_8}.

If r is a uniform randomly chosen element of $\mathbb{Z}_p$, the ℓth key is distributed exactly like a semi-functional key of type 0R where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$, and $y''_K = \tilde{y}''_K r$, which

are all distributed as uniformly random elements of $\mathbb{Z}_p$. So, if r is a uniform randomly chosen element of $\mathbb{Z}_p$, the simulator's behavior is exactly that of Game_{ℓ_7} .

Therefore, any adversary with non-negligible difference in advantage between Game_{ℓ_7} and Game_{ℓ_8} could be used to achieve the same non-negligible advantage in deciding the 2-Linear Problem. By assumption this is not possible, so such an adversary cannot exist. □

This set of hybrids takes us to Game_{Q_8}: where the semi-functional ciphertext is of type 0 and all keys are semi-functional of type 4R. We take two more steps to get to our final game where the distribution of the ciphertext is independent of the message - a game in which the adversary cannot have any advantage:

Lemma 38. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between* Game_{Q_8} *and* $\text{Game}_{penultimate}$.

Proof. If an algorithm $\mathcal{A}$ is able to achieve a non-negligible difference in advantage between Game_{Q_8} and $\text{Game}_{penultimate}$, then we could use $\mathcal{A}$ to break the 2-Linear Assumption as follows:

Given $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ and $T = g^{c_1+c_2+r} \in G$, where either $r = 0$ or is a uniform random element of $\mathbb{Z}_p$, consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g, choosing $\tilde{\alpha}, \tilde{\alpha}', a_j, b_j \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

The simulator then implicitly defines the sets $(\mathbb{B}, \mathbb{B}^*)$ as:

$$\begin{array}{lll}
\vec{b}^*_{1,j} = \vec{d}^*_{1,j} & \vec{b}^*_2 = \vec{d}^*_2 & \vec{b}^*_{3,j} = \vec{d}^*_{3,j} \\
\vec{b}^*_{4,j} = \vec{d}^*_{4,j} & \vec{b}^*_5 = \vec{d}^*_5 & \vec{b}^*_{6,j} = \vec{d}^*_{6,j} \\
\vec{b}^*_{7,j} = \vec{d}^*_{7,j} - c_1\vec{d}^*_{1,j} - c_2\vec{d}^*_{4,j} & \vec{b}^*_8 = \vec{d}^*_8 - c_1\vec{d}^*_2 - c_2\vec{d}^*_5 & \vec{b}^*_{9,j} = \vec{d}^*_{9,j} - c_1\vec{d}^*_{3,j} - c_2\vec{d}^*_{6,j}
\end{array}$$

$$\begin{array}{lll}
\vec{b}_1 = \vec{d}_1 + c_1\vec{d}_7 & \vec{b}_{2,j} = \vec{d}_{2,j} + c_1\vec{d}_{8,j} & \vec{b}_3 = \vec{d}_3 + c_1\vec{d}_9 \\
\vec{b}_4 = \vec{d}_4 + c_2\vec{d}_7 & \vec{b}_{5,j} = \vec{d}_{5,j} + c_2\vec{d}_{8,j} & \vec{b}_6 = \vec{d}_6 + c_2\vec{d}_9 \\
\vec{b}_7 = \vec{d}_7 & \vec{b}_{8,j} = \vec{d}_{8,j} & \vec{b}_9 = \vec{d}_9
\end{array}$$

(The distribution of the sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to that produced by $Dual(\mathbb{Z}_p^{3k+6})$, since there is a one to one mapping between any sets produced this way and the sets produced by the $Dual(\mathbb{Z}_p^{3k+6})$ procedure.)

The public parameters are constructed as:

$$\begin{array}{l}
p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\} \\
\{a_j\vec{\mathbf{b}}^*_2, b_j\vec{\mathbf{b}}^*_2, a_j\vec{\mathbf{b}}^*_5, b_j\vec{\mathbf{b}}^*_5 : j \in [k]\}
\end{array}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be easily made by the simulator from the $\vec{d}^*$ vectors)

The simulator then gives the public parameters to $\mathcal{A}$.

To return the challenge ciphertext for a set of attributes S when it is requested, first, $s'' \leftarrow \mathbb{Z}_p$ is chosen. Then, for each $K \in S$, $\tilde{t}_K, \tilde{t}'_K, \tilde{t}''_K \leftarrow \mathbb{Z}_p$ are chosen. The following ciphertext is then constructed and provided:

$$
\begin{aligned}
&Me(g, g^{y_1 c_1})^{\tilde{\alpha} s''} e(g, g^{y_2 c_2})^{\tilde{\alpha}' s''}, \\
&\{ \prod_{j \in K} (g^{\vec{d}^*_{7,j}})^{s''} \\
&(g^{\vec{d}^*_8})^{-s'' A_K} \\
&(g^{\vec{d}^*_2})^{-\tilde{t}_K B_K} (g^{\vec{d}^*_5})^{-\tilde{t}'_K B_K} (g^{\vec{d}^*_8})^{-t''_K B_K} \\
&\prod_{j \in K} \left[(g^{\vec{d}^*_{3,j}})^{\tilde{t}_K} (g^{\vec{d}^*_{6,j}})^{\tilde{t}'_K} (g^{\vec{d}^*_{9,j}})^{t''_K} \right] \\
&: (\forall K \in S) \}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
&Me(g,g)^{y_1 \tilde{\alpha} s + y_2 \tilde{\alpha}' s'}, \\
&\{ c_1 s'' \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{1,j} \right) - c_1 s'' A_K \vec{\mathbf{b}}^*_2 - (\tilde{t}_K + c_1 t''_K) B_K \vec{\mathbf{b}}^*_2 + (\tilde{t}_K + c_1 t''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{3,j} \right) \\
&+ c_2 s'' \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{4,j} \right) - c_2 s'' A_K \vec{\mathbf{b}}^*_5 - (\tilde{t}'_K + c_2 t''_K) B_K \vec{\mathbf{b}}^*_5 + (\tilde{t}'_K + c_2 t''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{6,j} \right) \\
&+ s'' \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{7,j} \right) - s'' A_K \vec{\mathbf{b}}^*_8 - t''_K B_K \vec{\mathbf{b}}^*_8 + t''_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{9,j} \right) \\
&: (\forall K \in S) \}
\end{aligned}
$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$, $t_K = \tilde{t}_K + c_1 t''_K$, and $t'_K = \tilde{t}'_K + c_2 t''_K$ are all independently and uniformly randomly distributed in $\mathbb{Z}_p$.

To respond to key requests for policies Λ, the simulator first creates sharings $\tilde{\lambda}_K$, $\tilde{\lambda}'_K$, and λ''_K of $\tilde{\alpha}, \tilde{\alpha}'$, and u respectively (where u is a random element of $\mathbb{Z}_p$ which is fixed the first time it is created). It then generates uniformly random $\tilde{y}_K, \tilde{y}'_K, \tilde{y}''_K \in \mathbb{Z}_p$ and outputs:

$$
\begin{aligned}
SK_\Lambda = \{ &\left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7} \right)^{\tilde{\lambda}_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7} \right)^{\tilde{\lambda}'_K} \\
&\left(g^{\vec{d}_1} g^{\vec{d}_4} T^{\vec{d}_7} \right)^{\tilde{y}''_K A_K} \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_7} \right)^{\tilde{y}_K A_K} \left((g^{y_2})^{\vec{d}_4} (g^{y_2 c_2})^{\vec{d}_7} \right)^{\tilde{y}'_K A_K} \\
&\prod_{j \in K} \left[\left(g^{\vec{d}_{2,j}} g^{\vec{d}_{5,j}} T^{\vec{d}_8} \right)^{\tilde{y}''_K} \left((g^{y_1})^{\vec{d}_{2,j}} (g^{y_1 c_1})^{\vec{d}_{8,j}} \right)^{\tilde{y}_K} \left((g^{y_2})^{\vec{d}_{5,j}} (g^{y_2 c_2})^{\vec{d}_{8,j}} \right)^{\tilde{y}'_K} \right] \\
&\left(g^{\vec{d}_3} g^{\vec{d}_6} T^{\vec{d}_9} \right)^{\tilde{y}''_K B_K} \left((g^{y_1})^{\vec{d}_3} (g^{y_1 c_1})^{\vec{d}_9} \right)^{\tilde{y}_K B_K} \left((g^{y_2})^{\vec{d}_6} (g^{y_2 c_2})^{\vec{d}_9} \right)^{\tilde{y}'_K B_K} \\
&g^{\lambda''_K \vec{d}_7} \\
&: (\forall K \text{ labels} \in \Lambda) \}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$\{\tilde{\lambda}_K y_1 \vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_1 + (\tilde{y}_K y_1 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + (\tilde{y}_K y_1 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_3$$

$$+\tilde{\lambda}'_K y_2 \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) A_K \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_2 + \tilde{y}''_K) \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + (\tilde{y}'_K y_2 + \tilde{y}''_K) B_K \vec{\mathbf{b}}_{6,j}$$

$$+\lambda''_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r A_K \vec{\mathbf{b}}_7 + \tilde{y}''_K r \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + \tilde{y}''_K r B_K \vec{\mathbf{b}}_9$$

$$:(\forall K \text{ labels} \in \Lambda)\}$$

Since the $\tilde{\lambda}_K, \tilde{\lambda}'_K$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_K = \tilde{\lambda}_K y_1$ and $\lambda'_K = \tilde{\lambda}'_K y_2$ are sharings of $\alpha = \tilde{\alpha} y_1$ and $\alpha' = \tilde{\alpha}' y_2$ respectively, which are appropriately matching in the public parameters.

Notice that for $T = g^{c_1+c_2+r}$, if $r = 0$, then each key is distributed exactly like a semi-functional key of type 4R where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, and $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$ which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{Q_8}.

If r is a uniform randomly chosen element of $\mathbb{Z}_p$, each key is distributed exactly like a semi-functional key of type 0R where $y_K = \tilde{y}_K y_1 + \tilde{y}''_K$, $y'_K = \tilde{y}'_K y_2 + \tilde{y}''_K$, and $y''_K = \tilde{y}''_K r$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of $\text{Game}_{penultimate}$.

Therefore, any adversary with non-negligible difference in advantage between Game_{Q_8} and $\text{Game}_{penultimate}$ could be used to achieve the same non-negligible advantage in deciding the 2-Linear Problem. By assumption this is not possible, so such an adversary cannot exist.

□

Lemma 39. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $\text{Game}_{penultimate}$ and Game_{final}.*

Proof. If an algorithm $\mathcal{A}$ is able to achieve a non-negligible difference in advantage between $\text{Game}_{penultimate}$ and Game_{final}, then we could use $\mathcal{A}$ to break the 2-Linear Assumption as follows:

Given $g, g^{y_1}, g^{y_2}, g^{y_1 c_1}, g^{y_2 c_2}$ and $T = g^{c_1+c_2+r} \in G$, where either $r = 0$ or is a uniform random element of $\mathbb{Z}_p$, consider the following simulator $\mathcal{B}$ in the security game:

The public parameters are formed by using the given g, choosing $\tilde{\alpha}', a_j, b_j \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3k+6})$.

The simulator then implicitly defines the sets $(\mathbb{B}, \mathbb{B}^*)$ as:

$$\vec{b}^*_{1,j} = \vec{d}^*_{1,j} \qquad \vec{b}^*_2 = \vec{d}^*_2 \qquad \vec{b}^*_{3,j} = \vec{d}^*_{3,j}$$
$$\vec{b}^*_{4,j} = y_1 \vec{d}^*_{4,j} + y_1 c_1 \vec{d}^*_{1,j} \qquad \vec{b}^*_5 = y_1 \vec{d}^*_5 + y_1 c_1 \vec{d}^*_2 \qquad \vec{b}^*_{6,j} = y_1 \vec{d}^*_{6,j} + y_1 c_1 \vec{d}^*_{3,j}$$
$$\vec{b}^*_{7,j} = y_2 \vec{d}^*_{7,j} + y_2 c_2 \vec{d}^*_{1,j} \qquad \vec{b}^*_8 = y_2 \vec{d}^*_8 + y_2 c_2 \vec{d}^*_2 \qquad \vec{b}^*_{9,j} = y_2 \vec{d}^*_{9,j} + y_2 c_2 \vec{d}^*_{3,j}$$

$$\vec{b}_1 = \vec{d}_1 - c_1 \vec{d}_4 - c_2 \vec{d}_7 \qquad \vec{b}_{2,j} = \vec{d}_{2,j} - c_1 \vec{d}_{5,j} - c_2 \vec{d}_{8,j} \qquad \vec{b}_3 = \vec{d}_3 - c_1 \vec{d}_6 - c_2 \vec{d}_9$$
$$\vec{b}_4 = y_1^{-1} \vec{d}_4 \qquad \vec{b}_{5,j} = y_1^{-1} \vec{d}_{5,j} \qquad \vec{b}_6 = y_1^{-1} \vec{d}_6$$
$$\vec{b}_7 = y_2^{-1} \vec{d}_7 \qquad \vec{b}_{8,j} = y_2^{-1} \vec{d}_{8,j} \qquad \vec{b}_9 = y_2^{-1} \vec{d}_9$$

(The distribution of the sets $(\mathbb{B}, \mathbb{B}^*)$ produced this way is identical to that produced by $Dual(\mathbb{Z}_p^{3k+6})$, since there is a one to one mapping between any sets produced this way and the sets produced by the $Dual(\mathbb{Z}_p^{3k+6})$ procedure.)

The public parameters are constructed as:

$$p, g, e(g, g^{y_1}) = e(g, g)^{y_1}, e(g, g^{y_1})^{\tilde{\alpha}'} e(g^{y_1}, g^{y_1 c_1}) = e(g, g)^{y_1 \tilde{\alpha}' + y_1^2 c_1},$$
$$\{\vec{\mathbf{b}}^*_{1,j}, \vec{\mathbf{b}}^*_{3,j}, \vec{\mathbf{b}}^*_{4,j}, \vec{\mathbf{b}}^*_{6,j} : j \in [k]\}$$
$$\{a_j \vec{\mathbf{b}}^*_2, b_j \vec{\mathbf{b}}^*_2, a_j \vec{\mathbf{b}}^*_5, b_j \vec{\mathbf{b}}^*_5 : j \in [k]\}$$

implicitly defining $\alpha = y_1$, $\alpha' = y_1 \tilde{\alpha}' + \alpha y_1 c_1$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be easily made by the simulator from combinations of the challenge terms raised to the appropriate $\vec{d}^*$ vectors)

The simulator then gives the public parameters to $\mathcal{A}$.

To respond to key requests for policies Λ, the simulator generates $\tilde{\lambda}'_K$, and $\tilde{\lambda}''_K$: sharings of $\tilde{\alpha}'$, and $\tilde{\alpha}''$ (chosen uniformly at random from $\mathbb{Z}_p$ upon the first key request and fixed for each key thereafter) respectively under policy Λ. It then generates g^{λ_K} where the λ_K are a sharing of y_1 under Λ using the procedure detailed in the final lemma of the composite proof (Lemma 19) starting with the challenge element g^{y_1}. Next, the simulator draws $y_K, \tilde{y}'_K, \tilde{y}''_K \leftarrow \mathbb{Z}_p$ and constructs:

$$\begin{aligned} SK_\Lambda = \{ & g^{\lambda_K \vec{d}_1} g^{\tilde{\lambda}'_K \vec{d}_4} g^{\tilde{\lambda}''_K \vec{d}_7} \\ & g^{y_K A_K \vec{d}_1} g^{\tilde{y}'_K A_K \vec{d}_4} g^{\tilde{y}''_K A_K \vec{d}_7} \\ & \prod_{j \in K} \left[g^{y_K \vec{d}_{2,j}} g^{\tilde{y}'_K \vec{d}_{5,j}} g^{\tilde{y}''_K \vec{d}_{8,j}} \right] \\ & g^{y_K B_K \vec{d}_3} g^{\tilde{y}'_K B_K \vec{d}_6} g^{\tilde{y}''_K B_K \vec{d}_9} \\ & : (\forall K \text{ labels} \in \Lambda)\} \end{aligned}$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$\begin{aligned} SK_\Lambda = \{ & \lambda_K \vec{\mathbf{b}}_1 + y_K A_K \vec{\mathbf{b}}_1 + y_K \left(\sum_{j \in K} \vec{\mathbf{b}}_{2,j} \right) + \tilde{y}_K y_1 B_K \vec{\mathbf{b}}_3 \\ & + (\tilde{\lambda}'_K y_1 + \lambda_K y_1 c_1) \vec{\mathbf{b}}_4 + (\tilde{y}'_K y_1 + y_K y_1 c_1) A_K \vec{\mathbf{b}}_4 \\ & + (\tilde{y}'_K y_1 + y_K y_1 c_1) \left(\sum_{j \in K} \vec{\mathbf{b}}_{5,j} \right) + (\tilde{y}'_K y_1 + y_K y_1 c_1) B_K \vec{\mathbf{b}}_6 \\ & + (\tilde{\lambda}''_K y_2 + \lambda_K y_2 c_2) \vec{\mathbf{b}}_7 + (\tilde{y}''_K y_2 + y_K y_2 c_2) A_K \vec{\mathbf{b}}_7 \\ & + (\tilde{y}''_K y_2 + y_K y_2 c_2) \left(\sum_{j \in K} \vec{\mathbf{b}}_{8,j} \right) + (\tilde{y}''_K y_2 + y_K y_2 c_2) B_K \vec{\mathbf{b}}_9 \\ & : (\forall K \text{ labels} \in \Lambda)\} \end{aligned}$$

which is distributed as a semi-functional key of type 0R (which is appropriate for both games) where $y'_K = \tilde{y}'_K y_1 + y_K y_1 c_1$, and $y''_K = \tilde{y}''_K y_2 + y_K y_2 c_2$ which are independent uniformly distributed elements of $\mathbb{Z}_p$. Also, $\lambda'_K = \tilde{\lambda}'_K y_1 + \lambda_K y_1 c_1$ and $\lambda''_K = \tilde{\lambda}''_K y_2 + \lambda_K y_2 c_2$ are shares of $\alpha' = \tilde{\alpha}' y_1 + y_1^2 c_1$ and $u = \tilde{\alpha}'' y_2 + y_1 y_2 c_2$ where u is a fixed uniform random element of $\mathbb{Z}_p$ because

the λ_K are a sharing of $\alpha = y_1$. Note also that the shared α and α' match appropriately with the public parameters.

To return the challenge ciphertext for a set of attributes S when it is requested, first, $s, \tilde{s}' \leftarrow \mathbb{Z}_p$ are chosen. Then, for each $K \in S$, $t_K, t'_K, t''_K \leftarrow \mathbb{Z}_p$ are chosen. The following ciphertext is then constructed and provided:

$$
\begin{aligned}
& M(e(g,g)^{\alpha})^{s} e(g,g)^{\tilde{\alpha}'\tilde{s}'} e(g, g^{y_1 c_1})^{\tilde{s}'}, \\
& \{ \prod_{j \in K} g^{s\vec{d}^*_{1,j}} \\
& \prod_{j \in K} \left[g^{\tilde{s}'\vec{d}^*_{4,j}} g^{\tilde{s}'\vec{d}^*_{7,j}} T^{\tilde{s}'\vec{d}^*_{1,j}} \right] \\
& g^{-sA_K\vec{d}^*_2} \\
& g^{-\tilde{s}'A_K\vec{d}^*_5} g^{-\tilde{s}'A_K\vec{d}^*_8} T^{-\tilde{s}'A_K\vec{d}^*_2} \\
& g^{-t_K B_K \vec{d}^*_2} \\
& (g^{y_1})^{-t'_K B_K \vec{d}^*_5} (g^{y_1 c_1})^{-t'_K B_K \vec{d}^*_2} \\
& (g^{y_2})^{-t''_K B_K \vec{d}^*_8} (g^{y_2 c_2})^{-t''_K B_K \vec{d}^*_2} \\
& \prod_{j \in K} g^{-t_K \vec{d}^*_{3,j}} \\
& \prod_{j \in K} \left[(g^{y_1})^{-t'_K \vec{d}^*_{6,j}} (g^{y_1 c_1})^{-t'_K \vec{d}^*_{3,j}} \right] \\
& \prod_{j \in K} \left[(g^{y_2})^{-t''_K \vec{d}^*_{9,j}} (g^{y_2 c_2})^{-t''_K \vec{d}^*_{3,j}} \right] \\
& : (\forall K \in S) \}
\end{aligned}
$$

which, using the definition of our sets $(\mathbb{B}, \mathbb{B}^*)$, is equal to:

$$
\begin{aligned}
& M e(g,g)^{\alpha s + (\tilde{\alpha}' y_1 + y_1^2 c_1)\tilde{s}' y_1^{-1}}, \\
& \{ (s + r\tilde{s}') \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{1,j} \right) - (s + r\tilde{s}') A_K \vec{\mathbf{b}}^*_2 - t_K B_K \vec{\mathbf{b}}^*_2 + t_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{3,j} \right) \\
& + \tilde{s}' y_1^{-1} \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{4,j} \right) - \tilde{s}' y_1^{-1} A_K \vec{\mathbf{b}}^*_5 - t'_K B_K \vec{\mathbf{b}}^*_5 + t'_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{6,j} \right) \\
& + y_2^{-1} \tilde{s}' \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{7,j} \right) - y_2^{-1} \tilde{s}' A_K \vec{\mathbf{b}}^*_8 - t''_K B_K \vec{\mathbf{b}}^*_8 + t''_K \left(\sum_{j \in K} \vec{\mathbf{b}}^*_{9,j} \right) \\
& : (\forall K \in S) \}
\end{aligned}
$$

Notice that for $T = g^{c_1 + c_2 + r}$, if $r = 0$, then this ciphertext is distributed exactly like a semi-functional ciphertext of type 0 where $s' = \tilde{s}' y_1^{-1}$ and $s'' = y_2^{-1}\tilde{s}'$ are both independently and uniformly randomly distributed in $\mathbb{Z}_p$. So, the simulator behaves exactly as in $\text{Game}_{penultimate}$.

However, if r is a uniform random element of $\mathbb{Z}_p$ then the ciphertext is distributed exactly like a semi-functional ciphertext of type X where $s' = \tilde{s}' y_1^{-1}$ and $s'' = y_2^{-1}\tilde{s}'$ are both independently and uniformly randomly distributed in $\mathbb{Z}_p$ and $x = s + r\tilde{s}'$ is an independent randomly distributed element of $\mathbb{Z}_p$. So, the simulator behaves exactly as in Game_{final}.

Therefore, any adversary with non-negligible difference in advantage between $\text{Game}_{penultimate}$ and Game_{final} could be used to achieve the same non-negligible advantage in deciding the 2-Linear Problem. By assumption this is not possible, so such an adversary cannot exist. □

We have now proven the following theorem

Theorem 40. *Under the 2-Linear Computational Hardness Assumption, our prime order KP-ABE scheme is fully secure.*

Proof. If the 2-Linear Computational Hardness Assumption holds, then by the previous lemmas, we have shown that the real security game is computationally indistinguishable from Game_{final}, in which the challenge ciphertext's message is information-theoretically hidden from the attacker. Hence, no attacker can achieve a non-negligible advantage in breaking the KP-ABE scheme. □