Fully-secure Key Policy ABE on Prime-Order Bilinear Groups

Luke Kowalczyk, Jiahui Liu, Kailash Meiyappan

Abstract

We present a Key-Policy ABE scheme that is fully-secure under the Decisional Linear Assumption. The construction is essentially a translation of the classic scheme of [18] from composite-order bilinear groups to the prime-order setting.

1 Introduction

Since its conception in [31], 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 [10, 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 [17, 33] 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 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 [18], using the dual system encryption methodology [32] 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 [18] and many subsequent works (e.g. [27, 21]) 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 [24], 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 ob-

jects 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 [17]. 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 [24], 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 [33] 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 [33] 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 [17] 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 We present a KP-ABE construction in the prime-order setting which supports LSSS/MSP access policies. The construction's full security is based on DLIN.

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, 30, 16]), schemes supporting multiple authorities (e.g. [6, 7, 29, 21]), and schemes supporting large attribute universes (e.g. [22, 28]). Some of the structure for randomization in our schemes is inspired by [22].

There are also recent constructions of ABE schemes in the lattice setting. The construction of [15] 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 [12] based on multilin-

ear maps. This scheme is also proven selectively secure, under a particular computational hardness assumption for multilinear groups. The very recent multilinear scheme in [13] achieves full security, relying on computational hardness assumptions in multilinear groups. The fully secure general functional encryption scheme in [34], 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 [14] 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 [20, 9, 34, 1], applied to achieve leakage resilience in [23, 19, 11], and applied directly to computational assumptions in [8].

2 Preliminaries

2.1 Composite Order Bilinear Groups

We construct our system in prime order bilinear groups. 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 (p, G, G_T, e), where p is a prime, G and G_T are cyclic groups of order p, and $e : G \times G \rightarrow G_T$ is a map with the following properties:

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

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.

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 1. *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 indexes $i \in [k]$ for a fixed k.

Definition 2. *2(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* $i = 1, \ldots, \ell$*, the* i^{th} *row of* Λ *is labeled by an attribute* i*. 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* $(\Lambda v)_i = \lambda_i$ *belongs to attribute* i*.*

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 $\{\omega_i \in \mathbb{Z}_p\}_{i \in S^*}$ such that, for any valid shares $\{\lambda_i\}$ of a secret s according to Π, we have: $\sum_{i \in S^*} \omega_i \lambda_i = s$. These constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix Λ [3]. For unauthorized sets, no such S^*, $\{\omega_i\}$ exist.

2.3.2 KP-ABE Definition

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

Setup$(\lambda, \mathcal{U}) \rightarrow (\mathrm{PP}, \mathrm{MSK})$ 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$(\text{PP}, \text{CT}, \text{SK}) \rightarrow M$ 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 3. *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 4. *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.

3 Prime Order KP-ABE

We will now present a one-use KP-ABE scheme built with prime order bilinear groups and proven secure based on the Decisional Linear Assumption. We 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:

3.1 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 $\vec{\mathbf{b}}_{1,} = g^{\vec{b}_1}, ..., \vec{\mathbf{b}}_{6,j} = g^{\vec{b}_{6,j}}$ (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 [25, 26]. We will choose two random sets of vectors:

$$\mathbb{B} := \{\vec{b}_1, \vec{b}_{2,i}, \vec{b}_3, \vec{b}_{4,i}, \vec{b}_5, \vec{b}_{6,i}\}_{i\in[k]}$$

and

$$\mathbb{B}^* := \{\vec{b}^*_1, \vec{b}^*_{2,i}, \vec{b}^*_3, \vec{b}^*_{4,i}, \vec{b}^*_5, \vec{b}^*_{6,i}\}_{i\in[k]}$$

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

$\vec{b}_i \cdot \vec{b}^*_i = 1 \pmod p$, $\vec{b}_i$ is orthogonal to all other vectors in $\mathbb{B}^*$ for $i = 1, 3, 5$
$\vec{b}_{i,j} \cdot \vec{b}^*_{i,j} = 1 \pmod p$, $\vec{b}_{i,j}$ is orthogonal to all other vectors in $\mathbb{B}^*$ for $i = 2, 4, 6, j \in [k]$

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

3.2 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. We take the classic approach of using a fresh pair of vectors for each (one-time) attribute 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).

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

Setup$(\lambda, \mathcal{U}) \rightarrow PP, MSK$ 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 $i \in [k]$ where $k = |\mathcal{U}|$ it chooses values $a_i \leftarrow \mathbb{Z}_p$ and generates a random dual orthonormal set:

$$(\mathbb{B}, \mathbb{B}^*) = (\{\vec{b}_1, \vec{b}_{2,i}, \vec{b}_3, \vec{b}_{4,i}, \vec{b}_5, \vec{b}_{6,i}\} \\ \{\vec{b}^*_1, \vec{b}^*_{2,i}, \vec{b}^*_3, \vec{b}^*_{4,i}, \vec{b}^*_5, \vec{b}^*_{6,i}\}) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$$

The public parameters PP are:

$$p, g, e(g,g)^{\alpha}, e(g,g)^{\alpha'}, \\ \{\vec{\mathbf{b}}^*_1, \vec{\mathbf{b}}^*_3\} \\ \{a_i \vec{\mathbf{b}}^*_{2,i}, a_i \vec{\mathbf{b}}^*_{4,i}\}_{i \in [k]}$$

(We use the bolded vector notation to denote the vector in the exponent, as detailed in subsection 3.1.) The MSK is:

$$\alpha, \alpha', \{\vec{\mathbf{b}}_1, a_i \vec{\mathbf{b}}_1, \vec{\mathbf{b}}_{2,i}, \vec{\mathbf{b}}_3, a_i \vec{\mathbf{b}}_3, \vec{\mathbf{b}}_{4,i} : i \in [k]\}$$

Such a construction is equipped to create keys for access policies which include attributes $i \in \mathcal{U}$.

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 $\{\lambda_i, \lambda'_i\}$: 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 i labeling a row in the policy matrix Λ, it then chooses exponents $y_i, y'_i \leftarrow \mathbb{Z}_p$ and outputs the secret key:

$$SK_{\Lambda} = \{\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\ + \lambda'_i \vec{\mathbf{b}}_3 + y'_i a_i \vec{\mathbf{b}}_3 + y'_i \vec{\mathbf{b}}_{4,i}\}_{i \text{ labels } \in \Lambda}$$

Encrypt$(M, S, PP) \rightarrow CT$ The encryption algorithm first draws $s, s' \leftarrow \mathbb{Z}_p$.

$$CT = Me(g,g)^{\alpha s + \alpha' s'}, \{s \vec{\mathbf{b}}^*_1 - s a_i \vec{\mathbf{b}}^*_{2,i} \\ + s' \vec{\mathbf{b}}^*_3 - s' a_i \vec{\mathbf{b}}^*_{4,i}\}_{i \in S}$$

(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_i such that $\sum_{i \in S^*} \omega_i \lambda_i = \alpha$ and $\sum_{i \in S^*} \omega_i \lambda'_i = \alpha'$ (recall section 2.3.1). It then computes:

$$\prod_{i \in S^*} e_n \Bigg(s\vec{\mathbf{b}}_1^* - s a_i \vec{\mathbf{b}}_{2,i}^* + s' \vec{\mathbf{b}}_3^* - s' a_i \vec{\mathbf{b}}_{4,i}^*, \\ \lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} + \lambda'_i \vec{\mathbf{b}}_3 + y'_i a_i \vec{\mathbf{b}}_3 + y'_i \vec{\mathbf{b}}_{4,i} \Bigg)^{\omega_K}$$

$$= \prod_{i \in S^*} \Big(e(g,g)^{s \lambda_i \vec{b}_1 \cdot \vec{b}_1^*} e(g,g)^{s y_i a_i \vec{b}_1 \cdot \vec{b}_1^*} \\ e(g,g)^{-s y_i a_i \vec{b}_{2,i} \cdot \vec{b}_{2,i}^*} \\ e(g,g)^{s' \lambda'_i \vec{b}_3 \cdot \vec{b}_3^*} e(g,g)^{s' y'_i a_i \vec{b}_3 \cdot \vec{b}_3^*} \\ e(g,g)^{-s' y'_i a_i \vec{b}_{4,i} \cdot \vec{b}_{4,i}^*} \Big)^{\omega_i}$$

$$= \prod_{i \in S^*} (e(g,g)^{s \lambda_i} e(g,g)^{s' \lambda'_i})^{\omega_i} \\ = e(g,g)^{s \sum_{i \in S^*} \omega_i \lambda_i + s' \sum_{i \in S^*} \omega_i \lambda'_i} \\ = e(g,g)^{s\alpha + s'\alpha'}$$

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

4 Prime Order KP-ABE 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 will now describe.

Semi-functional Ciphertext We will use 2 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:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{b}}_1^* - sa_i\vec{\mathbf{b}}_{2,i}^* + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* : (\forall i \in S)\}$$

One then chooses $s'' \leftarrow \mathbb{Z}_p$. For all $i \in [k]$, $\tilde{a}_j \leftarrow \mathbb{Z}_p$ are drawn and fixed if they do not already exist (in a semi-functional key, for instance). 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:

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{b}}_1^* - sa_i\vec{\mathbf{b}}_{2,i}^* + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* + s''\vec{\mathbf{b}}_5^* - s''a_i\vec{\mathbf{b}}_{6,i}^*\}_{i\in S}$$

Notice that in this type, the $\tilde{a}_i$ are unused (instead, the normal a_i are re-used in the semifunctional space).

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

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{b}}_1^* - sa_i\vec{\mathbf{b}}_{2,i}^* + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* + s''\vec{\mathbf{b}}_5^* - s''\tilde{a}_i\vec{\mathbf{b}}_{6,i}^*\}_{i\in S}$$

Notice that in this type, the $\tilde{a}_i$ in the semifunctional space have been decoupled from the a_i in the normal space (they are chosen independently of the a_i in the normal space and public parameters and exist nowhere else except possibly in a semifunctional key of Type 1Z or 1R (to be defined)).

Semi-functional Keys We will use 5 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:

$$SK_\Lambda = \{\lambda_i\vec{\mathbf{b}}_1 + y_i a_i\vec{\mathbf{b}}_1 + y_i\vec{\mathbf{b}}_{2,i} + \lambda_i'\vec{\mathbf{b}}_3 + y_i' a_i\vec{\mathbf{b}}_3 + y_i'\vec{\mathbf{b}}_{4,i}\}_{i \text{ labels } \in\Lambda}$$

The first 4 types of keys fall under 2 classes: a "Z" class and an "R" class. For Z-class keys one computes a linear sharing *of 0* under access policy Λ, creating shares λ_i''. For R-class 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_i''. For each $i \in [k]$, $\tilde{a}_i \leftarrow \mathbb{Z}_p$ are drawn and fixed if they do not already exist (in a semifunctional ciphertext, for instance). The next steps depend on the type of the key:

Type 0 For each i in the normal key, one chooses a new $y_i'' \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 0Z or 0R (depending on the sharing λ_i'') as:

$$\begin{aligned} SK'_\Lambda = \{&\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\ &+\lambda_i' \vec{\mathbf{b}}_3 + y_i' a_i \vec{\mathbf{b}}_3 + y_i' \vec{\mathbf{b}}_{4,i} \\ &+\lambda_i'' \vec{\mathbf{b}}_5 + y_i'' a_i \vec{\mathbf{b}}_5 + y_i'' \vec{\mathbf{b}}_{6,i}\}_{i \text{ labels } \in \Lambda} \end{aligned}$$

Notice that in this type, the $\tilde{a}_i$ are unused (instead, the normal a_i are re-used in the semifunctional space).

Type 1 For each i in the normal key, one chooses a new $y_i'' \leftarrow \mathbb{Z}_p$ and forms the semi-functional key of type 0Z or 0R (depending on the sharing λ_i'') as:

$$\begin{aligned} SK'_\Lambda = \{&\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\ &+\lambda_i' \vec{\mathbf{b}}_3 + y_i' a_i \vec{\mathbf{b}}_3 + y_i' \vec{\mathbf{b}}_{4,i} \\ &+\lambda_i'' \vec{\mathbf{b}}_5 + y_i'' \tilde{a}_i \vec{\mathbf{b}}_5 + y_i'' \vec{\mathbf{b}}_{6,i}\}_{i \text{ labels } \in \Lambda} \end{aligned}$$

Notice that in this type, the $\tilde{a}_i$ in the semifunctional space have been decoupled from the a_i in the normal space (they are chosen independently of the a_i in the normal space and public parameters and exist nowhere else except possibly in a semifunctional ciphertext of Type 1).

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

Type 2R Using shares λ_i'' 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 2R as:

$$\begin{aligned} SK'_\Lambda = \{&\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\ &+\lambda_i' \vec{\mathbf{b}}_3 + y_i' a_i \vec{\mathbf{b}}_3 + y_i' \vec{\mathbf{b}}_{4,i} \\ &+\lambda_i'' \vec{\mathbf{b}}_5\}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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}, and Game_{ℓ_4} for $\ell = 0, ..., Q$. In each game, the first ℓ keys are semi-functional of type 2R, and all other keys are normal. They differ in the construction of the ℓth key and the ciphertext as follows:

Game$_{\ell_0}$ In this game, the ℓth key is type 0Z and the ciphertext is type 0.

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 1R and the ciphertext is type 1.

Game$_{\ell_3}$ In this game, the ℓth key is type 0R and the ciphertext is type 0.

Game$_{\ell_4}$ In this game, the ℓth key is type 2R and the ciphertext is type 0.

Note that under this definition, we have that in **Game**$_{0_4}$, 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$_{0_4}$, then to Game$_{1_0}$, next to Game$_{1_1}$, next to Game$_{1_2}$, next to Game$_{1_3}$, next to Game$_{1_4}$ and then to Game$_{2_0}$ and so on. We then arrive at Game$_{Q_4}$, where the ciphertext is semifunctional of type 0 and *all* of the keys given to the attacker are semi-functional of type 2R.

There are two more games in the security proof: Game$_{penultimate}$ and Game$_{final}$. We transition from Game$_{Q_4}$ 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:

$$\begin{aligned} Me(g,g)^{\alpha s+\alpha' s'}, \{ & x\vec{\mathbf{b}}_1^* - xa_i\vec{\mathbf{b}}_{2,i}^* \\ & + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* \\ & + s''\vec{\mathbf{b}}_5^* - s''a_i\vec{\mathbf{b}}_{6,i}^* \}_{i\in S} \end{aligned}$$

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 5. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{real}$ and $Game_{0_4}$.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between Game_{real} and Game_{0_4}, 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 $\tilde{\alpha}, \tilde{\alpha}', a_i \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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

$$\begin{aligned}
\vec{b}_1^* &= y_1 \vec{d}_{1,}^* + y_1 c_1 \vec{d}_5^* & \vec{b}_{2,i}^* &= y_1 \vec{d}_{2,i}^* + y_1 c_1 \vec{d}_{6,i}^* \\
\vec{b}_3^* &= y_2 \vec{d}_3^* + y_2 c_2 \vec{d}_5^* & \vec{b}_{4,i}^* &= y_2 \vec{d}_{4,i}^* + y_2 c_2 \vec{d}_{6,i}^* \\
\vec{b}_5^* &= \vec{d}_5^* & \vec{b}_{6,i}^* &= \vec{d}_{6,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= y_1^{-1} \vec{d}_1 & \vec{b}_{2,i} &= y_1^{-1} \vec{d}_{2,i} \\
\vec{b}_3 &= y_2^{-1} \vec{d}_3 & \vec{b}_{4,i} &= y_2^{-1} \vec{d}_{4,i} \\
\vec{b}_5 &= \vec{d}_5 - c_1 \vec{d}_1 - c_2 \vec{d}_3 & \vec{b}_{6,i} &= \vec{d}_{6,i} - c_1 \vec{d}_{2,i} - c_2 \vec{d}_{4,i}
\end{aligned}$$

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

The public parameters are constructed as:

$$\begin{aligned}
&p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1 \tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2 \tilde{\alpha}'}, \\
&\{\vec{\mathbf{b}}_1^*, \vec{\mathbf{b}}_3^*\} \\
&\{a_i \vec{\mathbf{b}}_{2,i}^*, a_i \vec{\mathbf{b}}_{4,i}^*\}_{i \in [k]}
\end{aligned}$$

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 $\tilde{\lambda}_i, \tilde{\lambda}_i'$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively. For each i attribute label in Λ, it generates $\tilde{y}_i, \tilde{y}_i' \leftarrow \mathbb{Z}_p$ and outputs:

$$\begin{aligned}
SK_\Lambda = \{&\tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i a_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 \vec{\mathbf{b}}_{2,i} \\
&+ \tilde{\lambda}_i' y_2 \vec{\mathbf{b}}_3 + \tilde{y}_i' a_i y_2 \vec{\mathbf{b}}_3 + \tilde{y}_i' y_2 \vec{\mathbf{b}}_{4,i} \\
&: (\forall i \text{ labels} \in \Lambda)\}
\end{aligned}$$

Note that the simulator cannot make any of the $\vec{\mathbf{b}}_x$ in this honest key alone (because of the y_1^{-1}, y_2^{-1} terms). However, the simulator is able to make $y_1\vec{\mathbf{b}}_1 = \vec{\mathbf{d}}_1$, 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 $y_i = \tilde{y}_i y_1$ and $y'_i = \tilde{y}'_i y_2$, which are uniformly distributed elements of $\mathbb{Z}_p$. The shares $\lambda_i = \tilde{\lambda}_i y_1$ and $\lambda'_i = \tilde{\lambda}'_i y_2$ 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, The following ciphertext is then constructed and provided:

$$\begin{aligned} & Me(g,g)^{y_1\tilde{\alpha}s} e(g,g)^{y_2\tilde{\alpha}'s}, \\ & \{g^{\vec{d}^*_1} g^{\vec{d}^*_3} T^{\vec{d}^*_5} \\ & \quad (g^{\vec{d}^*_{2,i}} g^{\vec{d}^*_{4,i}} T^{\vec{d}^*_{6,i}})^{-a_i}\}_{i\in S} \end{aligned}$$

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

$$\begin{aligned} & Me(g,g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'}, \\ & \{y_1^{-1}\vec{\mathbf{b}}^*_1 - y_1^{-1}a_i\vec{\mathbf{b}}^*_{2,i} \\ & + y_2^{-1}\vec{\mathbf{b}}^*_3 - y_2^{-1}a_i\vec{\mathbf{b}}^*_{4,i} \\ & + r\vec{\mathbf{b}}^*_5 - ra_i\vec{\mathbf{b}}^*_{6,i}\}_{i\in S} \end{aligned}$$

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 $s = y_1^{-1}$, $s' = y_2^{-1}$, 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 $s = y_1^{-1}$, $s' = y_2^{-1}$, $s'' = r$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{0_4} .

Therefore, any adversary with non-negligible difference in advantage between Game_{real} and Game_{0_4} 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 6. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between* $\text{Game}_{(\ell-1)_4}$ *and* Game_{ℓ_0} *for any* ℓ *from* 1 *to* Q.

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between $\text{Game}_{(\ell-1)_4}$ 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 $\tilde{\alpha}, \tilde{\alpha}', a_i \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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

$$\begin{aligned}
\vec{b}_1^* &= \vec{d}_1^* & \vec{b}_{2,i}^* &= \vec{d}_{2,i}^* \\
\vec{b}_3^* &= \vec{d}_3^* & \vec{b}_{4,i}^* &= \vec{d}_{4,i}^* \\
\vec{b}_5^* &= \vec{d}_5^* - c_1\vec{d}_1^* - c_2\vec{d}_3^* & \vec{b}_{6,i}^* &= \vec{d}_{6,i}^* - c_1\vec{d}_{2,i}^* - c_2\vec{d}_{4,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= \vec{d}_1 + c_1\vec{d}_5 & \vec{b}_{2,i} &= \vec{d}_{2,i} + c_1\vec{d}_{6,i} \\
\vec{b}_3 &= \vec{d}_3 + c_2\vec{d}_5 & \vec{b}_{4,i} &= \vec{d}_{4,i} + c_2\vec{d}_{6,i} \\
\vec{b}_5 &= \vec{d}_5 & \vec{b}_{6,i} &= \vec{d}_{6,i}
\end{aligned}$$

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

The public parameters are constructed as:

$$\begin{aligned}
&p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
&\{\vec{\mathbf{b}}_1^*, \vec{\mathbf{b}}_3^*\} \\
&\{a_i\vec{\mathbf{b}}_{2,i}^*, a_i\vec{\mathbf{b}}_{4,i}^*\}_{i\in S}
\end{aligned}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be 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. The following ciphertext is then constructed and provided:

$$\begin{aligned}
&Me(g, g^{y_1\tilde{\alpha}})^{s''} e(g, g^{y_2\tilde{\alpha}'})^{s''}, \\
&\{(g^{\vec{d}_5^*})^{s''} \\
&(g^{\vec{d}_{6,i}^*})^{-s''a_i}\}_{i\in S}
\end{aligned}$$

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

$$\begin{aligned}
&Me(g, g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'}, \\
&\{c_1 s''\vec{\mathbf{b}}_1^* - c_1 s'' a_i \vec{\mathbf{b}}_{2,i}^* \\
&+c_2 s''\vec{\mathbf{b}}_3^* - c_2 s'' a_i \vec{\mathbf{b}}_{4,i}^* \\
&+s''\vec{\mathbf{b}}_5^* - s'' a_i \vec{\mathbf{b}}_{6,i}^*\}_{i\in S}
\end{aligned}$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$ are both 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 $\tilde{\lambda}_i, \tilde{\lambda}'_i$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ under policy Λ respectively.

$$\begin{aligned} SK_\Lambda = \{ & \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{\lambda}_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{\lambda}'_i} \\ & \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{y}_i a_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{y}'_i a_i} \\ & \left((g^{y_1})^{\vec{d}_{2,i}}(g^{y_1 c_1})^{\vec{d}_{6,i}}\right)^{\tilde{y}_i} \left((g^{y_2})^{\vec{d}_{4,i}}(g^{y_2 c_2})^{\vec{d}_{6,i}}\right)^{\tilde{y}'_i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\begin{aligned} SK_\Lambda = \{ & \tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 a_i \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 \vec{\mathbf{b}}_{2,i} \\ & + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 a_i \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 \vec{\mathbf{b}}_{4,i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

The distribution of keys constructed this way is identical to the distribution of normal honest keys where $y_i = \tilde{y}_i y_1$ and $y'_i = \tilde{y}'_i y_2$, which are uniformly distributed. The shares $\lambda_i = \tilde{\lambda}_i y_1$ and $\lambda'_i = \tilde{\lambda}'_i y_2$ 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''_i of u. It then creates a semi-functional key of type 4R by adding $\lambda''_i \vec{\mathbf{b}}_5$ to each term in the secret key set (it can create these using its knowledge of $\vec{b}_5 = \vec{d}_5$):

$$\begin{aligned} SK_\Lambda = \{ & \tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 a_i \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 \vec{\mathbf{b}}_{2,i} \\ & + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 a_i \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 \vec{\mathbf{b}}_{4,i} \\ & + \lambda''_i \vec{\mathbf{b}}_5 \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\begin{aligned} SK_\Lambda = \{ & \left(g^{\vec{d}_1} g^{\vec{d}_3} T^{\vec{d}_5}\right)^{\tilde{\lambda}''_i} \\ & \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{\lambda}_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{\lambda}'_i} \\ & \left(g^{\vec{d}_1} g^{\vec{d}_3} T^{\vec{d}_5}\right)^{\tilde{y}''_i a_i} \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{y}_i a_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{y}'_i a_i} \\ & \left(g^{\vec{d}_{2,i}} g^{\vec{d}_{4,i}} T^{\vec{d}_{6,i}}\right)^{\tilde{y}''_i} \left((g^{y_1})^{\vec{d}_{2,i}}(g^{y_1 c_1})^{\vec{d}_{6,i}}\right)^{\tilde{y}_i} \left((g^{y_2})^{\vec{d}_{4,i}}(g^{y_2 c_2})^{\vec{d}_{6,i}}\right)^{\tilde{y}'_i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\{(\tilde{\lambda}_i'' + \tilde{\lambda}_i y_1)\vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}_i'')a_i\vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}_i'')\vec{\mathbf{b}}_{2,i}$$
$$+(\tilde{\lambda}_i'' + \tilde{\lambda}_i' y_2)\vec{\mathbf{b}}_3 + (\tilde{y}_i' y_2 + \tilde{y}_i'')a_i\vec{\mathbf{b}}_3 + (\tilde{y}_i' y_2 + \tilde{y}_i'')\vec{\mathbf{b}}_{4,i}$$
$$+r\tilde{\lambda}_i''\vec{\mathbf{b}}_5 + \tilde{y}_i'' r a_i\vec{\mathbf{b}}_5 + \tilde{y}_i'' r\vec{\mathbf{b}}_{6,i}\}_{i \text{ labels} \in \Lambda}$$

Since the $\tilde{\lambda}_i''$ are a sharing of zero and the $\tilde{\lambda}_i, \tilde{\lambda}_i'$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_i = \tilde{\lambda}_i'' + \tilde{\lambda}_i y_1$ and $\lambda_i' = \tilde{\lambda}_i'' + \tilde{\lambda}_i' y_2$ 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 $y_i = \tilde{y}_i y_1 + \tilde{y}_i''$, and $y_i' = \tilde{y}_i' y_2 + \tilde{y}_i''$ which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of $\text{Game}_{(\ell-1)_6}$.

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 $y_i = \tilde{y}_i y_1 + \tilde{y}_i''$, $y_i' = \tilde{y}_i' y_2 + \tilde{y}_i''$, and $y_i'' = \tilde{y}_i'' r$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$. Since the $\tilde{\lambda}_i''$ are a sharing of zero, the shares $\lambda_i'' = r\tilde{\lambda}_i''$ 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 $\text{Game}_{(\ell-1)_4}$ 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 7. *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_i in the semi-functional space.

Consider the following simulation of the ABE security game: first use the normal $Dual(\mathbb{Z}_p^{3+3k})$ procedure to generate orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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 2R 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:

$$\begin{aligned}
\vec{b}_1^* &= \vec{d}_1^* & \vec{b}_{2,i}^* &= \vec{d}_{2.i}^* \\
\vec{b}_3^* &= \vec{d}_3^* & \vec{b}_{4,i}^* &= \vec{d}_{4,i}^* \\
\vec{b}_5^* &= \vec{d}_5^* + a_i' \vec{d}_{6,i}^* & \vec{b}_{6,i}^* &= \vec{d}_{6,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= \vec{d}_1 & \vec{b}_{2,i} &= \vec{d}_{2,i} \\
\vec{b}_3 &= \vec{d}_3 & \vec{b}_{4,i} &= \vec{d}_{4,i} \\
\vec{b}_5 &= \vec{d}_5 & \vec{b}_{6,i} &= \vec{d}_{6,i} - a_i' \vec{d}_5
\end{aligned}$$

(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).

So, the $(\mathbb{B}, \mathbb{B}^*)$ formed using this alternative transformation have the same distribution as the sets straight from the $Dual(\mathbb{Z}_p^{3+3k})$ procedure. Furthermore, since this transformation only causes differences in the $\vec{d}_5^*$, and $\vec{b}_{6,i}$ 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 2R, and all honest keys (since no $\vec{d}_5^*$, and $\vec{b}_{6,i}$ are used in these objects). The only difference is seen in the ℓth key and the challenge ciphertext, where the transformation causes the a_i in the semifunctional space to lose their correlation with the a_i in the normal space.

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

$$\begin{aligned}
&\{\lambda_i \vec{\mathbf{d}}_1 + y_i a_i \vec{\mathbf{d}}_1 + y_i \vec{\mathbf{d}}_{2,i} \\
&+\lambda_i' \vec{\mathbf{d}}_3 + y_i' a_i \vec{\mathbf{d}}_3 + y_i' \vec{\mathbf{d}}_{4,i} \\
&+\lambda_i'' \vec{\mathbf{d}}_5 + y_i'' a_i \vec{\mathbf{d}}_5 + y_i'' \vec{\mathbf{d}}_{6,i}\}_{i \text{ labels } \in \Lambda}
\end{aligned}$$

$$\begin{aligned}
&= \{\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\
&+\lambda_i' \vec{\mathbf{b}}_3 + y_i' a_i \vec{\mathbf{b}}_3 + y_i' \vec{\mathbf{b}}_{4,i} \\
&+\lambda_i'' \vec{\mathbf{b}}_5 + y_i'' a_i \vec{\mathbf{b}}_5 + y_i'' (\vec{\mathbf{b}}_{6,i} + a_i' \vec{\mathbf{b}}_5)\}_{i \text{ labels } \in \Lambda}
\end{aligned}$$

$$\begin{aligned}
&= \{\lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{b}_{2,i} \\
&+\lambda_i' \vec{\mathbf{b}}_3 + y_i' a_i \vec{\mathbf{b}}_3 + y_i' \vec{\mathbf{b}}_{4,i} \\
&+\lambda_i'' \vec{\mathbf{b}}_5 + y_i'' (a_i + a_i') \vec{\mathbf{b}}_5 + y_i'' \vec{\mathbf{b}}_{6,i}\}_{i \text{ labels } \in \Lambda}
\end{aligned}$$

where here, the decoupled $\tilde{a}_i = a_i + a_i'$.

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

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{d}}_1^* - sa_i\vec{\mathbf{d}}_{2,i}^* + s'\vec{\mathbf{d}}_3^* - s'a_i\vec{\mathbf{d}}_{4,i}^* + s''\vec{\mathbf{d}}_5^* - s''a_i\vec{\mathbf{d}}_{6,i}^*\}_{i\in S}$$

$$Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{b}}_1^* - sa_i\vec{\mathbf{b}}_{2,i}^* + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* + s''(\vec{\mathbf{b}}_5^* - a_i'\vec{\mathbf{b}}_{6,i}^*) - s''a_i\vec{\mathbf{b}}_{6,i}^*\}_{i\in S}$$

$$= Me(g,g)^{\alpha s+\alpha' s'}, \{s\vec{\mathbf{b}}_1^* - sa_i\vec{\mathbf{b}}_{2,i}^* + s'\vec{\mathbf{b}}_3^* - s'a_i\vec{\mathbf{b}}_{4,i}^* + s''\vec{\mathbf{b}}_5^* - s''(a_i + a_i')\vec{\mathbf{b}}_{6,i}^*\}_{i\in S}$$

where again, the $\tilde{a}_i = ai + a_i'$ are now independent of the a_i in the normal space 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 attacker can achieve a non-negligible difference in advantage between them. □

Lemma 8. *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. Recall that in both Game_{ℓ_1} and Game_{ℓ_2}, the ciphertext is semi-functional of type 1, all keys after the ℓth key request are normal, and the first $\ell - 1$ keys are semi-functional of type 2R. The only difference is the ℓth key (either semi-functional of type 1Z or 1R). In Game_{ℓ_1}, the shares λ_i'' are a sharing of 0, while in Game_{ℓ_2}, 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 i not included in the ciphertext, then the distributions of the elements of the key indexed by i are identical in both cases, since the λ_i'' are masked by a $\tilde{a}_i y_i''$ in the exponent of the $\vec{b}_5$ vector, where the $\tilde{a}_i$ 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, and further do not appear in the public parameters or the challenge ciphertext since we

are considering the case that i is not included in the challenge ciphertext), therefore the value of all such λ_i'' is hidden information-theoretically.

For attributes i used in the ciphertext, this argument does not apply (since these $\tilde{a}_i$ do appear elsewhere - in the challenge 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 p). Therefore, there exists some vector $\vec{w} \in \mathbb{Z}_p^n$ orthogonal to the span of these rows which is not orthogonal to $(1, 0, ..., 0)$ modulo p (since $\mathbb{Z}_p$ 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 λ_i'' which comprise $\vec{\lambda''}$ were generated by taking $\Lambda\vec{r} = \vec{\lambda''}$ 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 λ_i'' that are not information-theoretically hidden by the previous argument are the same. All that is seen are λ''_i 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 p 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 λ_i'' are formed by taking a sharing of zero or a random element of $\mathbb{Z}_p$, then no algorithm can tell the difference between the two games, and so we have proven the lemma.

□

Lemma 9. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_2}$ and $Game_{\ell_3}$ for any ℓ from 1 to Q.*

Proof. This information-theoretic argument is the same as Lemma 7 in reverse, with the only difference being that the λ_i'' 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_i'' either way. □

The last step in the hybrid is to change the ℓth key from semi-functional of type 0R to 2R (losing its semifunctional space a_i):

Lemma 10. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between $Game_{\ell_3}$ and $Game_{\ell_4}$ for any ℓ from 1 to Q.*

Proof. If an algorithm $\mathcal{A}$ has non-negligible difference in advantage between Game_{ℓ_3} and Game_{ℓ_4} 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 $\tilde{\alpha}, \tilde{\alpha}', a_i \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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

$$\begin{aligned}
\vec{b}_1^* &= \vec{d}_1^* & \vec{b}_{2,i}^* &= \vec{d}_{2,i}^* \\
\vec{b}_3^* &= \vec{d}_3^* & \vec{b}_{4,i}^* &= \vec{d}_{4,i}^* \\
\vec{b}_5^* &= \vec{d}_5^* - c_1\vec{d}_1^* - c_2\vec{d}_3^* & \vec{b}_{6,i}^* &= \vec{d}_{6,i}^* - c_1\vec{d}_{2,i}^* - c_2\vec{d}_{4,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= \vec{d}_1 + c_1\vec{d}_5 & \vec{b}_{2,i} &= \vec{d}_{2,i} + c_1\vec{d}_{6,i} \\
\vec{b}_3 &= \vec{d}_3 + c_2\vec{d}_5 & \vec{b}_{4,i} &= \vec{d}_{4,i} + c_2\vec{d}_{6,i} \\
\vec{b}_5 &= \vec{d}_5 & \vec{b}_{6,i} &= \vec{d}_{6,i}
\end{aligned}$$

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

The public parameters are constructed as:

$$\begin{aligned}
&p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
&\{\vec{\mathbf{b}}_1^*, \vec{\mathbf{b}}_3^*\} \\
&\{a_i\vec{\mathbf{b}}_{2,i}^*, a_i\vec{\mathbf{b}}_{4,i}^*\}_{i\in[k]}
\end{aligned}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be 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. The following ciphertext is then constructed and provided:

$$\begin{aligned}
&Me(g, g^{y_1c_1})^{\tilde{\alpha}s''}e(g, g^{y_2c_2})^{\tilde{\alpha}'s''}, \\
&\{(g^{\vec{d}_5^*})^{s''} \\
&\quad (g^{\vec{d}_{6,i}^*})^{-s''a_i}\}_{i\in S}
\end{aligned}$$

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

$$\begin{aligned} & Me(g,g)^{y_1\tilde{\alpha}s + y_2\tilde{\alpha}'s'}, \\ & \{c_1 s'' \vec{\mathbf{b}}_1^* - c_1 s'' a_i \vec{\mathbf{b}}_{2,i}^* \\ & + c_2 s'' \vec{\mathbf{b}}_3^* - c_2 s'' a_i \vec{\mathbf{b}}_{4,i}^* \\ & + s'' \vec{\mathbf{b}}_5^* - s'' a_i \vec{\mathbf{b}}_{6,i}^* \}_{i \in S} \end{aligned}$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$ are 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 $\tilde{\lambda}_i, \tilde{\lambda}'_i$: sharings of $\tilde{\alpha}, \tilde{\alpha}'$ under policy Λ respectively.

$$\begin{aligned} SK_\Lambda = \{ & \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_5} \right)^{\tilde{\lambda}_i} \left((g^{y_2})^{\vec{d}_3} (g^{y_2 c_2})^{\vec{d}_5} \right)^{\tilde{\lambda}'_i} \\ & \left((g^{y_1})^{\vec{d}_1} (g^{y_1 c_1})^{\vec{d}_5} \right)^{\tilde{y}_i a_i} \left((g^{y_2})^{\vec{d}_3} (g^{y_2 c_2})^{\vec{d}_5} \right)^{\tilde{y}'_i a_i} \\ & \left((g^{y_1})^{\vec{d}_{2,i}} (g^{y_1 c_1})^{\vec{d}_{6,i}} \right)^{\tilde{y}_i} \left((g^{y_2})^{\vec{d}_{4,i}} (g^{y_2 c_2})^{\vec{d}_{6,i}} \right)^{\tilde{y}'_i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\begin{aligned} SK_\Lambda = \{ & \tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 a_i \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 \vec{\mathbf{b}}_{2,i} \\ & + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 a_i \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 \vec{\mathbf{b}}_{4,i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

The distribution of keys constructed this way is identical to the distribution of normal honest keys where $y_i = \tilde{y}_i y_1$ and $y'_i = \tilde{y}'_i y_2$, which are uniformly distributed. The shares $\lambda_i = \tilde{\lambda}_i y_1$ and $\lambda'_i = \tilde{\lambda}'_i y_2$ 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''_i of u. It then creates a semi-functional key of type 2R by adding $\lambda''_i \vec{\mathbf{b}}_5$ to each term in the secret key set (it can create these using its knowledge of $\vec{b}_5 = \vec{d}_5$):

$$\begin{aligned} SK_\Lambda = \{ & \tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 a_i \vec{\mathbf{b}}_1 + \tilde{y}_i y_1 \vec{\mathbf{b}}_{2,i} \\ & + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 a_i \vec{\mathbf{b}}_3 + \tilde{y}'_i y_2 \vec{\mathbf{b}}_{4,i} \\ & + \lambda''_i \vec{\mathbf{b}}_5 \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

On the ℓth key request, for each i attribute label in Λ, the simulator first creates sharings $\tilde{\lambda}_i$, $\tilde{\lambda}'_i$ of $\tilde{\alpha}, \tilde{\alpha}'$ respectively. It then creates a sharing λ''_i of

u under Λ, then generates uniformly random $\tilde{y}_i, \tilde{y}'_i, \tilde{y}''_i \in \mathbb{Z}_p$ and outputs:

$$SK_\Lambda = \{ \left(g^{\vec{d}_5}\right)^{\lambda''_i} \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{\lambda}_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{\lambda}'_i} \left(g^{\vec{d}_1} g^{\vec{d}_3} T^{\vec{d}_5}\right)^{\tilde{y}''_i a_i} \left((g^{y_1})^{\vec{d}_1}(g^{y_1 c_1})^{\vec{d}_5}\right)^{\tilde{y}_i a_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2 c_2})^{\vec{d}_5}\right)^{\tilde{y}'_i a_i} \left(g^{\vec{d}_{2,i}} g^{\vec{d}_{4,i}} T^{\vec{d}_{6,i}}\right)^{\tilde{y}''_i} \left((g^{y_1})^{\vec{d}_{2,i}}(g^{y_1 c_1})^{\vec{d}_{6,i}}\right)^{\tilde{y}_i} \left((g^{y_2})^{\vec{d}_{4,i}}(g^{y_2 c_2})^{\vec{d}_{6,i}}\right)^{\tilde{y}'_i} \}_{i \text{ labels } \in \Lambda}$$

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

$$\{\tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}''_i) a_i \vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}''_i) \vec{\mathbf{b}}_{2,i} + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_2 + \tilde{y}''_i) a_i \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_2 + \tilde{y}''_i) \vec{\mathbf{b}}_{6,i} + \lambda''_i \vec{\mathbf{b}}_5 + \tilde{y}''_i r a_i \vec{\mathbf{b}}_5 + \tilde{y}''_i r \vec{\mathbf{b}}_{6,i} \}_{i \text{ labels } \in \Lambda}$$

Since the $\tilde{\lambda}_i, \tilde{\lambda}'_i$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_i = \tilde{\lambda}_i y_1$ and $\lambda'_i = \tilde{\lambda}'_i y_2$ 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 2R where $y_i = \tilde{y}_i y_1 + \tilde{y}''_i$, and $y'_i = \tilde{y}'_i y_2 + \tilde{y}''_i$, which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{ℓ_4}.

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 $y_i = \tilde{y}_i y_1 + \tilde{y}''_i$, $y'_i = \tilde{y}'_i y_2 + \tilde{y}''_i$, and $y''_i = \tilde{y}''_i r$, 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_3} .

Therefore, any adversary with non-negligible difference in advantage between Game_{ℓ_3} and Game_{ℓ_4} 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_4}: where the semi-functional ciphertext is of type 0 and all keys are semi-functional of type 2R. 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 11. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between* Game_{Q_4} *and* $\text{Game}_{penultimate}$.

Proof. If an algorithm $\mathcal{A}$ is able to achieve a non-negligible difference in advantage between Game_{Q_4} 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 $\tilde{\alpha}, \tilde{\alpha}', a_i \leftarrow \mathbb{Z}_p$, and generating orthonormal sets $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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

$$\begin{aligned}
\vec{b}_1^* &= \vec{d}_1^* & \vec{b}_{2,i}^* &= \vec{d}_{2,i}^* \\
\vec{b}_3^* &= \vec{d}_3^* & \vec{b}_{4,i}^* &= \vec{d}_{4,i}^* \\
\vec{b}_5^* &= \vec{d}_5^* - c_1\vec{d}_1^* - c_2\vec{d}_3^* & \vec{b}_{6,i}^* &= \vec{d}_{6,i}^* - c_1\vec{d}_{2,i}^* - c_2\vec{d}_{4,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= \vec{d}_1 + c_1\vec{d}_5 & \vec{b}_{2,i} &= \vec{d}_{2,i} + c_1\vec{d}_{6,i} \\
\vec{b}_3 &= \vec{d}_3 + c_2\vec{d}_5 & \vec{b}_{5,i} &= \vec{d}_{4,i} + c_2\vec{d}_{6,i} \\
\vec{b}_5 &= \vec{d}_5 & \vec{b}_{6,i} &= \vec{d}_{6,i}
\end{aligned}$$

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

The public parameters are constructed as:

$$\begin{aligned}
&p, g, e(g, g^{y_1})^{\tilde{\alpha}} = e(g, g)^{y_1\tilde{\alpha}}, e(g, g^{y_2})^{\tilde{\alpha}'} = e(g, g)^{y_2\tilde{\alpha}'}, \\
&\{\vec{\mathbf{b}}_1^*, \vec{\mathbf{b}}_3^* \\
&\{a_i\vec{\mathbf{b}}_{2,i}^*, a_i\vec{\mathbf{b}}_{4,i}^*\}_{i\in[k]}
\end{aligned}$$

implicitly defining $\alpha = y_1\tilde{\alpha}$ and $\alpha' = y_2\tilde{\alpha}'$. (Note that all the $\vec{\mathbf{b}}^*$ terms can be 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. The following ciphertext is then constructed and provided:

$$\begin{aligned}
&Me(g, g^{y_1c_1})^{\tilde{\alpha}s''}e(g, g^{y_2c_2})^{\tilde{\alpha}'s''}, \\
&\{(g^{\vec{d}_5^*})^{s''} \\
&\ (g^{\vec{d}_{6,i}^*})^{-s''a_i} \\
&\ :(\forall i \in S)\}
\end{aligned}$$

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

$$\begin{aligned} & Me(g,g)^{y_1\tilde{\alpha}s+y_2\tilde{\alpha}'s'}, \\ & \{c_1 s''\vec{\mathbf{b}}_1^* - c_1 s'' a_i \vec{\mathbf{b}}_{2,i}^* \\ & + c_2 s''\vec{\mathbf{b}}_3^* - c_2 s'' a_i \vec{\mathbf{b}}_{4,i}^* \\ & + s''\vec{\mathbf{b}}_5^* - s'' a_i \vec{\mathbf{b}}_{6,i}^* \}_{i\in S} \end{aligned}$$

Which is properly distributed as a semi-functional ciphertext of type 0 (appropriate for both games) where $s = c_1 s''$, $s' = c_2 s''$ are all independently and uniformly randomly distributed in $\mathbb{Z}_p$.

To respond to key requests for policies Λ, the simulator first creates sharings $\tilde{\lambda}_i$, $\tilde{\lambda}'_i$, and λ''_i 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 $\tilde{y}_i, \tilde{y}'_i, \tilde{y}''_i \in \mathbb{Z}_p$ and outputs:

$$\begin{aligned} SK_\Lambda = \{ & \left((g^{y_1})^{\vec{d}_1}(g^{y_1c_1})^{\vec{d}_5}\right)^{\tilde{\lambda}_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2c_2})^{\vec{d}_5}\right)^{\tilde{\lambda}'_i} \\ & \left(g^{\vec{d}_1} g^{\vec{d}_3} T^{\vec{d}_5}\right)^{\tilde{y}''_i a_i} \left((g^{y_1})^{\vec{d}_1}(g^{y_1c_1})^{\vec{d}_5}\right)^{\tilde{y}_i a_i} \left((g^{y_2})^{\vec{d}_3}(g^{y_2c_2})^{\vec{d}_5}\right)^{\tilde{y}'_i a_i} \\ & \left(g^{\vec{d}_{2,i}} g^{\vec{d}_{4,i}} T^{\vec{d}_{6,i}}\right)^{\tilde{y}''_i} \left((g^{y_1})^{\vec{d}_{2,i}}(g^{y_1c_1})^{\vec{d}_{6,i}}\right)^{\tilde{y}_i} \left((g^{y_2})^{\vec{d}_{4,i}}(g^{y_2c_2})^{\vec{d}_{6,i}}\right)^{\tilde{y}'_i} \\ & g^{\lambda''_i \vec{d}_5} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\begin{aligned} & \{\tilde{\lambda}_i y_1 \vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}''_i) a_i \vec{\mathbf{b}}_1 + (\tilde{y}_i y_1 + \tilde{y}''_i)\vec{\mathbf{b}}_{2,i} \\ & + \tilde{\lambda}'_i y_2 \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_2 + \tilde{y}''_i) a_i \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_2 + \tilde{y}''_i)\vec{\mathbf{b}}_{4,i} \\ & + \lambda''_i \vec{\mathbf{b}}_5 + \tilde{y}''_i r a_i \vec{\mathbf{b}}_5 + \tilde{y}''_i r \vec{\mathbf{b}}_{6,i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

Since the $\tilde{\lambda}_i, \tilde{\lambda}'_i$ are sharings of $\tilde{\alpha}, \tilde{\alpha}'$ respectively, then the $\lambda_i = \tilde{\lambda}_i y_1$ and $\lambda'_i = \tilde{\lambda}'_i y_2$ 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 2R where $y_i = \tilde{y}_i y_1 + \tilde{y}''_i$, and $y'_i = \tilde{y}'_i y_2 + \tilde{y}''_i$ which are all distributed as uniformly random elements of $\mathbb{Z}_p$, so the simulator's behavior is exactly that of Game_{Q_4}.

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 $y_i = \tilde{y}_i y_1 + \tilde{y}''_i$, $y'_i = \tilde{y}'_i y_2 + \tilde{y}''_i$, and $y''_i = \tilde{y}''_i r$, 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_4} 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 12. *Under the 2-Linear Assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between* $Game_{penultimate}$ *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 $(\mathbb{D}, \mathbb{D}^*) \leftarrow Dual(\mathbb{Z}_p^{3+3k})$.

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

$$\begin{aligned}
\vec{b}_1^* &= \vec{d}_1^* & \vec{b}_{2,i}^* &= \vec{d}_{2,i}^* \\
\vec{b}_3^* &= y_1 \vec{d}_3^* + y_1 c_1 \vec{d}_1^* & \vec{b}_{4,i}^* &= y_1 \vec{d}_{4,i}^* + y_1 c_1 \vec{d}_{2,i}^* \\
\vec{b}_5^* &= y_2 \vec{d}_5^* + y_2 c_2 \vec{d}_1^* & \vec{b}_{6,i}^* &= y_2 \vec{d}_{6,i}^* + y_2 c_2 \vec{d}_{2,i}^*
\end{aligned}$$

$$\begin{aligned}
\vec{b}_1 &= \vec{d}_1 - c_1 \vec{d}_3 - c_2 \vec{d}_5 & \vec{b}_{2,i} &= \vec{d}_{2,i} - c_1 \vec{d}_{4,i} - c_2 \vec{d}_{6,i} \\
\vec{b}_3 &= y_1^{-1} \vec{d}_3 & \vec{b}_{4,i} &= y_1^{-1} \vec{d}_{4,i} \\
\vec{b}_5 &= y_2^{-1} \vec{d}_5 & \vec{b}_{6,i} &= y_2^{-1} \vec{d}_{6,i}
\end{aligned}$$

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

The public parameters are constructed as:

$$p, g, e(g, g^{y_1}) = e(g, g)^{y_1}, e(g, g^{y_1})^{\tilde{\alpha}'} e(g^{y_1}, g^{y_1 c_1}) = e(g, g)^{y_1 \tilde{\alpha}' + y_1^2 c_1},$$
$$\{\vec{\mathbf{b}}_1^*, \vec{\mathbf{b}}_3^*\}$$
$$\{a_i \vec{\mathbf{b}}_{2,i}^*, a_i \vec{\mathbf{b}}_{4,i}^*\}_{i \in [k]}$$

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 $\tilde{\lambda}_i'$, and $\tilde{\lambda}_i''$: 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_i} where the λ_i are a sharing of y_1 under Λ. Note that

the simulator does not have the value of y_1 directly. Instead, it has g^{y_1} from the 2-LIN challenge. It can still generate g^{λ_i} where the λ_i are a sharing of y_1 by performing the same share-generation procedure "in the exponent" by choosing random r_i, computing g^{r_i}, and raising these elements along with g^{y_1} to appropriate exponents from the policy matrix / computing products to get the desired dot product in the exponent.

Next, the simulator draws $y_i, \tilde{y}'_i, \tilde{y}''_i \leftarrow \mathbb{Z}_p$ and constructs:

$$\begin{aligned} SK_\Lambda = \{ & g^{\lambda_i \vec{d}_1} g^{\tilde{\lambda}'_i \vec{d}_3} g^{\tilde{\lambda}''_i \vec{d}_5} \\ & g^{y_i a_i \vec{d}_1} g^{\tilde{y}'_i a_i \vec{d}_3} g^{\tilde{y}''_i a_i \vec{d}_5} \\ & g^{y_i \vec{d}_{2,i}} g^{\tilde{y}'_i \vec{d}_{4,i}} g^{\tilde{y}''_i \vec{d}_{6,i}} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

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

$$\begin{aligned} SK_\Lambda = \{ & \lambda_i \vec{\mathbf{b}}_1 + y_i a_i \vec{\mathbf{b}}_1 + y_i \vec{\mathbf{b}}_{2,i} \\ & + (\tilde{\lambda}'_i y_1 + \lambda_i y_1 c_1) \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_1 + y_i y_1 c_1) a_i \vec{\mathbf{b}}_3 + (\tilde{y}'_i y_1 + y_i y_1 c_1) \vec{\mathbf{b}}_{4,i} \\ & + (\tilde{\lambda}''_i y_2 + \lambda_i y_2 c_2) \vec{\mathbf{b}}_5 + (\tilde{y}''_i y_2 + y_i y_2 c_2) a_i \vec{\mathbf{b}}_5 + (\tilde{y}''_i y_2 + y_i y_2 c_2) \vec{\mathbf{b}}_{6,i} \}_{i \text{ labels } \in \Lambda} \end{aligned}$$

which is distributed as a semi-functional key of type 0R (which is appropriate for both games) where $y'_i = \tilde{y}'_i y_1 + y_i y_1 c_1$, and $y''_i = \tilde{y}''_i y_2 + y_i y_2 c_2$ which are independent uniformly distributed elements of $\mathbb{Z}_p$. Also, $\lambda'_i = \tilde{\lambda}'_i y_1 + \lambda_i y_1 c_1$ and $\lambda''_i = \tilde{\lambda}''_i y_2 + \lambda_i y_2 c_2$ 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_i 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. The following ciphertext is then constructed and provided:

$$\begin{aligned} & M(e(g,g)^\alpha)^s e(g,g)^{\tilde{\alpha}' \tilde{s}'} e(g, g^{y_1 c_1})^{\tilde{s}'}, \\ & \{ g^{s \vec{d}^*_1} \\ & g^{\tilde{s}' \vec{d}^*_3} g^{\tilde{s}' \vec{d}^*_5} T^{\tilde{s}' \vec{d}^*_1} \\ & g^{-s a_i \vec{d}^*_{2,i}} \\ & g^{-\tilde{s}' a_i \vec{d}^*_{4,i}} g^{-\tilde{s}' a_i \vec{d}^*_{6,i}} T^{-\tilde{s}' a_i \vec{d}^*_{2,i}} \}_{i \in S} \end{aligned}$$

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

$$\begin{aligned} & M e(g,g)^{\alpha s + (\tilde{\alpha}' y_1 + y_1^2 c_1) \tilde{s}' y_1^{-1}}, \\ & \{ (s + r\tilde{s}') \vec{\mathbf{b}}^*_1 - (s + r\tilde{s}') a_i \vec{\mathbf{b}}^*_{2,i} \\ & + \tilde{s}' y_1^{-1} \vec{\mathbf{b}}^*_3 - \tilde{s}' y_1^{-1} a_i \vec{\mathbf{b}}^*_{4,i} \\ & + y_2^{-1} \tilde{s}' \vec{\mathbf{b}}^*_5 - y_2^{-1} \tilde{s}' a_i \vec{\mathbf{b}}^*_{6,i} \}_{i \in S} \end{aligned}$$

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 13. *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. □

5 Concluding Remarks

We have presented a prime order KP-ABE scheme fully secure under the DLIN assumption. An interesting question for future work is whether the ciphertext sizes can be significantly reduced (our schemes have ciphertexts still growing linearly with the number of attribute-uses, and therefore have a dependence on the set of allowed policies).

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] 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.

[7] 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.

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

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

[10] 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.

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

[12] 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.

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

[14] 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.

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

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

[17] 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.

[18] 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.

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

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

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

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

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

[24] 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.

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

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

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

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

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

[30] 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.

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

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

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

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