The Dissertation Committee for Allison Bishop Lewko
certifies that this is the approved version of the following dissertation:

Functional Encryption: New Proof Techniques and Advancing Capabilities

Committee:

Brent Waters, Supervisor

David Zuckerman

Vitaly Shmatikov

Adam Klivans

Amit Sahai

Dan Boneh

Functional Encryption: New Proof Techniques and Advancing Capabilities

by

Allison Bishop Lewko, A.B.

DISSERTATION

Presented to the Faculty of the Graduate School of

The University of Texas at Austin

in Partial Fulfillment

of the Requirements

for the Degree of

DOCTOR OF PHILOSOPHY

THE UNIVERSITY OF TEXAS AT AUSTIN

May 2012

Acknowledgments

I wish to thank my advisor, Brent Waters, for the time and energy he invested in our collaboration, as well as for his advice and encouragement. I wish to thank my collaborators for the works comprising this thesis: Brent Waters, Amit Sahai, Tatsuaki Okamoto, and Katsuyuki Takashima. I also wish to thank my husband, Mark Lewko, for his support.

Functional Encryption: New Proof Techniques and Advancing Capabilities

Publication No. _____

Allison Bishop Lewko, Ph.D.
The University of Texas at Austin, 2012

Supervisor: Brent Waters

We develop the dual system encryption methodology to provide fully secure functional encryption systems. We introduce new proof techniques and explore their applications, resulting in systems that advance the state of the art in terms of functionality, security, and efficiency. Our approach constructs versatile tools for adapting the dual system encryption methodology to new functionalities and efficiency goals. As particular demonstrations of our techniques, we obtain fully secure ciphertext-policy attribute-based encryption systems in the single authority and decentralized settings. Our work has provided the first fully secure attribute-based encryption schemes as well as the first decentralized schemes achieving desired levels of flexibility.

Table of Contents

List of Figures

Chapter 1

Introduction

A traditional public key encryption system is designed to provide secure communication between two parties over a public channel. Though the security guarantees that can be achieved in this setting are very strong (e.g. indistinguishability under chosen plaintext attack or chosen ciphertext attack), the functionality provided is rather limited. For example, the encryptor must know the (single) intended recipient and retrieve his public key before encrypting. If there are many recipients, the encryptor must produce a separate ciphertext for each, which may be very inefficient.

Functional encryption presents a new vision for public key cryptosystems that provide a strong combination of flexibility, efficiency, and security. In a functional encryption scheme, ciphertexts are associated with descriptive values x, secret keys are associated with descriptive values y, and a function $f(x, y)$ determines what a user with a key for value y should learn from a ciphertext with value x. One well-studied example of functional encryption is attribute-based encryption (ABE), first introduced by Sahai and Waters in [63], in which ciphertexts and keys are associated with access policies over attributes and subsets of attributes. A key will decrypt a ciphertext if and only

if the associated set of attributes satisfies the associated access policy. There are two types of ABE systems: Ciphertext-Policy ABE (CP-ABE), where ciphertexts are associated with access policies and keys are associated with sets of attributes, and Key-Policy ABE (KP-ABE), where keys are associated with access policies and ciphertexts are associated with sets of attributes.

In this work, we will present new techniques for constructing provably secure functional encryption systems. We will focus on CP-ABE schemes as our application. Our goal is to advance the state of the art in achieving strong security guarantees for flexible, efficient systems allowing suitably expressive access policies and providing new capabilities. Before summarizing our results, we provide relevant context on the history of functional encryption and prior work in this area.

1.1 A Brief History of Functional Encryption

The roots of functional encryption can be traced back to Identity-Based Encryption (IBE), first proposed by Shamir [64]. An identity-based encryption scheme is designed to allow any string to function as a "public key," instead of requiring public keys to be constructed in tandem with secret keys. For example, this allows a user to send an encrypted message to a recipient described by an email address, without requiring the recipient to have established a public key. Secret keys corresponding to strings (referred to as "identities") must be obtained from a central authority who holds a master secret key. If we wish to impose a hierarchical structure on keys, we can generalize identity-based

encryption to hierarchical identity-based encryption (HIBE), in which a user can delegate secret keys to her subordinates.

There are innate challenges to proving security for a functionality like IBE which involves producing many secret keys for individual users from a single master secret key. It is not sufficient to prevent one user from maliciously using his own secret key to decrypt a ciphertext intended for another user - a strong notion of security must consider collusion attacks, where a group of users may collude in an attempt to decrypt a ciphertext encrypted to an identity outside of the group. To model such attacks, we consider an adversary who can collect many secret keys, choosing the corresponding identities adaptively. At some point, the adversary must decide on an identity to attack (for which it has not collected a secret key), and it may then continue collecting keys for any other identities. This requires security reductions to balance two competing goals: the simulator must be powerful enough to provide the attacker with the many keys that it adaptively requests, but it must also lack some critical knowledge that it can gain from the attacker's success.

The first security proofs for IBE schemes relied on the random oracle model, a heuristic treating a fixed function as if it were truly random. The first security proofs given in the standard model (not relying on such a heuristic) achieved a weaker notion of security known as *selective security*. The selective security model constrains the attacker to choose the target of the attack before seeing the public parameters of the system. This is an unrealistic restriction, and so achieving selective security should be viewed as partial progress towards

achieving full security, rather than as an end in itself.

The notion of selective security is quite natural in the context of the proof technique employed by all early works in IBE and HIBE, known as *partitioning*. In a partitioning proof, the simulator sets up the system so that the space of all possible identities is partitioned into two pieces: identities for which the simulator can make secret keys and those for which it cannot. This provides a clear way to balance the competing goals of the simulator, who must ensure that all the adversary's key requests fall in the set of keys the simulator can make and that the attacked identity falls in the complement. This is much easier to do in the selective model, since the simulator knows ahead of time what the attacked identity is. This allows it to form a perfectly fitting partition, with the attacked identity being the only identity for which the simulator cannot make a secret key.

The need for selectivity to obtain an IBE security proof in the standard model was overcome by Boneh and Boyen [11] and also by Waters [67]. The security proof in [67] arranges for the simulator to "guess" a partition and abort when the attacker violates its constraints. However, the richer structure of more advanced systems like HIBE and ABE appears to doom this approach to incur exponential loss, since one must guess a partition that respects the partial ordering induced by the powers allocated to the individual keys.

Meanwhile, progress for attributed-based encryption systems remained stalled at selective security in the standard model. Following the introduction of attribute-based systems by Sahai and Waters in [63], the subsequent works

of [24, 38, 39, 62, 69] provided security proofs for various kinds of ABE systems only in the selective model[1].

Motivated by the relative stagnation in proof methodology for functional encryption systems, Waters introduced the dual system encryption methodology [68]. His initial work provided fully secure and efficient IBE and HIBE systems under standard assumptions. In [50], we provided a more elegant realization of dual system encryption allowing for further efficiency improvements in the context of HIBE. In [48], we obtained the first fully secure ABE systems in the standard model by extending our dual system encryption methodology. In a close follow-up work, Okamoto and Takashima [60] obtained similar results relying on the simple and relatively standard Decisional Linear Assumption (DLIN). In further works [49, 51–53], we continued expanding and enhancing the dual system encryption methodology to provide stronger security guarantees and a growing variety of advanced functionalities, including a multi-authority ABE system allowing the same access structures as state of the art single authority systems.

A Contextual Note: In the works referenced above, we developed the dual system encryption methodology in the context of composite order bilinear groups, relying on instances and close variants of the General Subgroup Decision Assumption [8]. Okamoto and Takashima developed the framework

[1] [9] relied on the generic group heuristic instead. This should be considered as a weaker guarantee than selective security.

of dual pairing vector spaces in prime order bilinear groups [58, 59] and observed that it could be used to implement the same proof techniques under the more standard Decisional Linear Assumption [60, 61]. Working in prime order groups is also advantageous for efficiency reasons, since the group orders can be taken to be much smaller and hence pairing computations can be much faster. In [46], we further developed the connection between the dual pairing vector space framework and our prior approach in the composite order setting. The collective result of these works is a functional understanding of how to transfer dual system encryption proofs between the composite order and prime order settings.

In this work, we have chosen to work wholly in the prime order setting. Thus, the constructions and proofs we give here do not precisely match those given in the works cited above. Instead, they can be viewed as the result of translating those composite order schemes and proofs into the prime order setting using the tools developed in [46, 58–60]. However, we will present our tools and results in the prime order setting here from the ground up, and there is no need for the reader to reference the prior composite order analogs.

1.2 Summary of Our Results

In this work, we develop several fully secure CP-ABE systems. We will start with the most basic application of the dual system encryption methodology and present a full security proof for a CP-ABE scheme that holds under the restriction that attributes are only used in the description of the access

policy. (For more details on the precise meaning of this restriction, see Chapter 5.) Our proof relies only on the Decisional Linear Assumption. We also describe an encoding technique that can be used to allow re-use of attributes, though this may produce a significant loss in efficiency. This should be viewed as a combination of the results we presented in [46, 48].

We next present a new scheme and accompanying proof of full security that does not place any restriction on the use of attributes. The first two schemes we present here shed light on the value of selective security as stepping stone to full security for ABE schemes, as the constructions of the schemes bear a very close resemblance to the selectively secure ABE constructions in [39, 69], and the proof strategy for the second scheme directly incorporates selective security proof techniques as building blocks that can be leveraged within the dual system encryption methodology. In this way, we demonstrate how dual system encryption can be viewed as a mechanism to "boost" selective security to full security (though the details of the arguments must be adapted to individual constructions). This result also appears in [53]. This approach obtains much better efficiency, but comes at the cost of relying on a "q-type" assumption (i.e. an assumption parameterized by a value depending on the adversary's behavior). This is a feature inherited from the state of the art selectively secure CP-ABE scheme in [69].

In our first two CP-ABE constructions, private keys are issued by one central authority that must be in a position to verify all the attributes or credentials it issues for each user in the system. However, in some applications a

party will want to share data according to a policy written over attributes or credentials issued across different trust domains and organizations. Motivated by this, we next provide a multi-authority CP-ABE system. In our system, any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. A party can simply act as an authority by creating a public key and issuing private keys to different users that reflect their attributes. Different authorities need not even be aware of each other. We use the concept of global identifiers to link private keys together that were issued to the same user by different authorities. A user can encrypt data according to access policies that mix attributes issued from any chosen set of authorities. Most notably, our decentralized system retains strong security guarantees while avoiding placing absolute trust in any single designated entity. The decentralized nature of our system gives rise to new technical challenges both in designing the construction and in proving security. We discuss these in detail when we present our solution in Chapter 7. This should be viewed as a combination of the results we presented in [46, 51].

1.3 Related Work

Identity-based encryption was first constructed by Boneh and Franklin [14] and Cocks [25]. Both security proofs relied on the random oracle heuristic. Constructions with proofs in the standard model were then presented by Canetti, Halevi, and Katz [20] and Boneh and Boyen [10], but these proofs

provided only selective security.

The notion of Hierarchical Identity-Based Encryption (HIBE) was introduced by Horwitz and Lynn [42]. In a HIBE system, user's identities are arranged in a hierarchy, and a superior can use his own secret key to issue a secret key to any of his subordinates. Essentially, this extends the functionality of an IBE system to include key delegation. The first construction of HIBE was provided by Gentry and Silverberg [36], who proved security in the random oracle model. Selectively secure constructions in the standard model were then given in the work of Canetti, Halevi, and Katz [20], Boneh and Boyen [10], and Boneh, Boyen, and Goh [12].

Gentry [32] presented a security proof for an IBE system outside of the partitioning paradigm, but at the cost of employing a non-standard complexity assumption parameterized by the number of key queries made by the adversary (a "q-type" assumption). These techniques were then extended by Gentry and Halevi [34] to provide the first fully secure HIBE system to allow a hierarchy of polynomial depth (the exponential degradation of prior full security reductions limited the security proofs applicable to only constant depth hierarchies).

Chase [22] presented a "multi-authority" attribute-based encryption scheme, allowing secret keys for different attributes to be dispensed by different authorities. Her work introduced the concept of using a global identifier as a mechanism for tying users' keys together. Her system relied on a central authority and was limited to expressing a strict"AND" policy over a *pre-determined* set of authorities. Chase and Chow [23] showed how to remove

the central authority using a distributed PRF; however, the same limitations of an AND policy of a determined set of authorities remained.

Other forms of functional encryption have also been proposed. Most notably, Katz, Sahai, and Waters proposed predicate encryption [44] and presented a selectively secure construction for inner product predicates. This is a less expressive functionality than the access policies achieved for ABE, but the goal of such systems is to provide security not only for the underlying message, but also to hide the access policy itself, which is not a feature that is considered in the ABE setting.

1.4 Organization

In Chapter 2, we provide an overview of the core concepts of the dual system encryption methodology. In Chapter 3, we present tools for implementing dual system encryption in the setting of prime order bilinear groups. In Chapter 4, we provide background on single and multi-authority CP-ABE systems, giving formal definitions for their functionality and security. In Chapter 5, we construct our simplest CP-ABE scheme and prove its full security. In Chapter 6, we provide a more efficient scheme and prove its full security. In Chapter 7, we provide a multi-authority scheme and prove its full security. In Chapter 8, we discuss additional work and further directions.

Chapter 2

Conceptual Overview of Dual System Encryption

Dual System Encryption is a proof methodology that we have designed to address the inherent challenges in proving security for systems with advanced functionalities against collusion attacks. In the case of ciphertext-policy attribute-based encryption, an adversary may gather secret keys for many different users, each key corresponding to a set of attributes. We do not want to allow the adversary to learn a message encrypted under an access policy which is not satisfied by any of the individual keys he has collected. In other words, a colluding set of malicious users should not be able to combine their secret keys to decrypt a ciphertext that none of them are individually authorized to decrypt. It is crucial to note that the *union* of all their attributes might satisfy the policy, but as long as no single user does, the message should remain protected. In this sense, we expect a secure CP-ABE scheme to enforce that attributes are not transferable among users.

The formal definition of full security we employ for CP-ABE systems is given in Chapter 4. We briefly summarize its relevant features for the purposes of our discussion here. CP-ABE security is formulated as a game

between a challenger and an attacker: the challenger sets up the CP-ABE system and gives the attacker only the public parameters. The attacker then adaptively queries the challenger for secret keys corresponding to attribute sets of its choice. At some point, it declares two messages and an access policy it wishes to be challenged on: this access policy must not be satisfied by any of the individual keys it has collected so far. The challenger randomly chooses one of the two declared messages and encrypts it under the declared access policy. It gives the resulting ciphertext to the attacker. The attacker may then continue to request keys, except for those authorized to decrypt the ciphertext. Finally, it must guess which of the two messages was encrypted. We say that a system is *secure* when any polynomial time attacker guesses correctly only with probability negligibly close to $\frac{1}{2}$.

As previously noted, this security definition places a heavy burden on a reduction, as a simulator must both answer the attacker's many key queries and leverage the attacker's success. Since trying to anticipate any structure in the attacker's queries appears to be a losing battle (unless one works in the selective security model), dual system encryption is designed to prepare a simulator to answer *any* key query. To explain how this can remain consistent with the simulator's ability to leverage the attacker's success, we must first describe the dual system encryption paradigm.

Normal and Semi-functional Ciphertexts/Keys In a dual system, there are two forms of keys and ciphertexts: normal and semi-functional. Semi-

functional ciphertexts and keys are not used in the real system, they are only used in the security proof. A normal key can decrypt normal or semi-functional ciphertexts, and a normal ciphertext can be decrypted by normal or semi-functional keys. However, when a semi-functional key is used to decrypt a semi-functional ciphertext, decryption fails (with high probability). More specifically, the semi-functional components of the key and ciphertext will interact to mask the blinding factor by an additional random term. Security for dual systems is proved using a sequence of games which are shown to be indistinguishable. The first game is the real security game (with normal ciphertext and keys). In the next game, the ciphertext is semi-functional, while all the keys are normal. For an attacker that makes Q key requests, games 1 through Q follow. In game k, the first k keys are semi-functional while the remaining keys are normal. In game Q, all the keys and the challenge ciphertext given to the attacker are semi-functional. Hence none of the given keys are useful for decrypting the challenge ciphertext. At this point, proving security becomes relatively easy, as we have reduced the burden on the simulator who no longer has to give out functioning keys and ciphertexts.

Nominal Semi-functionality The most critical step of the hybrid proof is when a key turns semi-functional: at this point, we must leverage the fact that the key is not authorized to decrypt the (now semi-functional) challenge ciphertext in order to argue that the attacker cannot detect the change in the key. However, since we are not imposing a partition on the simulator, there

is no constraint preventing the simulator itself from creating a key that is authorized to decrypt and testing the nature of the key for itself by attempting to decrypt the semi-functional ciphertext. To resolve this potential paradox, we design our system so that such a test decryption would *unconditionally succeed*.

We introduce a variant of semi-functionality which we call *nominal* semi-functionality. A nominal semi-functional key and ciphertext pair is semi-functional in name only, meaning that they are individually distributed like semi-functional keys, but are actually correlated to each other so that the interaction of their semi-functional components results in cancelation during decryption, and hence decryption remains successful. If the simulator attempts to answer its own question by creating the k^{th} key and challenge ciphertext for an attribute set and access policy that is satisfied, the created pair will be nominally semi-functional and hence test decrypting will not distinguish the nature of the key. We note that the first introduction of dual system encryption in [68] took the opposite approach, arranging for test decryptions by the simulator to unconditionally fail by the use of "tags" that also caused a negligible decryption error and other undesirable structural features. We introduced the concept of nominal semi-functionality in [50] as a cleaner, more flexible alternative to tags.

Of course, nominal semi-functionality is only useful as a stepping stone to regular semi-functionality, since the primary goal is to arrive at a game where the simulator is no longer burdened by the need to produce keys and

ciphertexts that function properly together. One way of getting from nominal semi-functionality to regular semi-functionality is via an information-theoretic argument, leveraging the fact that the attacker is not allowed to ask for a key that is authorized to decrypt the ciphertext. This is the path that we took in the first works employing nominality and has been followed by all of the works since [48–52, 60], except for our most recent work in [53]. The new path we designed in [53] instead relies on a computational assumption to transition from nominal semi-functionality to regular semi-functionality.

Structure of Semi-functional Space Before describing the main ideas behind these two approaches, we first develop the high-level principles underpinning the design of semi-functional objects. We think of the normal keys and ciphertexts as residing in a "normal space", while semi-functional keys and ciphertexts additionally have components in a "semi-functional space". These two spaces should be constructed so that normal components do not interact with semi-functional components and vice versa - one should think of the normal space and the semi-functional space as being "orthogonal" to each other.

In essence, we will typically design the semi-functional components of keys and ciphertexts to form a parallel copy of the normal system residing in semi-functional space. The semi-functional copy of the system has the advantage of not being tied by public parameters, since the published public parameters will only reside in the normal space. The phantom parameters

in the semi-functional space (that are uncorrelated from the published public parameters) can then be utilized in multiple ways. We first describe the information-theoretic approach we introduced in [48, 50].

Information-theoretic Arguments for Hiding Nominality We imagine for simplicity that the attacker makes only one key request. To produce a ciphertext and key pair that is either semi-functional and normal respectively or nominally semi-functional, the simulator will create a parallel copy of the system in the semi-functional space. If semi-functional components are present on the key, they will obey the proper distribution defined by the unpublished, freshly random parameters in the semi-functional space and allow decryption to succeed for an authorized key.

However, if one is given only a single ciphertext and a single *unauthorized* key, the additional randomness supplied by the unknown semi-functional parameters can function to information-theoretically hide this correlation. Arguing this in the CP-ABE setting leads one to make an additional restriction that attributes are only used once in the LSSS matrices specifying access policies. This technical restriction is required to enforce that there is a fresh random parameter available to hide each unauthorized share.

Since entropy here is a scarce resource, we also cannot afford for the same semi-functional parameters to be involved in multiple keys. This is rather easily handled by exploiting the hybrid organization of dual system encryption arguments - since we turn one key semi-functional at a time, we can design a

key isolation mechanism which allows us to limit the effect of some or all of the relevant semi-functional space parameters to a single key at a time. In the case of CP-ABE, we introduce two types of semi-functional keys to execute this. Our techniques here exploit the structure of keys, which have a "header" portion introducing the random values that tie the key together and then have individual pieces corresponding to each attribute. When a key first turns semi-functional, it will have semi-functional components on all of these pieces, but then it will transition to having a random semi-functional component only on the header portion. This means that information about the semi-functional spaces attached to the attribute pieces is no longer leaked, thereby preserving this entropy for use in turning the next key to be semi-functional. We can conceptualize the semi-functional space attached to the header piece as a "permanent" semi-functional space (where all semi-functional keys will have components), and the semi-functional spaces attached to the attribute pieces as "temporary" semi-functional spaces (where at most one semi-functional key will have components at any point in the hybrid argument).

To adapt this same proof strategy to a multi-authority ABE scheme, we face an additional technical challenge. Namely, we can no longer tie keys together with common secret randomness, since the different pieces of the key are produced by different authorities who do not coordinate. We will address this by assuming users have global identifiers and introducing a hash function (modeled as a random oracle) to map these identifiers to group elements. These group elements can be computed by each authority and will form the

common link between the different attributes in a user's key. This requires a re-design of our construction and our key isolation mechanism, since we no longer have a header piece of each key to use as a structural linchpin and as a storage location for permanent semi-functionality. Instead, we design the semi-functional spaces attached to each piece of the key to have "permanent" and "temporary" portions - when a key first becomes semi-functional, its semi-functional components will reside only in the temporary portions of semi-functional space, where the information-theoretic argument takes place. It will then switch to having semi-functional components residing only in the permanent portions of semi-functional space, thereby releasing the entropy available in temporary semi-functional space to be used to hide nominality for the next key as it becomes semi-functional.

Once we have full security proofs for systems where attributes can only be used once in each access policy, we can extend this to systems allowing reuse of attributes by setting a fixed bound M on the maximum number of times an attribute may be used and having separate parameters for each use. This scales the size of the public parameters by M, as well as the size of secret keys for CP-ABE systems[1]. This approach may incur a significant loss in efficiency.

A New Computational Approach for Addressing Nominality Our next observation is motivated by the intuition that the information-theoretic step of the above dual system proof strategy is ceding too much ground to the

[1]For KP-ABE systems, it is the ciphertext size that will grow multiplicatively with M.

attacker, since a computational argument would suffice. In fact, we are able to resurrect the earlier selective proof techniques inside the framework of dual system encryption in order to retake ground and obtain a wholly computational proof of full security.

As we have discussed above, dual system encryption is typically implemented by designing a "semi-functional space" where semi-functional components of keys and ciphertexts will behave like a parallel copy of the normal components of the system, except divorced from the public parameters. Instead of conceptualizing the initially hidden parameters in the semi-functional space as a source of entropy, we now think of this as a mechanism for allowing *delayed parameters* in the semi-functional space, meaning that relevant variables can be defined later in the simulation instead of needing to be fixed in the setup phase. We will still additionally use a mechanism for *key isolation*, meaning that some or all of the semi-functional parameters will only be relevant to the distribution of a single semi-functional key at a time.

In combination, these two mechanisms mean that the semi-functional space has its own fresh parameters that can be decided on the fly by the simulator when they become relevant, and they are only relevant for the semi-functional ciphertext and a single semi-functional key. We now observe that these mechanisms can be used to implement prior techniques for selective security proofs, without needing to impose the selective restriction on the attacker.

To explain this more precisely, we consider the critical step in the hy-

brid security proof when a particular key becomes semi-functional. We conceptualize the unpublished semi-functional parameters as being defined belatedly when the simulator first issues either the key in question or the semi-functional ciphertext. If the ciphertext is issued first, then the simulator learns the challenge policy *before* defining the delayed semi-functional parameters - this is closely analogous to the setting of selective security for a CP-ABE system. If the key is issued first, then the simulator learns the relevant set of attributes before defining the delayed semi-functional parameters, and this is closely analogous to the setting of selective security for a KP-ABE system. This provides us with an opportunity to combine the techniques used to prove selective security for both CP-ABE and KP-ABE systems with the dual system encryption methodology in order to obtain a new proof of full security while maintaining the efficiency of selectively secure systems.

Chapter 3

Tools for Implementing the Dual System Encryption Methodology in Prime Order Bilinear Groups

We now build some general tools for implementing dual system encryption proofs in the context of prime order bilinear groups. Bilinear groups are a natural setting for CP-ABE systems and dual system encryption proofs, since they come equipped with nice randomization properties that can be used for collusion resistance and provide convenient instantiations of orthogonal "spaces." Constructing expressive CP-ABE systems or implementing dual system encryption techniques in any other setting (lattices, for example), is currently an open problem.

Composite order bilinear groups come equipped with prime order subgroups that are mutually orthogonal and thus can directly serve as orthogonal spaces. This is why many of the first works using dual system encryption worked in this setting. In this work, however, we instead use the dual pairing vector space framework developed by Okamoto and Takashima to instantiate our orthogonal spaces in prime order bilinear groups.

3.1 Prime Order Bilinear Groups

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

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

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

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

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

We define e_n to denote the product of the componentwise pairings (the product is taken in the group G_T):

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

Here, the dot product is taken modulo p.

Decisional Linear Assumption The main complexity assumption we will rely on in prime order bilinear groups is the Decisional Linear Assumption, introduced in [13]. To define this formally, we let $\mathcal{G}$ denote a group generation

algorithm, which takes in a security parameter λ and outputs a bilinear group G of order p. We write $(p, G, G_T, e) \xleftarrow{R} \mathcal{G}$ to denote the generation of a group by running $\mathcal{G}$. Below, we also use the notation $x \xleftarrow{R} S$ to denote a uniformly random sample x taken from a finite set S.

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

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

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

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

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

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

is negligible in the security parameter λ.

3.2 Dual Pairing Vector Spaces

We will employ the concept of dual pairing vector spaces from [58, 59]. For a fixed (constant) dimension n, we will choose two random bases $\mathbb{B} := (\vec{b}_1, \ldots, \vec{b}_n)$ and $\mathbb{B}^* := (\vec{b}_1^*, \ldots, \vec{b}_n^*)$ of $\mathbb{Z}_p^n$, subject to the constraint that they are "dual orthonormal", meaning that

$$\vec{b}_i \cdot \vec{b}_j^* = 0 \ (\mathrm{mod}\, p),$$

whenever $i \neq j$, and

$$\vec{b}_i \cdot \vec{b}_i^* = 1 \,(\mathrm{mod}\, p)$$

for all i. (Some prior works have allowed $\vec{b}_i \cdot \vec{b}_i^* = \psi$ for $\psi \neq 1$, but we restrict to 1 for convenience.) For a generator $g \in G$, we note that

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

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

We note that choosing random dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ can equivalently be thought of as choosing a random basis $\mathbb{B}$, choosing a vector $\vec{b}_1^*$ so that it is orthogonal to $\vec{b}_2, \ldots, \vec{b}_n$ and satisfies $1 = \vec{b}_1 \cdot \vec{b}_1^*$, and then choosing $\vec{b}_2^*$ so that it is orthogonal to $\vec{b}_1, \vec{b}_3, \ldots, \vec{b}_n$, and satisfies $1 = \vec{b}_2 \cdot \vec{b}_2^*$, and so on. We will later use the notation $(\mathbb{D}, \mathbb{D}^*)$ and $\vec{d}_1, \ldots$, etc. to also denote dual orthonormal bases and their vectors. This is because we will sometimes be handling more than one pair of dual orthonormal bases at a time, and we use different notation to avoid confusing them.

By employing dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ in the exponents of our ciphertexts and keys respectively, we can naturally partition these basis vectors into disjoint sets spanning the "normal space" and the "semi-functional space." For instance, for a bases pair of dimension 3, we may declare $\vec{b}_1, \vec{b}_2$ to span the normal space on the ciphertext side and $\vec{b}_3$ to span the semi-functional space on the ciphertext side, while $\vec{b}_1^*, \vec{b}_2^*$ span the normal space on the key side and $\vec{b}_3^*$ spans the semi-functional space on the key side. We note that the normal

space on the ciphertext side will be orthogonal to the semi-functional space on the key side, and vice versa.

3.2.1 Parameter Hiding in Dual Orthonormal Bases

It will be crucial in our security proofs that when one publishes parameters for the normal space and produces normal keys, one only information-theoretically reveals the bases vectors for the normal spaces, while some information about the remaining bases vectors for the semi-functional spaces remains hidden.

More generally, we observe a mechanism for *parameter hiding* using dual orthonormal bases: one can generate a random pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ for $\mathbb{Z}_p^n$, apply an invertible change of basis matrix A to a subset of these basis vectors, and produce a new pair of dual orthonormal bases which is also randomly distributed, *independently of* A. This allows us to *hide* a random matrix A. We formulate this precisely below.

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

To describe this formally, we let $m \leq n$ be fixed positive integers and

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

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

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

denotes the set of dual orthonormal bases.

Lemma 2. For any fixed positive integers $m \leq n$, any fixed invertible $A \in \mathbb{Z}_p^{m \times m}$ and set $S_m \subseteq [n]$ of size m, if $(\mathbb{B}, \mathbb{B}^*) \xleftarrow{R} Dual(\mathbb{Z}_p^n)$, then $(\mathbb{B}_A, \mathbb{B}_A^*)$ is also distributed as a random sample from $Dual(\mathbb{Z}_p^n)$. In particular, the distribution of $(\mathbb{B}_A, \mathbb{B}_A^*)$ is independent of A.

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

3.2.2 The Subspace Assumption

We now describe the "Subspace Assumption," which is implied by the decisional linear assumption. We introduced this in [46] to help clarify how DLIN allows one to expand/contract spaces in the exponent in the dual pairing vector space framework. It is based on the observation that if we fix a pair $(\mathbb{B}, \mathbb{B}^*)$ of dual orthonormal bases and one is given $g^{\vec{v}}$ say, then one cannot tell if $\vec{v}$ is in the span of $\vec{b}_1^*, \vec{b}_2^*$ or the larger span of $\vec{b}_1^*, \vec{b}_2^*, \vec{b}_3^*$ when one is *not* given $g^{\vec{b}_3}$ (though one can be given $g^{\vec{w}}$ for $\vec{w}$ in the span of $\vec{b}_1, \vec{b}_2, \vec{b}_3$, for example). At its core, the subspace assumption translates the DLIN assumption into a change from a 2-dimensional space in the exponent to a 3-dimensional one, and then replicates this in parallel both within a bases pair and over many bases pairs. In our security proofs, we will employ the subspace assumption

to move ciphertext and key components in and out of various portions of the semi-functional space. We will also use it to transition to the final security game, where the semi-functional ciphertext becomes an encryption of a random message.

The statement of the subspace assumption we give here is parameterized by several values which can be set to be any positive integers. We let the parameter m denote the number of pairs of dual orthonormal bases involved. Each basis pair has its own dimension n_i and its own parameter $k_i \leq n_i/3$ (this parameter describes how many 2-dimensional subspaces within each basis are expanding to be 3-dimensional).

The Subspace Assumption will consider m bases pairs chosen independently at random. These bases pairs will be denoted by $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_m, \mathbb{B}_m^*)$, and the vectors comprising each $(\mathbb{B}_i, \mathbb{B}_i^*)$ will be denoted by $\vec{b}_{1,i}, \ldots, \vec{b}_{n_i,i}$ and $\vec{b}^*_{1,i}, \ldots, \vec{b}^*_{n_i,i}$. For a simpler statement of the subspace assumption considering only one bases pair, see [46]. Compared to the statement in [46], we have also added a few extra terms that will be useful in the security proof for our multi-authority CP-ABE system. We have also chosen to directly reveal the basis vectors that are not involved in the expanding subspaces, whereas the original statement in [46] left these in the exponent. This change is not necessary for any of our proofs here, but we feel it is worthwhile to point out that giving out these basis vectors directly does not affect the reduction from the decisional linear assumption.

Definition 3. (The Subspace Assumption) Given a group generator $\mathcal{G}$, we define

the following distribution:

$$\mathbb{G} := (p, G, G_T, e) \xleftarrow{R} \mathcal{G},\ g \xleftarrow{R} G,\ \eta, \beta, \tau_1, \tau_2, \tau_3, \mu_1, \mu_2, \mu_3 \xleftarrow{R} \mathbb{Z}_p,$$

$$(\mathbb{B}_1, \mathbb{B}_1^*) \xleftarrow{R} Dual(\mathbb{Z}_p^{n_1}), \ldots, (\mathbb{B}_m, \mathbb{B}_m^*) \xleftarrow{R} Dual(\mathbb{Z}_p^{n_m}),$$

$$U_{1,i} := g^{\mu_1 \vec{b}_{1,i} + \mu_2 \vec{b}_{k_i+1,i} + \mu_3 \vec{b}_{2k_i+1,i}},\ U_{2,i} := g^{\mu_1 \vec{b}_{2,i} + \mu_2 \vec{b}_{k_i+2,i} + \mu_3 \vec{b}_{2k_i+2,i}},$$

$$\ldots,\ U_{k_i,i} := g^{\mu_1 \vec{b}_{k_i,i} + \mu_2 \vec{b}_{2k_i,i} + \mu_3 \vec{b}_{3k_i,i}}\ \forall i \in [m],$$

$$V_{1,i} := g^{\tau_1 \eta \vec{b}^*_{1,i} + \tau_2 \beta \vec{b}^*_{k_i+1,i}},\ V_{2,i} := g^{\tau_1 \eta \vec{b}^*_{2,i} + \tau_2 \beta \vec{b}^*_{k_i+2,i}},$$

$$\ldots,\ V_{k_i,i} := g^{\tau_1 \eta \vec{b}^*_{k_i,i} + \tau_2 \beta \vec{b}^*_{2k_i,i}}\ \forall i \in [m],$$

$$W_{1,i} := g^{\tau_1 \eta \vec{b}^*_{1,i} + \tau_2 \beta \vec{b}^*_{k_i+1,i} + \tau_3 \vec{b}^*_{2k_i+1,i}},\ W_{2,i} := g^{\tau_1 \eta \vec{b}^*_{2,i} + \tau_2 \beta \vec{b}^*_{k_i+2,i} + \tau_3 \vec{b}^*_{2k_i+2,i}},$$

$$\ldots,\ W_{k_i,i} := g^{\tau_1 \eta \vec{b}^*_{k_i,i} + \tau_2 \beta \vec{b}^*_{2k_i,i} + \tau_3 \vec{b}^*_{3k_i,i}}\ \forall i \in [m],$$

$$D := \big(\mathbb{G}, g, g^{\eta}, g^{\beta}, g^{\tau_1 \eta}, g^{\tau_2 \beta}, \{g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, \ldots, g^{\vec{b}_{2k_i,i}}, \vec{b}_{3k_i+1,i}, \ldots, \vec{b}_{n_i,i},$$

$$g^{\eta \vec{b}^*_{1,i}}, \ldots, g^{\eta \vec{b}^*_{k_i,i}}, g^{\beta \vec{b}^*_{k_i+1,i}}, \ldots, g^{\beta \vec{b}^*_{2k_i,i}}, g^{\vec{b}^*_{2k_i+1,i}}, \ldots, g^{\vec{b}^*_{3k_i,i}},$$

$$\vec{b}^*_{3k_i+1,i}, \ldots, \vec{b}^*_{n_i,i}, U_{1,i}, U_{2,i}, \ldots, U_{k_i,i}\}_{i=1}^m,\ \mu_3\big).$$

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

$$Adv_{\mathcal{G},\mathcal{A}} := |Pr\left[\mathcal{A}\left(D, \{V_{1,i}, \ldots, V_{k_i,i}\}_{i=1}^m\right) = 1\right] - Pr\left[\mathcal{A}\left(D, \{W_{1,i}, \ldots, W_{k_i,i}\}_{i=1}^m\right) = 1\right]|$$

is negligible in the security parameter λ.

We have included in D more terms than will be necessary for many applications of this assumption, and in what follows we will often omit those we do not need. In our proofs below, we will also ignore that the final vectors

are given directly and write all vectors as being given in the exponent for notational simplicity, since this suffices for our purposes.

We will work exclusively with the $k = 1$ and $k = 2$ cases. To help the reader see the main structure of this assumption through the burdensome notation, we include heuristic illustrations of the $m = 1$, $n = 3$, $k = 1$ and $m = 1$, $n = 6$, $k = 2$ cases below.

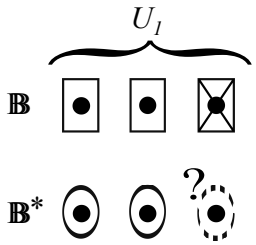

Figure 3.1: Subspace Assumption with $m = 1$, $n = 3$, $k = 1$

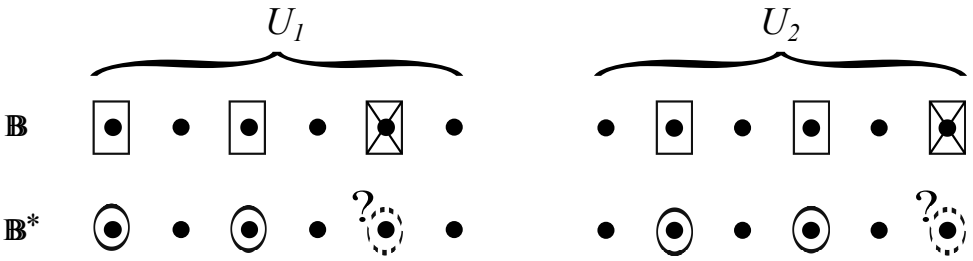

Figure 3.2: Subspace Assumption with $m = 1$, $n = 6$, $k = 2$

In these diagrams, the top rows illustrate the U terms, while the bottom rows illustrate the V, W terms. The solid ovals and rectangles indicate the presence of basis vectors. The crossed rectangles indicate basis elements of $\mathbb{B}$ which are present in U_1, U_2 but are not given out in isolation. The dotted ovals

adorned by question marks indicate the basis vectors whose presence depends on whether we consider the V's or the W's.

We now prove:

Lemma 4. If the decisional linear assumption holds for a group generator $\mathcal{G}$, then the subspace assumption stated in Definition 3 also holds for $\mathcal{G}$ (for any fixed values of $m, n_1 \geq 3k_1, \ldots, n_m \geq 3k_m$ that are polynomial in the security parameter).

Proof. We assume there exists a PPT algorithm $\mathcal{A}$ breaking the subspace assumption with non-negligible advantage (for some fixed positive integers $m, n_1, \ldots, n_m, k_1, \ldots, k_m$ satisfying $n_1 \geq 3k_1, \ldots, n_m \geq 3k_m$). We create a PPT algorithm $\mathcal{B}$ which breaks the decisional linear assumption with non-negligible advantage. $\mathcal{B}$ is given $g, f, v, f^{c_1}, v^{c_2}, T$, where T is either $g^{c_1+c_2}$ or $T = g^{c_1+c_2+w}$ is a uniformly random element of G. We let ℓ_f denote the discrete logarithm base g of f and ℓ_v denote the discrete logarithm base g of v, i.e. $f = g^{\ell_f}$ and $v = g^{\ell_v}$.

$\mathcal{B}$ simulates the subspace assumption for $\mathcal{A}$ as follows. $\mathcal{B}$ first (independently) samples random dual orthonormal bases $(\mathbb{D}_1, \mathbb{D}_1^*) \in Dual(Z_p^{n_1})$, ..., $(\mathbb{D}_m, \mathbb{D}_m^*) \in Dual(\mathbb{Z}_p^{n_m})$. $\mathcal{B}$ then implicitly sets $\eta = \ell_f$, $\beta = \ell_v$, and:

$$\eta\vec{b}^*_{1,i} = \vec{d}^*_{2k_i+1,i} + \ell_f\vec{d}^*_{1,i},\ \eta\vec{b}^*_{2,i} = \vec{d}^*_{2k_i+2,i} + \ell_f\vec{d}^*_{2,i},\ \ldots,\ \eta\vec{b}^*_{k_i,i} = \vec{d}^*_{3k_i,i} + \ell_f\vec{d}^*_{k_i,i},$$

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

$$\vec{b}^*_{2k_i+1,i} = \vec{d}^*_{2k_i+1,i},\ \ldots, \vec{b}^*_{n_i,i} = \vec{d}^*_{n_i,i}$$

for each i from 1 to m. In other words, $\mathcal{B}$ has set $\vec{b}^*_{1,i} = \eta^{-1}\vec{d}^*_{2k_i+1,i} + \vec{d}^*_{1,i}$ for example. We note that $\mathcal{B}$ can produce $g^\eta = f$ and $g^\beta = v$.

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

$$\vec{b}_{1,i} = \vec{d}_{1,i},\ \vec{b}_{2,i} = \vec{d}_{2,i},\ \ldots, \vec{b}_{2k_i,i} = \vec{d}_{2k_i,i},$$

$$\vec{b}_{2k_i+1,i} = \vec{d}_{2k_i+1,i} - \ell_f^{-1}\vec{d}_{1,i} - \ell_v^{-1}\vec{d}_{k_i+1,i},\ \ldots,\ \vec{b}_{3k_i,i} = \vec{d}_{3k_i,i} - \ell_f^{-1}\vec{d}_{k_i,i} - \ell_v^{-1}\vec{d}_{2k_i,i},$$

$$\vec{b}_{3k_i+1,i} = \vec{d}_{3k_i+1,i},\ \ldots,\ \vec{b}_{n_i,i} = \vec{d}_{n_i,i}.$$

We note that each pair $(\mathbb{B}_1, \mathbb{B}^*_1), \ldots, (\mathbb{B}_m, \mathbb{B}^*_m)$ is indeed a pair of dual orthonormal bases. We also note that $\mathcal{B}$ can produce all of $g^{\eta\vec{b}^*_{1,i}}, \ldots, g^{\eta\vec{b}^*_{k_i,i}}$, $g^{\beta\vec{b}^*_{k_i+1,i}}, \ldots, g^{\beta\vec{b}^*_{2k_i,i}}$, $g^{\vec{b}^*_{2k_i+1,i}}, \ldots, g^{\vec{b}^*_{3k_i,i}}$, $\vec{b}^*_{3k_i+1,i}, \ldots, \vec{b}^*_{n_i,i}$, $g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{2k_i,i}}$, and $\vec{b}_{3k_i+1,i}$, $\ldots$, $\vec{b}_{n_i,i}$ for each i from 1 to m, but *cannot produce* $g^{\vec{b}_{2k_i+1,i}}, \ldots, g^{\vec{b}_{3k_i,i}}$.

We now observe that $\eta = \ell_f$, $\beta = \ell_v$, $\{\vec{b}_{1,i}, \ldots, \vec{b}_{n_i,i}\}$ and $\{\vec{b}^*_{1,i}, \ldots, \vec{b}^*_{n_i,i}\}$ are properly distributed. Indeed, this follows from the fact that ℓ_f, ℓ_v are randomly distributed and from Lemma 2, since we have applied a change of basis to each $(\mathbb{D}_i, \mathbb{D}^*_i)$.

Now $\mathcal{B}$ creates $U_{1,i}, \ldots, U_{k_i,i}$ for each i as follows. It chooses random values $\mu'_1, \mu'_2, \mu'_3 \in \mathbb{Z}_p$. It sets:

$$U_{1,i} = g^{\mu'_1\vec{b}_{1,i} + \mu'_2\vec{b}_{k_i+1,i} + \mu'_3\vec{d}_{2k_i+1,i}}.$$

We note that

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

In other words, $\mathcal{B}$ has implicitly set $\mu_1 = \mu'_1 + \ell_f^{-1}\mu'_3$, $\mu_2 = \mu'_2 + \ell_v^{-1}\mu'_3$, and $\mu_3 = \mu'_3$. We note that these values are uniformly random, and μ_3 is known to $\mathcal{B}$. $\mathcal{B}$ can then similarly form $U_{2,i}, \ldots, U_{k_i,i}$ as:

$$U_{2,i} = g^{\mu'_1\vec{b}_{2,i}+\mu'_2\vec{b}_{k_i+2,i}+\mu'_3\vec{d}_{2k_i+2,i}}, \ \ldots, \ U_{k_i,i} = g^{\mu'_1\vec{b}_{k_i,i}+\mu'_2\vec{b}_{2k_i,i}+\mu'_3\vec{d}_{3k_i,i}}.$$

$\mathcal{B}$ then implicitly sets $\tau_1 = c_1$ and $\tau_2 = c_2$. This allows $\mathcal{B}$ to produce $g^{\eta\tau_1} = f^{c_1}$ and $g^{\beta\tau_2} = v^{c_2}$. We note that for each i from 1 to m:

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

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

$\mathcal{B}$ computes for each i:

$$T_{1,i} = T^{\vec{d}^*_{2k_i+1,i}} \left(f^{c_1}\right)^{\vec{d}^*_{1,i}} \left(v^{c_2}\right)^{\vec{d}^*_{k_i+1,i}}, \ \ldots, \ T_{k_i,i} = T^{\vec{d}^*_{3k_i,i}} \left(f^{c_1}\right)^{\vec{d}^*_{k_i,i}} \left(v^{c_2}\right)^{\vec{d}^*_{2k_i,i}}.$$

If $T = g^{c_1+c_2}$, then these are distributed as $V_{1,i}, \ldots, V_{k_i,i}$. If $T = g^{c_1+c_2+w}$, then these are distributed as $W_{1,i}, \ldots, W_{k_i,i}$, with τ_3 implicitly set to w.

$\mathcal{B}$ gives

$$D := \big(g, g^{\eta}, g^{\beta}, g^{\eta\tau_1}, g^{\beta\tau_2}, \{g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, \ldots, g^{\vec{b}_{2k_i,i}}, \vec{b}_{3k_i+1,i}, \ldots, \vec{b}_{n_i,i}, g^{\eta\vec{b}^*_{1,i}}, \ldots, g^{\eta\vec{b}^*_{k_i,i}},$$

$$g^{\beta\vec{b}^*_{k_i+1,i}}, \ldots, g^{\beta\vec{b}^*_{2k_i,i}}, g^{\vec{b}^*_{2k_i+1,i}}, \ldots, g^{\vec{b}^*_{3k_i,i}}, \vec{b}^*_{3k_i+1,i}, \ldots, \vec{b}^*_{n_i,i}, U_{1,i}, U_{2,i}, \ldots, U_{k_i,i}\}_{i=1}^m, \mu_3\big)$$

to $\mathcal{A}$, along with $\{T_{1,i} \ldots, T_{k_i,i}\}_{i=1}^{m}$. $\mathcal{B}$ can then leverage $\mathcal{A}$'s non-negligible advantage in distinguishing between the distributions $\{V_{1,i}, \ldots, V_{k_i,i}\}$ and $\{W_{1,i}, \ldots, W_{k_i,i}\}$ to achieve a non-negligible advantage in distinguishing $T = g^{c_1+c_2}$ from $T = g^{c_1+c_2+w}$, violating the decisional linear assumption. □

Chapter 4

Background on CP-ABE Systems

In this section, we give required background material on access structures, the formal definition of a CP-ABE scheme, and the security definition we will use. We also present the formal definitions for multi-authority CP-ABE schemes.

4.1 Access Structures

Definition 5. (Access Structure [7]) Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An *access structure* (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \backslash \{\}$. The sets in $\mathbb{A}$ are called the *authorized sets*, and the sets not in $\mathbb{A}$ are called the *unauthorized sets*.

In our setting, attributes will play the role of parties and we will consider only monotone access structures. One can (somewhat inefficiently) realize general access structures with our techniques by having the negation of an attribute be a separate attribute (so the total number of attributes doubles).

Linear Secret-Sharing Schemes Our constructions will employ linear secret-sharing schemes (LSSS). We use the following definition adapted from [7].

Definition 6. (Linear Secret-Sharing Schemes (LSSS)) A secret sharing scheme Π over a set of parties $\mathcal{P}$ is called *linear* (over $\mathbb{Z}_p$) if

1. The shares for each party form a vector over $\mathbb{Z}_p$.

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

We note the *linear reconstruction* property: we suppose that Π is an LSSS for access structure $\mathbb{A}$. We let S denote an authorized set, and define $I \subseteq \{1, \ldots, \ell\}$ as $I = \{j | \rho(j) \in S\}$. Then the vector $(1, 0, \ldots, 0)$ is in the span of rows of A indexed by I, and there exist constants $\{\omega_j \in \mathbb{Z}_p\}_{j \in I}$ such that, for any valid shares $\{\lambda_j\}$ of a secret s according to Π, we have: $\sum_{j \in I} \omega_j \lambda_j = s$. These constants $\{\omega_j\}$ can be found in time polynomial in the size of the share-generating matrix A [7]. For unauthorized sets, no such constants $\{\omega_j\}$ exist.

4.2 CP-ABE Definition

A ciphertext-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$(\text{PP}, M, \mathbb{A}) \rightarrow \text{CT}$ The encryption algorithm takes in the public parameters PP, the message M, and an access structure $\mathbb{A}$ over the universe of attributes. It will output a ciphertext CT such that only users whose private keys satisfy the access structure $\mathbb{A}$ should be able to extract M. We assume that $\mathbb{A}$ is implicitly included in CT.

KeyGen$(\text{MSK}, \text{PP}, S) \rightarrow \text{SK}$ The key generation algorithm takes in the master secret key MSK, the public parameters PP, and a set of attributes S. It outputs a private key SK. We assume that S 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, and a private key SK. If the set of attributes of the private key satisfies the access structure of the ciphertext, it outputs the message M.

Definition 7. A CP-ABE system is said to be correct if whenever $\mathrm{PP}, \mathrm{MSK}$ are obtained by running the setup algorithm, CT is obtained by running the encryption algorithm on $\mathrm{PP}, M, \mathbb{A}$, SK is obtained by running the key generation algorithm on $\mathrm{MSK}, \mathrm{PP}, S$ and S satisfies $\mathbb{A}$, then $\mathrm{Decrypt}(\mathrm{PP}, \mathrm{CT}, \mathrm{SK}) = M$.

4.3 Security Model for CP-ABE

We now give the full security definition for CP-ABE systems. This is described by a game between a challenger and an attacker. The game proceeds as follows:

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

Phase 1 The attacker adaptively queries the challenger for private keys corresponding to sets of attributes $S_1, \ldots, S_{Q_1}$. Each time, the challenger responds with a secret key SK_k obtained by running $\mathrm{KeyGen}(\mathrm{MSK}, \mathrm{PP}, S_k)$.

Challenge The attacker declares two equal length messages M_0 and M_1 and an access structure $\mathbb{A}$. This access structure cannot be satisfied by any of the queried attribute sets $S_1, \ldots, S_{Q_1}$. The challenger flips a random coin $b \in \{0, 1\}$, and encrypts M_b under $\mathbb{A}$, producing CT. It sends CT to the attacker.

Phase 2 The attacker adaptively queries the challenger for private keys corresponding to sets of attributes $S_{Q_1+1}, \ldots, S_Q$, with the added restriction that none of these satisfy $\mathbb{A}$. Each time, the challenger responds with a secret key SK_k obtained by running $\mathrm{KeyGen}(\mathrm{MSK}, \mathrm{PP}, S_k)$.

Guess The attacker outputs a guess b' for b.

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

Definition 8. A ciphertext-policy attribute-based encryption system is fully secure if all polynomial time attackers have at most a negligible advantage in this security game.

Selective security is defined by adding an initialization phase where the attacker must declare $\mathbb{A}$ before seeing PP.

4.4 Formal Definitions for Multi-Authority CP-ABE Systems

We now present formal definitions for multi-authority CP-ABE systems and their security. A multi-authority Ciphertext-Policy Attribute-Based Encryption system is comprised of the following five algorithms:

Global Setup$(\lambda) \rightarrow \mathrm{GP}$ The global setup algorithm takes in the security parameter λ and outputs global parameters GP for the system.

Authority Setup$(\mathrm{GP}) \rightarrow \mathrm{SK}, \mathrm{PK}$ Each authority runs the authority setup algorithm with GP as input to produce its own secret key and public key pair, SK, PK.

Encrypt$(M, \mathbb{A}, \mathrm{GP}, \{\mathrm{PK}\}) \rightarrow \mathrm{CT}$ The encryption algorithm takes in a message M, an access structure $\mathbb{A}$, the set of public keys for relevant authorities, and the global parameters. It outputs a ciphertext CT.

KeyGen$(\mathrm{GID}, \mathrm{GP}, i, \mathrm{SK}) \rightarrow \mathrm{K}_{i,\mathrm{GID}}$ The key generation algorithm takes in an identity GID, the global parameters, an attribute i belonging to some authority, and the secret key SK for this authority. It produces a key $\mathrm{K}_{i,\mathrm{GID}}$ for this attribute, identity pair.

Decrypt$(\mathrm{CT}, \mathrm{GP}, \{\mathrm{PK}\}, \{\mathrm{K}_{i,\mathrm{GID}}\}) \rightarrow M$ The decryption algorithm takes in a ciphertext, the global parameters, the public keys for the relevant authorities, and a collection of keys corresponding to attribute, identity pairs all with the same fixed identity GID. It outputs the message M when the collection of attributes i satisfies the access structure corresponding to the ciphertext.

Definition 9. A multi-authority CP-ABE system is said to be correct if whenever GP is obtained from the global setup algorithm, $\{\mathrm{PK}, \mathrm{SK}\}$ are obtained by running the authority setup algorithm, CT is obtained from the encryption algorithm on the message M using the public keys $\{PK\}$, and $\{\mathrm{K}_{i,\mathrm{GID}}\}$ is a set of keys obtained from running the key generation algorithms with $\{\mathrm{SK}\}$ for

the same identity GID and for a set of attributes satisfying the access structure of the ciphertext, $\mathit{Decrypt}(\mathrm{CT}, \mathrm{GP}, \{\mathrm{PK}\}, \{\mathrm{K}_{i,\mathrm{GID}}\}) = M$.

4.4.1 Security Definition

We define security for multi-authority Ciphertext-Policy Attribute-Based Encryption systems by the following game between a challenger and an attacker. We assume that adversaries can corrupt authorities only statically, but key queries are made adaptively. A static corruption model is also used by Chase [22] and Chase and Chow [23], but we note that our model additionally allows the adversary to choose the public keys of the corrupted authorities for itself, instead of having these initially generated by the challenger.

We let $\mathcal{S}$ denote the set of authorities and $\mathcal{U}$ denote the universe of attributes. We assume each attribute is assigned to one authority (though each authority may control multiple attributes). In practice, we can think of an attribute as being the concatenation of an authority's public key and a string attribute. This will ensure that if multiple authorities choose the same string attribute, these will still correspond to distinct attributes in the system.

Setup The global setup algorithm is run. The attacker specifies a set $\mathcal{S}' \subseteq \mathcal{S}$ of corrupt authorities. For good (non-corrupt) authorities in $\mathcal{S} - \mathcal{S}'$, the challenger obtains public key, private key pairs by running the authority setup algorithm, and gives the public keys to the attacker.

Key Query Phase 1 The attacker makes key queries by submitting pairs (i, GID) to the challenger, where i is an attribute belonging to a good authority and GID is an identity. The challenger responds by running the key generation algorithm and giving the attacker the resulting key, $\text{K}_{i,\text{GID}}$.

Challenge Phase The attacker must specify two equal length messages, M_0, M_1, and an access structure $\mathbb{A}$. The access structure must satisfy the following constraint. For each identity GID, the union of the requested attributes and all attributes controlled by the corrupt authorities must fail to satisfy $\mathbb{A}$. (In other words, the attacker cannot ask for a set of keys for one identity that allow decryption in combination with any keys that can obtained from corrupt authorities.) The attacker must also give the challenger the public keys for any corrupt authorities whose attributes appear in the access structure. The challenger flips a random coin $b \in \{0, 1\}$ and sends the attacker an encryption of M_b under access structure $\mathbb{A}$ (obtained by running the encryption algorithm).

Key Query Phase 2 The attacker may submit additional key queries (i, GID), as long as they do not violate the constraint on the challenge access structure.

Guess The attacker must submit a guess b' for b. The attacker wins if $b = b'$.

The attacker's advantage in this game is defined to be $Pr[b = b'] - \frac{1}{2}$.

Definition 10. A multi-authority Ciphertext-Policy Attribute-Based Encryption system is secure (against static corruption of authorities) if all polynomial

time attackers have at most a negligible advantage in this security game.

There are a few ways in which one might alter this security definition. One could extend the definition to allow adaptive corruption of authorities, for example, instead of requiring that all corrupt authorities be declared up front. We will not address such extensions in this work.

Chapter 5

A Basic CP-ABE System

We now present our core CP-ABE scheme, which inherits its main structure from the selectively secure scheme of Waters [69]. In [48], we observed that by simply embedding the selectively secure CP-ABE and KP-ABE schemes of [39, 69] into composite order bilinear groups, we obtain schemes that can be proven fully secure using the dual system encryption methodology. The scheme we present here can be viewed as an analog of the schemes of [48, 69] in the dual pairing vector space (DPVS) framework in prime order bilinear groups. Since the more general scheme in [60] is also an analog of these schemes in the DPVS framework, it is naturally very closely related to the construction we give here. For those familiar with these results, our construction and proof in this section is best conceptualized as a DPVS translation of our composite-order construction and proof in [48], using the translation techniques we developed in [46].

The genealogy of all these schemes should be viewed as a compelling demonstration of the value of the selective security model as a vehicle for inciting further progress, as the core construction techniques first presented in selectively secure schemes have outlived selectivity and served as a foundation

for fully secure systems. Historically, the progress of proof techniques lagged behind the insights of construction designs, but our new proof methodology can be interpreted as validating the partial progress achieved by selectively secure constructions.

The intuition behind the core of the construction is rather elegant. We work in a bilinear group, where key elements and ciphertext elements can be paired together to cancel the effects of interaction between random exponents chosen independently during encryption and key generation. To prevent collusion attacks, individual secret keys are tied together by user-specific random values that are intertwined with elements of the master secret key as well as being attached to each attribute. To enforce the access policy of the ciphertext, the encryptor chooses a random secret to be split into linear shares in the exponent of a bilinear group. The secret itself blinds the message, while the individual shares are also given out, each blinded by terms specific to the associated attribute.

The first step of the decryption process transfers the goal of recovering a fixed blinding factor into a "local" goal of recovering a value that involves both the ciphertext secret and the user-specific randomness introduced by the key. To achieve this localized goal, the user must "unblind" individual shares of the ciphertext secret, which it should only be able to accomplish when it has the corresponding attribute pieces in its secret key to pair the shares with.

Since we are working with the DPVS framework and will prove security under the decisional linear assumption, it is convenient to duplicate the

ciphertext secret as two parts (called s_1 and s_2 below) and to duplicate the key randomness as well (called t_1 and t_2 below). This is because the decisional linear assumption allows us to expand from two-dimensional to three-dimensional spaces in the exponent (as described in the previous section), so we start with two-dimensional randomness.

5.1 Construction

We will work with a bilinear group G, and we will assume that messages to be encrypted are elements of the target group G_T. We will allow linear secret sharing schemes, represented by access matrices (A, ρ), as access structures. We will assume that the size of the attribute universe is polynomial in the security parameter (this is known as "small universe" ABE). We conflate notation to let $\mathcal{U}$ denote the size of the attribute universe. In other words, the attribute universe is assumed to be $\{1, 2, \ldots, \mathcal{U}\}$.

Setup$(\lambda, \mathcal{U}) \to \text{PP}, \text{MSK}$ The setup algorithm chooses a bilinear group G of prime order p and a generator g. It randomly chooses one pair of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*)$ of dimension 3 and $\mathcal{U}$ pairs of dual orthonormal bases $(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*)$ of dimension 6. We let $\vec{b}_i, \vec{b}_i^*$ denote the basis vectors belonging to $(\mathbb{B}, \mathbb{B}^*)$, and $\vec{b}_{i,j}, \vec{b}_{i,j}^*$ denote the basis vectors belong to $(\mathbb{B}_j, \mathbb{B}_j^*)$ for each j from 1 to $\mathcal{U}$. The setup algorithm also chooses two random exponents $\alpha_1, \alpha_2 \in \mathbb{Z}_p$. The public parameters consist of:

$$\text{PP} := \{G, p, g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}} \; \forall i \in [\mathcal{U}],\; e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}\}.$$

The master secret key additionally contains:

$$\text{MSK} := \{g^{\alpha_1 \vec{b}^*_1}, g^{\alpha_2 \vec{b}^*_2}, g^{\vec{b}^*_1}, g^{\vec{b}^*_2}, g^{\vec{b}^*_{1,i}}, \ldots, g^{\vec{b}^*_{4,i}} \; \forall i \in [\mathcal{U}] \}.$$

KeyGen($\text{MSK}, S, \text{PP}) \to \text{SK}$ The key generation algorithm chooses random exponents $t_1, t_2 \in \mathbb{Z}_p$ and computes:

$$K := g^{(\alpha_1 + t_1)\vec{b}^*_1 + (\alpha_2 + t_2)\vec{b}^*_2},$$

$$K_i := g^{t_1 \vec{b}^*_{1,i} + t_1 \vec{b}^*_{2,i} + t_2 \vec{b}^*_{3,i} + t_2 \vec{b}^*_{4,i}} \; \forall i \in S.$$

The secret key is (it additionally includes S):

$$\text{SK} := \{K, \; \{K_i\}_{i \in S}\}.$$

Encrypt($(A, \rho), \text{PP}, M) \to \text{CT}$ We assume $M \in G_T$. We let $\ell \times n$ denote the dimensions of the matrix A and we recall that ρ is map from each row A_j of A to an attribute $\rho(j)$ (the index j ranges from 1 to ℓ). The encryption algorithm chooses random exponents $s_1, s_2, \{r^1_j, r^2_j\}_{j=1}^{\ell} \in \mathbb{Z}_p$. It also chooses random vectors $v_1, v_2 \in \mathbb{Z}_p^n$ with first entries equal to s_1 and s_2 respectively. The ciphertext is formed as (it additionally includes (A, ρ)):

$$M' := Me(g,g)^{\alpha_1 s_1} e(g,g)^{\alpha_2 s_2}, \; C := g^{s_1 \vec{b}_1 + s_2 \vec{b}_2},$$

$$C_j := g^{(A_j \cdot v_1 + r^1_j)\vec{b}_{1,\rho(j)} - r^1_j \vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r^2_j)\vec{b}_{3,\rho(j)} - r^2_j \vec{b}_{4,\rho(j)}} \; \forall j = 1, \ldots, \ell.$$

Decrypt$(\mathrm{CT}, \mathrm{PP}, \mathrm{SK}) \to M$ The decryption algorithm computes constants $\omega_j \in \mathbb{Z}_p$ such that $\sum_{\rho(j)\in S} \omega_j A_j = (1, 0, \ldots, 0)$. It computes:

$$X := \prod_{\rho(j)\in S} e_6(C_j, K_{\rho(j)})^{\omega_j}$$

It then computes:

$$Y := e_3(K, C).$$

The message is recovered as:

$$M = M'X/Y.$$

Correctness We observe that for each j,

$$e_6(C_j, K_{\rho(j)}) = e(g, g)^{(t_1 A_j \cdot v_1 + t_2 A_j \cdot v_2)}.$$

Thus,

$$X := \prod_{\rho(j)\in S} e_6(C_j, K_{\rho(j)})^{\omega_j} = e(g, g)^{(t_1 \sum_{\rho(j)\in S} \omega_j A_j \cdot v_1 + t_2 \sum_{\rho(j)\in S} \omega_j A_j \cdot v_2)} = e(g, g)^{(t_1 s_1 + t_2 s_2)}.$$

We note that

$$Y = e(g, g)^{(s_1(\alpha_1 + t_1) + s_2(\alpha_2 + t_2))},$$

and therefore:

$$M'X/Y = Me(g, g)^{(s_1\alpha_1 + s_2\alpha_2)} e(g, g)^{(t_1 s_1 + t_2 s_2)} / e(g, g)^{(s_1(\alpha_1 + t_1) + s_2(\alpha_2 + t_2))} = M.$$

5.2 Addressing Re-use of Attributes

We will give a security proof for the CP-ABE system above under what we call the *one-use restriction*. This means that the LSSS matrices used for the ciphertexts may only refer to a particular attribute once. In other words, the row-labeling function ρ must be injective. One way to address this limitation is to provide a general transformation from a scheme that is fully secure under this one-use restriction to a scheme that is fully secure when attributes are reused. This can be done with a simple encoding technique, as we previously observed in [48].

Suppose we have a CP-ABE system with a universe of $\mathcal{U}$ attributes with LSSS access structures that is secure when the function ρ is injective for each access structure associated to a ciphertext. Suppose we would like to have a system with $\mathcal{U}$ attributes where attributes can be used $\leq k$ times in the row labeling of a share-generating matrix. We can realize this by essentially taking k copies of each attribute in the system: instead of a single attribute B, we will have new "attributes" $B:1,\ldots,B:k$. Each time we want to label a row of an access matrix A with B, we label it with $B:i$ for a new value of i. We let ρ denote the original row labeling of A and ρ' denote this new row labeling. Each time we want to associate a subset S of attributes to a key, we instead use $S' := \{B:1,\ldots,B:k | B \in S\}$. We can then employ the one-use system on the new universe of $k\mathcal{U}$ attributes and retain its full security. We note that the set S' satisfies the access structure (A,ρ') if and only if the set S satisfies the access structure (A,ρ).

For our construction, the sizes of the public parameters and the secret keys grow linearly in the number of involved attributes, so these will expand by a factor of k under this transformation. Note that the size of the access matrix does not change, so ciphertexts in our construction will remain the same size.

Remark 11. In the next section, we will present an alternate proof strategy that allows us to prove full security directly without the one-use restriction, thus avoiding the need to set such a parameter k and the resulting efficiency costs incurred by this encoding. The trade-off will be the need for a q-type assumption (whose size depends on the behavior of the attacker). The security proof we give here under the one-use restriction relies only on the decisional linear assumption.

5.3 Security Proof Under the One-Use Restriction

We will now prove the following theorem:

Theorem 12. Under the decisional linear assumption, the CP-ABE scheme presented in Section 5.1 is fully secure in the sense of Definition 8 when access matrices (A, ρ) are required to have injective row labeling functions ρ.

We begin by defining our various types of semi-functional keys and ciphertexts. The semi-functional space in the exponent will correspond to the span of $\vec{b}_3, \vec{b}_3^*$ and the span of each $\vec{b}_{5,i}, \vec{b}_{6,i}, \vec{b}_{5,i}^*, \vec{b}_{6,i}^*$.

Semi-functional Keys To produce a semi-functional key for an attribute set S, one first calls the normal key generation algorithm to produce a normal key consisting of $K, \{K_i\}_{i \in S}$. One then chooses a random value $\gamma \in \mathbb{Z}_p$ and multiplies K by $g^{\gamma \vec{b}^*_3}$. The other components of the key remain unchanged.

Semi-functional Ciphertexts To produce a semi-functional ciphertext for an LSSS matrix (A, ρ), one first calls the normal encryption algorithm to produce a normal ciphertext consisting of $M', C, \{C_j\}$. One then chooses random values $s_3, \{r^3_j\} \in \mathbb{Z}_p$ and a random vector $v_3 \in \mathbb{Z}^n_p$ with first entry equal to s_3. The semi-functional ciphertext is:

$$M',\ Cg^{s_3 \vec{b}_3},\ C_j g^{(A_j \cdot v_3 + r^3_j)\vec{b}_{5,\rho(j)} - r^3_j \vec{b}_{6,\rho(j)}} \quad \forall j = 1, \ldots, \ell.$$

Temporary Semi-functional Keys To produce a temporary semi-functional key for an attribute set S, one first calls the normal key generation algorithm to produce a normal key consisting of $K, \{K_i\}_{i \in S}$. One then chooses random values $t_3, \gamma \in \mathbb{Z}_p$. The temporary semi-functional key is formed as:

$$Kg^{\gamma \vec{b}^*_3},\ K_i g^{t_3 \vec{b}^*_{5,i} + t_3 \vec{b}^*_{6,i}} \quad \forall i \in S.$$

Remark 13. Nominal semi-functionality will occur when $t_3 = \gamma$: in this case, a temporary semi-functional key will successfully decrypt a semi-functional ciphertext. This occurrence should be viewed as a correlation between the semi-functional ciphertext and the temporary semi-functional key, even though we semantically define it as a property of the coefficients in the key's semi-functional space. The correlation is realized through the dual orthonormal

bases. Note that $\vec{b}_3$ only appears in the semi-functional ciphertext, and without setting $\vec{b}_3$, there is ambiguity in the value of $\vec{b}_3^*$ (and hence in its coefficient in the key).

The purpose of a temporary semi-functional key is to accomplish the transition from nominal semi-functionality to regular semi-functionality via an information-theoretic argument. This argument will leverage the ambiguity in the choice of the $\vec{b}_{5,\rho(j)}, \vec{b}_{6,\rho(j)}$ vectors for attributes $\rho(j)$ that do not appear in the key. This argument relies on the one-use restriction, as well as the fact that only a single key will be a temporary semi-functional key at a time.

Our security proof will proceed as a hybrid argument over a sequence of games. We let Game_{real} denote the real security game and we let Q denote the number of key queries made by the adversary. For each k from 0 to Q, we define the following additional games:

Game_k This is like the real security game except that the challenge ciphertext given to the adversary is semi-functional, the first k keys given to the adversary are semi-functional, and the remaining keys are normal.

Game_k^T This is like Game_k, except that the k^{th} key given to the adversary is a temporary semi-functional key. The first $k-1$ keys are semi-functional, and the remaining keys are normal.

Our hybrid argument begins with Game_{real}, and then we transition to Game_0. For each k, we transition from Game_k to Game_{k+1}^T and then to

Game_{k+1}. Finally, we transition from Game_Q (where everything given to the adversary is semi-functional) to Game_{final}, which is like Game_Q except that the challenge ciphertext is a semi-functional encryption of a *random* message. In this final game, the adversary has advantage 0, since everything it receives is independent of the bit that it must guess. Our proof is accomplished in the following lemmas.

Lemma 14. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{real} and Game_0.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{real} and Game_0, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U} + 1$, $n_i = 3, k_i = 1$ for one value of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. In order to reconcile the notation of the assumption with the notation of our construction as conveniently as possible, we will denote the bases involved in the assumption by $(\mathbb{D}, \mathbb{D}^*) \in Dual(\mathbb{Z}_p^3)$ and $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore the U terms, g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, g^{\vec{d}_1}, g^{\vec{d}_2}, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}\}_{i \in [\mathcal{U}]},$$

$$g^{\eta\vec{d}_1^*}, g^{\beta\vec{d}_2^*}, g^{\vec{d}_3^*}, \{g^{\eta\vec{d}_{1,i}^*}, g^{\eta\vec{d}_{2,i}^*}, g^{\beta\vec{d}_{3,i}^*}, g^{\beta\vec{d}_{4,i}^*}, g^{\vec{d}_{5,i}^*}, g^{\vec{d}_{6,i}^*}\}_{i \in [\mathcal{U}]},$$

$$T_1, \{T_{1,i}, T_{2,i}\}_{i \in [\mathcal{U}]}.$$

The exponent of the unknown term T_1 is distributed either as $\tau_1\eta\vec{d}_1^* + \tau_2\beta\vec{d}_2^*$ or $\tau_1\eta\vec{d}_1^* + \tau_2\beta\vec{d}_2^* + \tau_3\vec{d}_3^*$. Similarly, the exponents of the unknown terms $T_{1,i}, T_{2,i}$

are distributed either as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i}$ respectively, or as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i} + \tau_3\vec{d}^*_{5,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i} + \tau_3\vec{d}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ implicitly sets the bases for the construction as:

$$\vec{b}_1 = \eta\vec{d}^*_1,\ \vec{b}_2 = \beta\vec{d}^*_2,\ \vec{b}_3 = \vec{d}^*_3,\ \vec{b}^*_1 = \eta^{-1}\vec{d}_1,\ \vec{b}^*_2 = \beta^{-1}\vec{d}_2,\ \vec{b}^*_3 = \vec{d}_3,$$

$$\vec{b}_{1,i} = \eta\vec{d}^*_{1,i},\ \vec{b}_{2,i} = \eta\vec{d}^*_{2,i},\ \vec{b}_{3,i} = \beta\vec{d}^*_{3,i},\ \vec{b}_{4,i} = \beta\vec{d}^*_{4,i},\ \vec{b}_{5,i} = \vec{d}^*_{5,i},\ \vec{b}_{6,i} = \vec{d}^*_{6,i}\ \forall i,$$

$$\vec{b}^*_{1,i} = \eta^{-1}\vec{d}_{1,i},\ \vec{b}^*_{2,i} = \eta^{-1}\vec{d}_{2,i},\ \vec{b}^*_{3,i} = \beta^{-1}\vec{d}_{3,i},\ \vec{b}^*_{4,i} = \beta^{-1}\vec{d}_{4,i},\ \vec{b}^*_{5,i} = \vec{d}_{5,i},\ \vec{b}^*_{6,i} = \vec{d}_{6,i}\ \forall i.$$

We note that these are properly distributed because $(\mathbb{D}, \mathbb{D}^*)$, $(\mathbb{D}_1, \mathbb{D}^*_1)$, etc. are randomly chosen.

$\mathcal{B}$ can use the terms given in the assumption to produce $g^{\vec{b}_1}$, $g^{\vec{b}_2}$, $\{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}$ for the public parameters. $\mathcal{B}$ chooses random values $\tilde{\alpha}_1, \tilde{\alpha}_2 \in \mathbb{Z}_p$. It implicitly sets $\alpha_1 = \eta\tilde{\alpha}_1$ and $\alpha_2 = \beta\tilde{\alpha}_2$. This allows it to produce

$$e(g,g)^{\alpha_1} = \left(e_3(g^{\vec{d}_1}, g^{\eta\vec{d}^*_1})\right)^{\tilde{\alpha}_1},\ e(g,g)^{\alpha_2} = \left(e_3(g^{\vec{d}_2}, g^{\beta\vec{d}^*_2})\right)^{\tilde{\alpha}_2}.$$

$\mathcal{B}$ gives the public parameters to $\mathcal{A}$.

To produce a normal key for an attribute set S, $\mathcal{B}$ proceeds as follows. It chooses random values $\tilde{t}_1, \tilde{t}_2 \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta\tilde{t}_1$ and $t_2 = \beta\tilde{t}_2$. It forms the key as:

$$K = g^{(\alpha_1+t_1)\vec{b}^*_1 + (\alpha_2+t_2)\vec{b}^*_2} = \left(g^{\vec{d}_1}\right)^{\tilde{\alpha}_1+\tilde{t}_1} \left(g^{\vec{d}_2}\right)^{\tilde{\alpha}_2+\tilde{t}_2},$$

$$K_i = g^{t_1\vec{b}^*_{1,i} + t_1\vec{b}^*_{2,i} + t_2\vec{b}^*_{3,i} + t_2\vec{b}^*_{4,i}} = \left(g^{\vec{d}_{1,i}}\right)^{\tilde{t}_1} \left(g^{\vec{d}_{2,i}}\right)^{\tilde{t}_1} \left(g^{\vec{d}_{3,i}}\right)^{\tilde{t}_2} \left(g^{\vec{d}_{4,i}}\right)^{\tilde{t}_2}\ \forall i \in S.$$

To produce the challenge ciphertext for an access matrix (A, ρ) of size $\ell \times n$, $\mathcal{B}$ implicitly sets $s_1 = \tau_1$ and $s_2 = \tau_2$. It chooses a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to 1. It also chooses random vectors $\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}_p^n$ with first entries equal to 0. It will implicitly set $v_1 = s_1 v + \tilde{v}_1$ and $v_2 = s_2 v + \tilde{v}_2$. We note that these are properly distributed as independent, random vectors with first entries equal to s_1 and s_2 respectively. For each j from 1 to ℓ, $\mathcal{B}$ also chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$. It implicitly sets $r_j^1 = \tilde{r}_j^3 \tau_1 + \tilde{r}_j^1$, $r_j^2 = \tilde{r}_j^3 \tau_2 + \tilde{r}_j^2$. We note that these values are properly distributed because $\tilde{r}_j^1, \tilde{r}_j^2$ are random. The ciphertext is formed as:

$$\begin{aligned} M' &= M_b \left(e_3(g^{\vec{d}_1}, T_1)\right)^{\tilde{\alpha}_1} \left(e_3(g^{\vec{d}_2}, T_1)\right)^{\tilde{\alpha}_2}, \; C = T_1, \\ C_j &= \left(T_{1,\rho(j)}\right)^{A_j \cdot v + \tilde{r}_j^3} \left(T_{2,\rho(j)}\right)^{-\tilde{r}_j^3} \left(g^{\eta \vec{d}^*_{1,\rho(j)}}\right)^{A_j \cdot \tilde{v}_1 + \tilde{r}_j^1} \\ &\quad \cdot \left(g^{\eta \vec{d}^*_{2,\rho(j)}}\right)^{-\tilde{r}_j^1} \left(g^{\beta \vec{d}^*_{3,\rho(j)}}\right)^{A_j \cdot \tilde{v}_2 + \tilde{r}_j^2} \left(g^{\beta \vec{d}^*_{4,\rho(j)}}\right)^{-\tilde{r}_j^2} \end{aligned}$$

for all j from 1 to ℓ.

If the exponents of the T terms *do not* include the τ_3 terms, then the exponent vector of C is $s_1 \vec{b}_1 + s_2 \vec{b}_2$ and the exponent vector of each C_j is:

$$\begin{aligned} &= (A_j \cdot \tau_1 v + A_j \cdot \tilde{v}_1 + \tau_1 \tilde{r}_j^3 + \tilde{r}_j^1) \eta \vec{d}^*_{1,\rho(j)} + (-\tau_1 \tilde{r}_j^3 - \tilde{r}_j^1) \eta \vec{d}^*_{2,\rho(j)} \\ &\quad + (A_j \cdot \tau_2 v + A_j \cdot \tilde{v}_2 + \tau_2 \tilde{r}_j^3 + \tilde{r}_j^2) \beta \vec{d}^*_{3,\rho(j)} + (-\tau_2 \tilde{r}_j^3 - \tilde{r}_j^2) \beta \vec{d}^*_{4,\rho(j)} \\ &= (A_j \cdot v_1 + r_j^1) \vec{b}_{1,\rho(j)} - r_j^1 \vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r_j^2) \vec{b}_{3,\rho(j)} - r_j^2 \vec{b}_{4,\rho(j)}. \end{aligned}$$

Thus we have a properly distributed normal ciphertext in this case.

If the exponents of the T terms *do* include the τ_3 terms, then the exponent vector of C is $s_1 \vec{b}_1 + s_2 \vec{b}_2 + s_3 \vec{b}_3$, where $s_3 := \tau_3$ and the exponent

vector of each C_j is:

$$(A_j \cdot v_1 + r_j^1)\vec{b}_{1,\rho(j)} - r_j^1\vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r_j^2)\vec{b}_{3,\rho(j)} - r_j^2\vec{b}_{4,\rho(j)}$$

$$+(A_j \cdot v + \tilde{r}_j^3)\tau_3\vec{b}_{5,\rho(j)} - \tilde{r}_j^3\tau_3\vec{b}_{6,\rho(j)}.$$

This is a properly distributed semi-functional ciphertext with $v_3 = \tau_3 v$ and $r_j^3 = \tau_3\tilde{r}_3^j$. (Note that these values are distributed randomly and independently from v_1, v_2, r_j^1, r_j^2.)

Thus, when the τ_3 terms are absent, $\mathcal{B}$ properly simulates Game_{real}, and when the τ_3 terms are present, $\mathcal{B}$ properly simulates Game_0. As a result, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to gain a non-negligible advantage against the subspace assumption. □

Lemma 15. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{k-1} and Game_k^T for any k from 1 to Q.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{k-1} and Game_k^T for some k, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U}+1$, $n_i = 3, k_i = 1$ for one value of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. In order to reconcile the notation of the assumption with the notation of our construction as conveniently as possible, we will denote the bases involved in the assumption by $(\mathbb{B}, \mathbb{B}^*) \in Dual(\mathbb{Z}_p^3)$ and

$(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore μ_3, g^β, g^η, $g^{\eta\tau_1}$, and $g^{\beta\tau_2}$ because they will not be needed):

$$G, p, g, g^{\vec{b}_1}, g^{\vec{b}_2}, \{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}_{i\in[\mathcal{U}]},$$

$$g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, \{g^{\eta\vec{b}_{1,i}^*}, g^{\eta\vec{b}_{2,i}^*}, g^{\beta\vec{b}_{3,i}^*}, g^{\beta\vec{b}_{4,i}^*}, g^{\vec{b}_{5,i}^*}, g^{\vec{b}_{6,i}^*}\}_{i\in[\mathcal{U}]},$$

$$U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3},$$

$$\{U_{1,i} = g^{\mu_1\vec{b}_{1,i}+\mu_2\vec{b}_{3,i}+\mu_3\vec{b}_{5,i}}, U_{2,i} = g^{\mu_1\vec{b}_{2,i}+\mu_2\vec{b}_{4,i}+\mu_3\vec{b}_{6,i}}\}_{i\in[\mathcal{U}]},$$

$$T_1, \{T_{1,i}, T_{2,i}\}_{i\in[\mathcal{U}]}.$$

The exponent of the unknown term T_1 is distributed either as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$ or as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*$. Similarly, the exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{b}_{1,i}^* + \tau_2\beta\vec{b}_{3,i}^*$ and $\tau_1\eta\vec{b}_{2,i}^* + \tau_2\beta\vec{b}_{4,i}^*$ respectively, or as $\tau_1\eta\vec{b}_{1,i}^* + \tau_2\beta\vec{b}_{3,i}^* + \tau_3\vec{b}_{5,i}^*$ and $\tau_1\eta\vec{b}_{2,i}^* + \tau_2\beta\vec{b}_{4,i}^* + \tau_3\vec{b}_{6,i}^*$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ implicitly sets $(\mathbb{B}, \mathbb{B}^*)$, $\{(\mathbb{B}_i, \mathbb{B}_i^*)\}$ as the bases for the construction. It chooses random values $\tilde{\alpha}_1, \tilde{\alpha}_2 \in \mathbb{Z}_p$ and implicitly sets $\alpha_1 = \eta\tilde{\alpha}_1$ and $\alpha_2 = \beta\tilde{\alpha}_2$. This allows it to compute $e(g,g)^{\alpha_1}$ as $e_3(g^{\vec{b}_1}, g^{\eta\vec{b}_1^*})^{\tilde{\alpha}_1}$, and $e(g,g)^{\alpha_2}$ as $e_3(g^{\vec{b}_2}, g^{\beta\vec{b}_2^*})^{\tilde{\alpha}_2}$. $\mathcal{B}$ can thus produce the public parameters, and it gives these to $\mathcal{A}$.

To respond to $\mathcal{A}$'s first $k-1$ key queries, $\mathcal{B}$ acts as follows. To produce a semi-functional key for an attribute set S, it chooses random values $\tilde{t}_1, \tilde{t}_2, \gamma \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta\tilde{t}_1$ and $t_2 = \beta\tilde{t}_2$. It forms the key as:

$$K = \left(g^{\eta\vec{b}_1^*}\right)^{\tilde{\alpha}_1+\tilde{t}_1} \left(g^{\beta\vec{b}_2^*}\right)^{\tilde{\alpha}_2+\tilde{t}_2} g^{\gamma\vec{b}_3^*},$$

$$K_i = \left(g^{\eta \vec{b}^*_{1,i}}\right)^{\tilde{t}_1} \left(g^{\eta \vec{b}^*_{2,i}}\right)^{\tilde{t}_1} \left(g^{\beta \vec{b}^*_{3,i}}\right)^{\tilde{t}_2} \left(g^{\beta \vec{b}^*_{4,i}}\right)^{\tilde{t}_2} \ \forall i \in S.$$

In this way, $\mathcal{B}$ produces properly distributed semi-functional keys in response to the first $k-1$ key requests. We note that $\mathcal{B}$ can similarly produce normal keys in response to key requests $k+1$ and onward using the same procedure except leaving off $g^{\gamma \vec{b}^*_3}$ from K.

To create the k^{th} key for some attribute set S, $\mathcal{B}$ proceeds as follows. It implicitly sets $t_1 = \eta\tau_1$ and $t_2 = \beta\tau_2$. The key is formed as:

$$K = \left(g^{\eta \vec{b}^*_1}\right)^{\tilde{\alpha}_1} \left(g^{\beta \vec{b}^*_2}\right)^{\tilde{\alpha}_2} T_1,$$

$$K_i = T_{1,i} T_{2,i} \ \forall i \in S.$$

If the exponents of the T terms here *do not* include the τ_3 terms, then this is a properly distributed normal key. If they *do* include the τ_3 terms, then this is a temporary semi-functional key with $\gamma = t_3 = \tau_3$. (In other words, the simulator is producing a nominal semi-functional key.)

To create the semi-functional ciphertext for some $n \times \ell$ access matrix (A, ρ), $\mathcal{B}$ chooses random values $\tilde{r}^1_j, \tilde{r}^2_j, \tilde{r}^3_j \in \mathbb{Z}_p$ for each j from 1 to ℓ. It also chooses a random vector $v \in \mathbb{Z}^n_p$ with first entry equal to 1, and random vectors $\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}^n_p$ with first entries equal to 0. It implicitly sets $s_1 = \mu_1$, $s_2 = \mu_2$, $s_3 = \mu_3$, $v_1 = s_1 v + \tilde{v}_1$, $v_2 = s_2 v + \tilde{v}_2$, $v_3 = s_3 v$, $r^1_j = \mu_1 \tilde{r}^3_j + \tilde{r}^1_j$, $r^2_j = \mu_2 \tilde{r}^3_j + \tilde{r}^2_j$, and $r^3_j = \mu_3 \tilde{r}^3_j$. We note that these values are properly distributed. The ciphertext is formed as:

$$M' = M_b e_3(U_1, g^{\eta \vec{b}^*_1})^{\tilde{\alpha}_1} e_3(U_1, g^{\beta \vec{b}^*_2})^{\tilde{\alpha}_2}, \ C = U_1,$$

$$C_j = \left(g^{\vec{b}_{1,\rho(j)}}\right)^{A_j \cdot \tilde{v}_1 + \tilde{r}_j^1} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{-\tilde{r}_j^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{A_j \cdot \tilde{v}_2 + \tilde{r}_j^2} \left(g^{\vec{b}_{4,\rho(j)}}\right)^{-\tilde{r}_j^2} U_{1,\rho(j)}^{A_j \cdot v + \tilde{r}_j^3} U_{2,\rho(j)}^{-\tilde{r}_j^3},$$

for all j from 1 to ℓ.

Now we must argue that in $\mathcal{A}$'s view, the k^{th} key is properly distributed as a temporary semi-functional key (i.e. nominality is hidden). In other words, we must argue that the correlation $\gamma = t_3$ that is present in $\mathcal{B}$'s view is information-theoretically hidden from $\mathcal{A}$. To see this, we consider the values of $j \in [\ell]$ in the ciphertext for which $\rho(j)$ is not in the attribute set S for the k^{th} key. For these attributes, the corresponding basis vectors $\vec{b}^*_{5,\rho(j)}, \vec{b}^*_{6,\rho(j)}$ for the semi-functional space on the key side are *never revealed* to $\mathcal{A}$, as they are not involved in any of the keys that $\mathcal{A}$ receives. This means that the ciphertext exponent vector $(A_j \cdot s_3 v + r_j^3)\vec{b}_{5,\rho(j)} - r_j^3 \vec{b}_{6,\rho(j)}$ is distributed *independently* of the share value, $A_j \cdot s_3 v$. This is because when $\vec{b}^*_{5,\rho(j)}, \vec{b}^*_{6,\rho(j)}$ are not revealed, there is ambiguity as to what $\vec{b}_{5,\rho(j)}$ and $\vec{b}_{6,\rho(j)}$ are - in particular, for any possible choices of $\vec{b}_{5,\rho(j)}$ and $\vec{b}_{6,\rho(j)}$, multiplying each by arbitrary scalars yields an alternate choice that is equally likely in $\mathcal{A}$'s view (this is a particular case of Lemma 2). Thus, no information about $A_j \cdot s_3 v$ can be learned for the values of j such that $\rho(j) \notin S$. It is important to note here that we are relying on the restriction that ρ is an injective map from rows to attributes - if the same bases vectors $\vec{b}_{5,\rho(j)}, \vec{b}_{6,\rho(j)}$ appeared repeatedly, then they would not provide fresh randomness to hide each share value.

We then observe that if one only has the values of $A_j \cdot s_3 v$ for j such that $\rho(j) \in S$, then these shares reveal no information about s_3, since they do not

correspond to a satisfying set. More precisely, the fact that S does not satisfy the access matrix implies that there exists some vector $w \in \mathbb{Z}_p^n$ such that w's first entry is 1 and w is orthogonal modulo p to all of the rows A_j such that $\rho(j) \in S$. Then $s_3 v$ can be written as $s_3 w + v'$, where v' has first coordinate equal to 0 and is distributed independently of s_3. Since the distribution of the shares $A_j \cdot s_3 v$ for $\rho(j)$ in S then only depends on v' and not on s_3, s_3 remains information-theoretically hidden.

Thus, only the distribution of C in the challenge ciphertext depends on s_3: this means that only a random multiple of $\vec{b}_3$ is revealed, and hence $t_3 \vec{b}_3^*$ is distributed as a random multiple of $\vec{b}_3^*$, even though t_3 is not random. The randomness here comes from the ambiguity in the choice of $\vec{b}_3, \vec{b}_3^*$: for any possible choice of these vectors, multiplying one by any nonzero scalar σ and the other by σ^{-1} yields an equally likely choice.

Thus, when the τ_3 terms are absent, $\mathcal{B}$ properly simulates Game_{k-1}, and when the τ_3 terms are present, $\mathcal{B}$ properly simulates Game_k^T. As a result, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to gain a non-negligible advantage against the subspace assumption.

□

Lemma 16. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^T and Game_k for any k from 1 to Q.

Proof. This is nearly identical to the proof of the previous lemma, except that

the simulator $\mathcal{B}$ will choose a random $\tilde{\gamma} \in \mathbb{Z}_p$ and multiply $(g^{\vec{b}_3^*})^{\tilde{\gamma}}$ into K when creating the k^{th} key. This ensures that this key is properly distributed as a temporary semi-functional key when the τ_3 terms are present and properly distributed as a semi-functional key when they are not. Note that the information-theoretic argument made in the previous proof is no longer needed here. □

Lemma 17. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_Q and Game_{final}.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_Q and Game_{final}, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U} + 1$, $n_i = 3, k_i = 1$ for one value of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. To coincide with our notation for the construction, we will denote the bases involved in the assumption by $(\mathbb{B}, \mathbb{B}^*) \in Dual(\mathbb{Z}_p^3)$ and $(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore μ_3, g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and $\{T_{1,i}, T_{2,i}\}_{i\in[\mathcal{U}]}$ because they will not be needed):

$$G, p, g, g^{\vec{b}_1}, g^{\vec{b}_2}, \{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}_{i\in[\mathcal{U}]},$$

$$g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, \{g^{\eta\vec{b}_{1,i}^*}, g^{\eta\vec{b}_{2,i}^*}, g^{\beta\vec{b}_{3,i}^*}, g^{\beta\vec{b}_{4,i}^*}, g^{\vec{b}_{5,i}^*}, g^{\vec{b}_{6,i}^*}\}_{i\in[\mathcal{U}]},$$

$$U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3},$$

$$\{U_{1,i} = g^{\mu_1\vec{b}_{1,i}+\mu_2\vec{b}_{3,i}+\mu_3\vec{b}_{5,i}}, U_{2,i} = g^{\mu_1\vec{b}_{2,i}+\mu_2\vec{b}_{4,i}+\mu_3\vec{b}_{6,i}}\}_{i\in[\mathcal{U}]},\ T_1.$$

The exponent of the unknown term T_1 is distributed either as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$, or as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*$. It is $\mathcal{B}$'s task to determine if this τ_3 contribution is present or not.

$\mathcal{B}$ sets $(\mathbb{B}, \mathbb{B}^*)$, $\{(\mathbb{B}_i, \mathbb{B}_i^*)\}$ as the bases for the construction. It will implicitly set $\alpha_1 = \eta\tau_1$ and $\alpha_2 = \beta\tau_2$. It forms $e(g,g)^{\alpha_1}$ as $e_3(T_1, g^{\vec{b}_1})$ and $e(g,g)^{\alpha_2}$ as $e_3(T_1, g^{\vec{b}_2})$. It gives the public parameters to $\mathcal{A}$.

To create a semi-functional key for an attribute set S, $\mathcal{B}$ proceeds as follows. It chooses random values $\tilde{t}_1, \tilde{t}_2, \tilde{\gamma} \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta\tilde{t}_1$ and $t_2 = \beta\tilde{t}_2$. It creates the key as:

$$K = T_1 \left(g^{\eta\vec{b}_1^*}\right)^{\tilde{t}_1} \left(g^{\beta\vec{b}_2^*}\right)^{\tilde{t}_2} \left(g^{\vec{b}_3^*}\right)^{\tilde{\gamma}},$$

$$K_i = \left(g^{\eta\vec{b}_{1,i}^*}\right)^{\tilde{t}_1} \left(g^{\eta\vec{b}_{2,i}^*}\right)^{\tilde{t}_1} \left(g^{\beta\vec{b}_{3,i}^*}\right)^{\tilde{t}_2} \left(g^{\beta\vec{b}_{4,i}^*}\right)^{\tilde{t}_2} \ \forall i \in S.$$

We note that the multiple of $\vec{b}_3^*$ appearing in the exponent of K is either equal to $\tilde{\gamma}$ or $\tilde{\gamma} + \tau_3$, depending on the nature of T_1. Either way, this is a properly distributed semi-functional key (whose distribution is independent of τ_3 even if it is present).

To create the semi-functional ciphertext for some $n \times \ell$ access matrix (A, ρ), $\mathcal{B}$ can use the same procedure employed in the proof of Lemma 15 to use the U terms to provide the semi-functional components. We repeat the description of this procedure below for the reader's convenience. The only difference here comes in computing the blinding factor for M'.

$\mathcal{B}$ chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$ for each j from 1 to ℓ. It also chooses a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to 1, and random vectors

$\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}_p^n$ with first entries equal to 0. It implicitly sets $s_1 = \mu_1$, $s_2 = \mu_2$, $s_3 = \mu_3$, $v_1 = s_1 v + \tilde{v}_1$, $v_2 = s_2 v + \tilde{v}_2$, $v_3 = s_3 v$, $r_j^1 = \mu_1 \tilde{r}_j^3 + \tilde{r}_j^1$, $r_j^2 = \mu_2 \tilde{r}_j^3 + \tilde{r}_j^2$, and $r_j^3 = \mu_3 \tilde{r}_j^3$. We note that these values are properly distributed. The ciphertext is formed as:

$$M' = M_b e_3(U_1, T_1),\ C = U_1,$$

$$C_j = \left(g^{\vec{b}_{1,\rho(j)}}\right)^{A_j \cdot \tilde{v}_1 + \tilde{r}_j^1} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{-\tilde{r}_j^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{A_j \cdot \tilde{v}_2 + \tilde{r}_j^2} \left(g^{\vec{b}_{4,\rho(j)}}\right)^{-\tilde{r}_j^2} U_{1,\rho(j)}^{A_j \cdot v + \tilde{r}_j^3} U_{2,\rho(j)}^{-\tilde{r}_j^3},$$

for all j from 1 to ℓ.

If the exponent of T_1 is equal to $\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_2^*$, then we have $e_3(U_1, T_1) = e(g,g)^{(\alpha_1 s_1 + \alpha_2 s_2)}$, and hence we have a properly distributed semi-functional encryption of M_b, as required in Game_Q. If instead the exponent of T_1 is equal to $\tau_1 \eta \vec{b}_1^* + \tau_2 \beta \vec{b}_2^* + \tau_3 \vec{b}_3^*$, then we have $e_3(U_1, T_1) = e(g,g)^{(\alpha_1 s_1 + \alpha_2 s_2 + \mu_3 \tau_3)}$. Since τ_3 is random (and independent of the semi-functional keys and the rest of the ciphertext), this blinding factor is distributed as a freshly random group element of G_T (note that μ_3 is nonzero with all but negligible probability). Therefore the ciphertext is distributed as a semi-functional encryption of a random message, as required in Game_{final}. Thus, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the subspace assumption.

□

Combining Lemmas 4, 14, 15, 16, and 17, Theorem 12 follows.

Chapter 6

An Unrestricted CP-ABE System

We now present a CP-ABE scheme that is proven fully secure without placing any restrictions on the reuse of attributes, retaining the efficiency of prior selectively secure schemes. Our proof strategy combines the dual system encryption methodology developed above with the selective security techniques for KP-ABE and CP-ABE systems in [39] and [69] respectively. We still rely on the decisional linear assumption to execute our dual system encryption methodology, but we will also inherit additional kinds of assumptions from the methodologies in [39, 69]. We state our additional assumptions formally below.

6.1 Additional Complexity Assumptions

The arguments in [39, 69] relied on assumptions very close to those we state below, with the main difference being that the assumptions in [39, 69] have challenge terms in the target group G_T while we will have challenge terms in the source group G. This is because the selective security arguments can afford to deal with all keys at once, and hence can use an assumption with a challenge in the target group to change the ciphertext to an encryption of a

random message. This kind of change simultaneously affects the interaction of the ciphertext with *all* keys. In our hybrid framework, we handle keys individually, and hence we use an assumption with a challenge in the source group to change the nature of individual keys one at a time, saving our progress incrementally until we arrive at the final step and can afford to change to an encryption of a random message. One could alternatively handle the Phase II queries all at once using an assumption with a challenge in the target group, but we prefer to address the two key query phases symmetrically.

We first introduce the Three Party Diffie-Hellman Assumption. This is a close relative of the standard Decisional Bilinear Diffie-Hellman Assumption, but it has a challenge term remaining in the source group. This adjustment from the usual DBDH assumption allows us to use our assumption in the semi-functional space for a particular key - without affecting the ciphertext or the other keys.

The Three Party Diffie-Hellman Assumption Given a group generator $\mathcal{G}$, we define the following distribution:

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

$$g \xleftarrow{R} G,\ x, y, z \xleftarrow{R} \mathbb{Z}_p,$$

$$D = (\mathbb{G}, g, g^x, g^y, g^z),$$

$$T_0 = g^{xyz},\ T_1 \xleftarrow{R} G.$$

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

$$Adv^{3DH}_{\mathcal{G},\mathcal{A}}(\lambda) := \left|Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1]\right|.$$

We say that $\mathcal{G}$ satisfies the Three Party Diffie-Hellman Assumption if $Adv^{3DH}_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

We next introduce a q-type assumption that we call the Source Group q-Parallel BDHE Assumption. This is a close relative of the Decisional q-Parallel Bilinear Diffie-Hellman Exponent Assumption introduced in [69], except that its challenge term remains in the source group. In Appendix 1, we prove that this assumption holds in the generic group model. Below, we use the notation $[q]$, for example, to denote the set $\{1, 2, \ldots, q\}$.

The Source Group q-Parallel BDHE Assumption Given a group generator $\mathcal{G}$ and a positive integer q, we define the following distribution:

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

$$g \xleftarrow{R} G,\ c, d, f, b_1, \ldots, b_q \xleftarrow{R} \mathbb{Z}_p,$$

The adversary will be given:

$$\begin{aligned} D \ = \ & (\mathbb{G}, g, g^f, g^{df}, g^c, g^{c^2}, \ldots, g^{c^q}, g^{c^{q+2}}, \ldots, g^{c^{2q}}, \\ & g^{c^i/b_j}\ \forall i \in [2q] \setminus \{q+1\}, j \in [q], \\ & g^{dfb_j}\ \forall j \in [q],\ g^{dfc^i b_{j'}/b_j}\ \forall i \in [q], j, j' \in [q] \text{ s.t. } j \neq j'). \end{aligned}$$

We additionally define

$$T_0 = g^{dc^{q+1}},\ T_1 \xleftarrow{R} G.$$

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

$$Adv^{q}_{\mathcal{G},\mathcal{A}}(\lambda) := \left| Pr[\mathcal{A}(D, T_0) = 1] - Pr[\mathcal{A}(D, T_1) = 1] \right|.$$

We say that $\mathcal{G}$ satisfies the Source Group q-Parallel BDHE Assumption if $Adv^{q}_{\mathcal{G},\mathcal{A}}$ is a negligible function of λ for any PPT algorithm $\mathcal{A}$.

6.2 Construction

This closely resembles the scheme from the previous section, but with one extra tuple of group elements for each key and ciphertext. This extra term is helpful in performing a cancelation during our security proof (when we are dealing with Phase II queries). We again assume that messages to be encrypted are elements of the target group G_T and we allow linear secret sharing schemes, represented by access matrices (A, ρ), as access structures. As before, we let the attribute universe be $\{1, 2, \ldots, \mathcal{U}\}$, where $\mathcal{U}$ is polynomial in the security parameter.

Setup$(\lambda, \mathcal{U}) \rightarrow \text{PP}, \text{MSK}$ The setup algorithm chooses a bilinear group G of prime order p and a generator g. It randomly chooses two pairs of dual orthonormal bases $(\mathbb{B}, \mathbb{B}^*), (\mathbb{B}_0, \mathbb{B}_0^*)$ of dimension 3 and $\mathcal{U}$ pairs of dual orthonormal bases $(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_{\mathcal{U}}, \mathbb{B}_{\mathcal{U}}^*)$ of dimension 6. We let $\vec{b}_i, \vec{b}_i^*$ denote the basis vectors belonging to $(\mathbb{B}, \mathbb{B}^*)$, and $\vec{b}_{i,j}, \vec{b}_{i,j}^*$ denote the basis vectors belonging to $(\mathbb{B}_j, \mathbb{B}_j^*)$ for each j from 0 to $\mathcal{U}$. The setup algorithm also chooses two

random exponents $\alpha_1, \alpha_2 \in \mathbb{Z}_p$. The public parameters consist of:

$$\text{PP} := \{G, p, g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_{1,0}}, g^{\vec{b}_{2,0}}, g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}} \ \forall i \in [\mathcal{U}],\ e(g,g)^{\alpha_1}, e(g,g)^{\alpha_2}\}.$$

The master secret key additionally contains:

$$\text{MSK} := \{g^{\alpha_1 \vec{b}_1^*}, g^{\alpha_2 \vec{b}_2^*}, g^{\vec{b}_1^*}, g^{\vec{b}_2^*}, g^{\vec{b}_{1,0}^*}, g^{\vec{b}_{2,0}^*}, g^{\vec{b}_{1,i}^*}, \ldots, g^{\vec{b}_{4,i}^*} \ \forall i \in [\mathcal{U}]\ \}.$$

KeyGen($\text{MSK}, S, \text{PP}) \to \text{SK}$ The key generation algorithm chooses random exponents $t_1, t_2, u_1, u_2 \in \mathbb{Z}_p$ and computes:

$$K := g^{(\alpha_1 + t_1 + u_1)\vec{b}_1^* + (\alpha_2 + t_2 + u_2)\vec{b}_2^*},$$

$$K_0 := g^{u_1 \vec{b}_{1,0}^* + u_2 \vec{b}_{2,0}^*},$$

$$K_i := g^{t_1 \vec{b}_{1,i}^* + t_1 \vec{b}_{2,i}^* + t_2 \vec{b}_{3,i}^* + t_2 \vec{b}_{4,i}^*} \ \forall i \in S.$$

The secret key is (it additionally includes S)

$$\text{SK} := \{K,\ K_0,\ \{K_i\}_{i \in S}\}.$$

Encrypt($(A, \rho), \text{PP}, M) \to \text{CT}$ We assume $M \in G_T$, A is an $\ell \times n$ matrix, and ρ is map from each row A_j of A to an attribute $\rho(j)$ (the index j ranges from 1 to ℓ). The encryption algorithm chooses random exponents $s_1, s_2, \{r_j^1, r_j^2\}_{j=1}^{\ell} \in \mathbb{Z}_p$. It also chooses random vectors $v_1, v_2 \in \mathbb{Z}_p^n$ with first entries equal to s_1 and s_2 respectively. The ciphertext is formed as (it additionally includes (A, ρ)):

$$M' := Me(g,g)^{\alpha_1 s_1} e(g,g)^{\alpha_2 s_2},\ C := g^{s_1 \vec{b}_1 + s_2 \vec{b}_2},$$

$$C_0 := g^{s_1 \vec{b}_{1,0} + s_2 \vec{b}_{2,0}},$$

$$C_j := g^{(A_j \cdot v_1 + r_j^1)\vec{b}_{1,\rho(j)} - r_j^1 \vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r_j^2)\vec{b}_{3,\rho(j)} - r_j^2 \vec{b}_{4,\rho(j)}} \ \forall j = 1, \ldots, \ell.$$

Decrypt$(\mathrm{CT}, \mathrm{PP}, \mathrm{SK}) \to M$ The decryption algorithm computes constants $\omega_j \in \mathbb{Z}_p$ such that $\sum_{\rho(j) \in S} \omega_j A_j = (1, 0, \ldots, 0)$. It computes:

$$X := \prod_{\rho(j) \in S} e_6(C_j, K_{\rho(j)})^{\omega_j}$$

It then computes:

$$Y := e_3(K, C)/e_3(K_0, C_0).$$

The message is recovered as:

$$M = M'X/Y.$$

Correctness We observe that for each j,

$$e_6(C_j, K_{\rho(j)}) = e(g, g)^{(t_1 A_j \cdot v_1 + t_2 A_j \cdot v_2)}.$$

Thus,

$$X := \prod_{\rho(j) \in S} e_6(C_j, K_{\rho(j)})^{\omega_j} = e(g, g)^{(t_1 \sum_{\rho(j) \in S} \omega_j A_j \cdot v_1 + t_2 \sum_{\rho(j) \in S} \omega_j A_j \cdot v_2)} = e(g, g)^{(t_1 s_1 + t_2 s_2)}.$$

We note that

$$Y = e(g, g)^{(s_1(\alpha_1 + t_1 + u_1) + s_2(\alpha_2 + t_2 + u_2))}/e(g, g)^{(s_1 u_1 + s_2 u_2)} = e(g, g)^{(s_1(\alpha_1 + t_1) + s_2(\alpha_2 + t_2))},$$

and therefore:

$$M'X/Y = Me(g, g)^{(s_1 \alpha_1 + s_2 \alpha_2)} e(g, g)^{(t_1 s_1 + t_2 s_2)}/e(g, g)^{(s_1(\alpha_1 + t_1) + s_2(\alpha_2 + t_2))} = M.$$

6.3 Security Proof

We now prove:

Theorem 18. Under the decisional linear assumption, the three party Diffie-Hellman assumption, and the source group q-parallel BDHE assumption defined in Section 6.1, the CP-ABE system presented in Section 6.2 is fully secure in the sense of Definition 8.

We begin by defining our various types of semi-functional keys and ciphertexts. The semi-functional space in the exponent will correspond to the span of $\vec{b}_3, \vec{b}_3^*$, the span of $\vec{b}_{3,0}, \vec{b}_{3,0}^*$, and the span of each $\vec{b}_{5,i}, \vec{b}_{6,i}, \vec{b}_{5,i}^*, \vec{b}_{6,i}^*$.

Semi-functional Keys To produce a semi-functional key for an attribute set S, one first calls the normal key generation algorithm to produce a normal key consisting of $K, K_0, \{K_i\}_{i \in S}$. One then chooses a random value $\gamma \in \mathbb{Z}_p$ and multiplies K by $g^{\gamma \vec{b}_3^*}$. The other components of the key remain unchanged.

Semi-functional Ciphertexts To produce a semi-functional ciphertext for an LSSS matrix (A, ρ), one first calls the normal encryption algorithm to produce a normal ciphertext consisting of $M', C, C_0, \{C_j\}$. One then chooses random values $s_3, \{r_j^3\} \in \mathbb{Z}_p$ and a random vector $v_3 \in \mathbb{Z}_p^n$ with first entry equal to s_3. The semi-functional ciphertext is:

$$M',\ C g^{s_3 \vec{b}_3},\ C_0 g^{s_3 \vec{b}_{3,0}},\ C_j g^{(A_j \cdot v_3 + r_j^3)\vec{b}_{5,\rho(j)} - r_j^3 \vec{b}_{6,\rho(j)}} \quad \forall j = 1, \ldots, \ell.$$

Nominal Semi-functional Keys To produce a nominal semi-functional key for an attribute set S, one first calls the normal key generation algorithm to produce a normal key consisting of $K, K_0, \{K_i\}_{i \in S}$. One then chooses random values $t_3, u_3 \in \mathbb{Z}_p$. The nominal semi-functional key is:

$$K g^{(t_3+u_3)\vec{b}^*_3},\ K_0 g^{u_3 \vec{b}^*_{3,0}},\ K_i g^{t_3 \vec{b}^*_{5,i} + t_3 \vec{b}^*_{6,i}}\ \ \forall i \in S.$$

We note that a nominal semi-functional key still correctly decrypts a semi-functional ciphertext.

Temporary Semi-functional Keys A temporary semi-functional key is similar to a nominal key, except that the semi-functional component attached to K will now be randomized (this will prevent correct decryption of a semi-functional ciphertext). More formally, to produce a temporary semi-functional key for an attribute set S, one first calls the normal key generation algorithm to produce a normal key consisting of $K, K_0, \{K_i\}_{i \in S}$. One then chooses random values $t_3, u_3, \gamma \in \mathbb{Z}_p$. The temporary semi-functional key is formed as:

$$K g^{\gamma \vec{b}^*_3},\ K_0 g^{u_3 \vec{b}^*_{3,0}},\ K_i g^{t_3 \vec{b}^*_{5,i} + t_3 \vec{b}^*_{6,i}}\ \ \forall i \in S.$$

We let Game_{real} denote the real security game. We let Q_1 denote the number of Phase I key queries the attacker makes, Q_2 denote the number of Phase II queries, and $Q = Q_1 + Q_2$ denote the total number of queries. For each k from 1 to Q, we define the following additional games:

Game$_k$ In this game, the ciphertext given to the attacker is semi-functional, as are the first k keys. The remaining keys are normal.

Game$_k^N$ This is like Game$_k$, except that the k^{th} key given to the attacker is a nominal semi-functional key. The first $k-1$ keys are still semi-functional in the original sense, while the remaining keys are normal.

Game$_k^T$ This is like Game$_k$, except that the k^{th} key given to the attacker is a temporary semi-functional key. The first $k-1$ keys are still semi-functional in the original sense, while the remaining keys are normal.

Lastly, we define Game$_{final}$ to be like Game$_Q$ (i.e. everything the attacker receives is semi-functional), except that the ciphertext is now a semi-functional encryption of a *random* message. In this final game, any attacker has advantage 0, since its view is distributed independently of the bit that it must guess.

The outer structure of our hybrid argument will progress as follows. First, we transition from Game$_{real}$ to Game$_0$, then to Game$_1$, next to Game$_2$, and so on. We ultimately arrive at Game$_Q$, where the ciphertext and *all* of the keys given to the attacker are semi-functional. We then transition to Game$_{final}$.

The transitions from Game$_{real}$ to Game$_0$ and from Game$_Q$ to Game$_{final}$ are relatively easy, and can be accomplished directly via a computational assumption. The transitions from Game$_{k-1}$ to Game$_k$ require more intricate

arguments. In order to get from Game_{k-1} to Game_k in our hybrid argument, we will transition first from Game_{k-1} to Game_k^N, then to Game_k^T, and finally to Game_k. The transition from Game_k^N to Game_k^T will require different computational assumptions for Phase I and Phase II key queries. We let Q_1 denote the number of Phase I queries, and we will address this transition separately for $k \leq Q_1$ and $k > Q_1$. Our handling of Phase I queries will closely resemble the selective security proof strategy for KP-ABE in [39], while our handling of Phase II queries will closely resemble the selective security proof strategy for CP-ABE in [69].

The hybrid security proof is accomplished in the following lemmas.

Lemma 19. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{real} and Game_0.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{real} and Game_0, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U} + 2$, $n_i = 3, k_i = 1$ for two values of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. In order to reconcile the notation of the assumption with the notation of our construction as conveniently as possible, we will denote the bases involved in the assumption by $(\mathbb{D}, \mathbb{D}^*), (\mathbb{D}_0, \mathbb{D}_0^*) \in Dual(\mathbb{Z}_p^3)$ and $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore the U terms, g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, g^{\vec{d}_1}, g^{\vec{d}_2}, g^{\vec{d}_{1,0}}, g^{\vec{d}_{2,0}}, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}\}_{i \in [\mathcal{U}]},$$

$$g^{\eta\vec{d}_1^*}, g^{\beta\vec{d}_2^*}, g^{\vec{d}_3^*}, g^{\eta\vec{d}_{1,0}^*}, g^{\beta\vec{d}_{2,0}^*}, g^{\vec{d}_{3,0}^*}, \{g^{\eta\vec{d}_{1,i}^*}, g^{\eta\vec{d}_{2,i}^*}, g^{\beta\vec{d}_{3,i}^*}, g^{\beta\vec{d}_{4,i}^*}, g^{\vec{d}_{5,i}^*}, g^{\vec{d}_{6,i}^*}\}_{i\in[\mathfrak{U}]},$$

$$T_1, T_{1,0}, \{T_{1,i}, T_{2,i}\}_{i\in[\mathfrak{U}]}.$$

The exponents of the unknown terms $T_1, T_{1,0}$ are distributed either as $\tau_1\eta\vec{d}_1^* + \tau_2\beta\vec{d}_2^*$ and $\tau_1\eta\vec{d}_{1,0}^* + \tau_2\beta\vec{d}_{2,0}^*$ respectively, or as $\tau_1\eta\vec{d}_1^* + \tau_2\beta\vec{d}_2^* + \tau_3\vec{d}_3^*$ and $\tau_1\eta\vec{d}_{1,0}^* + \tau_2\beta\vec{d}_{2,0}^* + \tau_3\vec{d}_{3,0}^*$ respectively. Similarly, the exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{d}_{1,i}^* + \tau_2\beta\vec{d}_{3,i}^*$ and $\tau_1\eta\vec{d}_{2,i}^* + \tau_2\beta\vec{d}_{4,i}^*$ respectively, or as $\tau_1\eta\vec{d}_{1,i}^* + \tau_2\beta\vec{d}_{3,i}^* + \tau_3\vec{d}_{5,i}^*$ and $\tau_1\eta\vec{d}_{2,i}^* + \tau_2\beta\vec{d}_{4,i}^* + \tau_3\vec{d}_{6,i}^*$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ implicitly sets the bases for the construction as:

$$\vec{b}_1 = \eta\vec{d}_1^*,\ \vec{b}_2 = \beta\vec{d}_2^*,\ \vec{b}_3 = \vec{d}_3^*,\ \vec{b}_1^* = \eta^{-1}\vec{d}_1,\ \vec{b}_2^* = \beta^{-1}\vec{d}_2,\ \vec{b}_3^* = \vec{d}_3,$$

$$\vec{b}_{1,0} = \eta\vec{d}_{1,0}^*,\ \vec{b}_{2,0} = \beta\vec{d}_{2,0}^*,\ \vec{b}_{3,0} = \vec{d}_{3,0}^*,\ \vec{b}_{1,0}^* = \eta^{-1}\vec{d}_{1,0},\ \vec{b}_{2,0}^* = \beta^{-1}\vec{d}_{2,0},\ \vec{b}_{3,0}^* = \vec{d}_{3,0},$$

$$\vec{b}_{1,i} = \eta\vec{d}_{1,i}^*,\ \vec{b}_{2,i} = \eta\vec{d}_{2,i}^*,\ \vec{b}_{3,i} = \beta\vec{d}_{3,i}^*,\ \vec{b}_{4,i} = \beta\vec{d}_{4,i}^*,\ \vec{b}_{5,i} = \vec{d}_{5,i}^*,\ \vec{b}_{6,i} = \vec{d}_{6,i}^*\ \forall i,$$

$$\vec{b}_{1,i}^* = \eta^{-1}\vec{d}_{1,i},\ \vec{b}_{2,i}^* = \eta^{-1}\vec{d}_{2,i},\ \vec{b}_{3,i}^* = \beta^{-1}\vec{d}_{3,i},\ \vec{b}_{4,i}^* = \beta^{-1}\vec{d}_{4,i},\ \vec{b}_{5,i}^* = \vec{d}_{5,i},\ \vec{b}_{6,i}^* = \vec{d}_{6,i}\ \forall i.$$

We note that these are properly distributed because $(\mathbb{D}, \mathbb{D}^*)$, $(\mathbb{D}_0, \mathbb{D}_0^*)$, etc. are randomly chosen.

$\mathcal{B}$ can use the terms given in the assumption to produce $g^{\vec{b}_1}$, $g^{\vec{b}_2}$, $g^{\vec{b}_{1,0}}$, $g^{\vec{b}_{2,0}}$, $\{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}$ for the public parameters. $\mathcal{B}$ chooses random values $\tilde{\alpha}_1, \tilde{\alpha}_2 \in \mathbb{Z}_p$. It implicitly sets $\alpha_1 = \eta\tilde{\alpha}_1$ and $\alpha_2 = \beta\tilde{\alpha}_2$. This allows it to produce

$$e(g,g)^{\alpha_1} = \left(e_3(g^{\vec{d}_1}, g^{\eta\vec{d}_1^*})\right)^{\tilde{\alpha}_1},\ e(g,g)^{\alpha_2} = \left(e_3(g^{\vec{d}_2}, g^{\beta\vec{d}_2^*})\right)^{\tilde{\alpha}_2}.$$

$\mathcal{B}$ gives the public parameters to $\mathcal{A}$.

To produce a normal key for an attribute set S, $\mathcal{B}$ proceeds as follows. It chooses random values $\tilde{t}_1, \tilde{t}_2, \tilde{u}_1, \tilde{u}_2 \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta\tilde{t}_1$, $t_2 = \beta\tilde{t}_2$, $u_1 = \eta\tilde{u}_1$, $u_2 = \beta\tilde{u}_2$. It forms the key as:

$$K = g^{(\alpha_1+t_1+u_1)\vec{b}_1^* + (\alpha_2+t_2+u_2)\vec{b}_2^*} = \left(g^{\vec{d}_1}\right)^{\tilde{\alpha}_1+\tilde{t}_1+\tilde{u}_1} \left(g^{\vec{d}_2}\right)^{\tilde{\alpha}_2+\tilde{t}_2+\tilde{u}_2},$$

$$K_0 = g^{u_1\vec{b}_{1,0}^* + u_2\vec{b}_{2,0}^*} = \left(g^{\vec{d}_{1,0}}\right)^{\tilde{u}_1} \left(g^{\vec{d}_{2,0}}\right)^{\tilde{u}_2},$$

$$K_i = g^{t_1\vec{b}_{1,i}^* + t_1\vec{b}_{2,i}^* + t_2\vec{b}_{3,i}^* + t_2\vec{b}_{4,i}^*} = \left(g^{\vec{d}_{1,i}}\right)^{\tilde{t}_1} \left(g^{\vec{d}_{2,i}}\right)^{\tilde{t}_1} \left(g^{\vec{d}_{3,i}}\right)^{\tilde{t}_2} \left(g^{\vec{d}_{4,i}}\right)^{\tilde{t}_2} \ \forall i \in S.$$

To produce the challenge ciphertext for an access matrix (A, ρ) of size $\ell \times n$, $\mathcal{B}$ implicitly sets $s_1 = \tau_1$ and $s_2 = \tau_2$. It chooses a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to 1. It also chooses random vectors $\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}_p^n$ with first entries equal to 0. It will implicitly set $v_1 = s_1 v + \tilde{v}_1$ and $v_2 = s_2 v + \tilde{v}_2$. We note that these are properly distributed as independent, random vectors with first entries equal to s_1 and s_2 respectively. For each j from 1 to ℓ, $\mathcal{B}$ also chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$. It implicitly sets $r_j^1 = \tilde{r}_j^3\tau_1 + \tilde{r}_j^1$, $r_j^2 = \tilde{r}_j^3\tau_2 + \tilde{r}_j^2$. We note that these values are properly distributed because $\tilde{r}_j^1, \tilde{r}_j^2$ are random. The ciphertext is formed as:

$$\begin{aligned} M' &= M_b \left(e_3(g^{\vec{d}_1}, T_1)\right)^{\tilde{\alpha}_1} \left(e_3(g^{\vec{d}_2}, T_1)\right)^{\tilde{\alpha}_2}, \ C = T_1, \ C_0 = T_{1,0}, \\ C_j &= \left(T_{1,\rho(j)}\right)^{A_j\cdot v + \tilde{r}_j^3} \left(T_{2,\rho(j)}\right)^{-\tilde{r}_j^3} \left(g^{\eta\vec{d}_{1,\rho(j)}^*}\right)^{A_j\cdot\tilde{v}_1 + \tilde{r}_j^1} \\ &\quad \cdot \left(g^{\eta\vec{d}_{2,\rho(j)}^*}\right)^{-\tilde{r}_j^1} \left(g^{\beta\vec{d}_{3,\rho(j)}^*}\right)^{A_j\cdot\tilde{v}_2 + \tilde{r}_j^2} \left(g^{\beta\vec{d}_{4,\rho(j)}^*}\right)^{-\tilde{r}_j^2} \end{aligned}$$

for all j from 1 to ℓ.

If the exponents of the T terms *do not* include the τ_3 terms, then the exponent vector of C is $s_1\vec{b}_1 + s_2\vec{b}_2$, the exponent vector of C_0 is $s_1\vec{b}_{1,0} + s_2\vec{b}_{2,0}$, and the exponent vector of C_j is:

$$\begin{aligned} &= (A_j \cdot \tau_1 v + A_j \cdot \tilde{v}_1 + \tau_1\tilde{r}_j^3 + \tilde{r}_j^1)\eta\vec{d}^*_{1,\rho(j)} + (-\tau_1\tilde{r}_j^3 - \tilde{r}_j^1)\eta\vec{d}^*_{2,\rho(j)} \\ &\quad +(A_j \cdot \tau_2 v + A_j \cdot \tilde{v}_2 + \tau_2\tilde{r}_j^3 + \tilde{r}_j^2)\beta\vec{d}^*_{3,\rho(j)} + (-\tau_2\tilde{r}_j^3 - \tilde{r}_j^2)\beta\vec{d}^*_{4,\rho(j)} \\ &= (A_j \cdot v_1 + r_j^1)\vec{b}_{1,\rho(j)} - r_j^1\vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r_j^2)\vec{b}_{3,\rho(j)} - r_j^2\vec{b}_{4,\rho(j)}. \end{aligned}$$

Thus we have a properly distributed normal ciphertext in this case.

If the exponents of the T terms *do* include the τ_3 terms, then the exponent vector of C is $s_1\vec{b}_1 + s_2\vec{b}_2 + s_3\vec{b}_3$, where $s_3 := \tau_3$, the exponent vector of C_0 is $s_1\vec{b}_{1,0} + s_2\vec{b}_{2,0} + s_3\vec{b}_{3,0}$, and the exponent vector of each C_j is:

$$(A_j \cdot v_1 + r_j^1)\vec{b}_{1,\rho(j)} - r_j^1\vec{b}_{2,\rho(j)} + (A_j \cdot v_2 + r_j^2)\vec{b}_{3,\rho(j)} - r_j^2\vec{b}_{4,\rho(j)}$$

$$+(A_j \cdot v + \tilde{r}_j^3)\tau_3\vec{b}_{5,\rho(j)} - \tilde{r}_j^3\tau_3\vec{b}_{6,\rho(j)}.$$

This is a properly distributed semi-functional ciphertext with $v_3 = \tau_3 v$ and $r_j^3 = \tau_3\tilde{r}_3^j$. (Note that these values are distributed randomly and independently from v_1, v_2, r_j^1, r_j^2.)

Thus, when the τ_3 terms are absent, $\mathcal{B}$ properly simulates Game_{real}, and when the τ_3 terms are present, $\mathcal{B}$ properly simulates Game_0. As a result, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to gain a non-negligible advantage against the subspace assumption.

□

Lemma 20. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_{k-1} and Game_k^N for any k from 1 to Q.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_{k-1} and Game_k^N for some k, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U}+2$, $n_i = 3, k_i = 1$ for two values of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. In order to reconcile the notation of the assumption with the notation of our construction as conveniently as possible, we will denote the bases involved in the assumption by $(\mathbb{B}, \mathbb{B}^*), (\mathbb{B}_0, \mathbb{B}_0^*) \in Dual(\mathbb{Z}_p^3)$ and $(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_{1,0}}, g^{\vec{b}_{2,0}}, \{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}_{i\in[\mathcal{U}]},$$

$$g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, g^{\eta\vec{b}_{1,0}^*}, g^{\beta\vec{b}_{2,0}^*}, g^{\vec{b}_{3,0}^*}, \{g^{\eta\vec{b}_{1,i}^*}, g^{\eta\vec{b}_{2,i}^*}, g^{\beta\vec{b}_{3,i}^*}, g^{\beta\vec{b}_{4,i}^*}, g^{\vec{b}_{5,i}^*}, g^{\vec{b}_{6,i}^*}\}_{i\in[\mathcal{U}]},$$

$$U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3}, U_{1,0} = g^{\mu_1\vec{b}_{1,0}+\mu_2\vec{b}_{2,0}+\mu_3\vec{b}_{3,0}},$$

$$\{U_{1,i} = g^{\mu_1\vec{b}_{1,i}+\mu_2\vec{b}_{3,i}+\mu_3\vec{b}_{5,i}}, U_{2,i} = g^{\mu_1\vec{b}_{2,i}+\mu_2\vec{b}_{4,i}+\mu_3\vec{b}_{6,i}}\}_{i\in[\mathcal{U}]},$$

$$T_1, T_{1,0}, \{T_{1,i}, T_{2,i}\}_{i\in[\mathcal{U}]}.$$

The exponents of the unknown terms $T_1, T_{1,0}$ are distributed either as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$ and $\tau_1\eta\vec{b}_{1,0}^* + \tau_2\beta\vec{b}_{2,0}^*$ respectively, or as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*$ and $\tau_1\eta\vec{b}_{1,0}^* + \tau_2\beta\vec{b}_{2,0}^* + \tau_3\vec{b}_{3,0}^*$ respectively. Similarly, the exponents of the unknown terms

$T_{1,i}, T_{2,i}$ are distributed either as $\tau_1 \eta \vec{b}^*_{1,i} + \tau_2 \beta \vec{b}^*_{3,i}$ and $\tau_1 \eta \vec{b}^*_{2,i} + \tau_2 \beta \vec{b}^*_{4,i}$ respectively, or as $\tau_1 \eta \vec{b}^*_{1,i} + \tau_2 \beta \vec{b}^*_{3,i} + \tau_3 \vec{b}^*_{5,i}$ and $\tau_1 \eta \vec{b}^*_{2,i} + \tau_2 \beta \vec{b}^*_{4,i} + \tau_3 \vec{b}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ implicitly sets $(\mathbb{B}, \mathbb{B}^*)$, $(\mathbb{B}_0, \mathbb{B}^*_0)$, $\{(\mathbb{B}_i, \mathbb{B}^*_i)\}$ as the bases for the construction. It chooses random values $\tilde{\alpha}_1, \tilde{\alpha}_2 \in \mathbb{Z}_p$ and implicitly sets $\alpha_1 = \eta \tilde{\alpha}_1$ and $\alpha_2 = \beta \tilde{\alpha}_2$. This allows it to compute $e(g,g)^{\alpha_1}$ as $e_3(g^{\vec{b}_1}, g^{\eta \vec{b}^*_1})^{\tilde{\alpha}_1}$, and $e(g,g)^{\alpha_2}$ as $e_3(g^{\vec{b}_2}, g^{\beta \vec{b}^*_2})^{\tilde{\alpha}_2}$. $\mathcal{B}$ can thus produce the public parameters, and it gives these to $\mathcal{A}$.

To respond to $\mathcal{A}$'s first $k-1$ key queries, $\mathcal{B}$ acts as follows. To produce a semi-functional key for an attribute set S, it chooses random values $\tilde{t}_1, \tilde{t}_2, \tilde{u}_1, \tilde{u}_2, \gamma \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta \tilde{t}_1$, $t_2 = \beta \tilde{t}_2$, $u_1 = \eta \tilde{u}_1$, $u_2 = \beta \tilde{u}_2$. It forms the key as:

$$K = \left(g^{\eta \vec{b}^*_1}\right)^{\tilde{\alpha}_1 + \tilde{t}_1 + \tilde{u}_1} \left(g^{\beta \vec{b}^*_2}\right)^{\tilde{\alpha}_2 + \tilde{t}_2 + \tilde{u}_2} g^{\gamma \vec{b}^*_3},$$

$$K_0 = \left(g^{\eta \vec{b}^*_{1,0}}\right)^{\tilde{u}_1} \left(g^{\beta \vec{b}^*_{2,0}}\right)^{\tilde{u}_2},$$

$$K_i = \left(g^{\eta \vec{b}^*_{1,i}}\right)^{\tilde{t}_1} \left(g^{\eta \vec{b}^*_{2,i}}\right)^{\tilde{t}_1} \left(g^{\beta \vec{b}^*_{3,i}}\right)^{\tilde{t}_2} \left(g^{\beta \vec{b}^*_{4,i}}\right)^{\tilde{t}_2} \ \forall i \in S.$$

In this way, $\mathcal{B}$ produces properly distributed semi-functional keys in response to the first $k-1$ key requests. We note that $\mathcal{B}$ can similarly produce normal keys in response to key requests $k+1$ and onward using the same procedure except leaving off $g^{\gamma \vec{b}^*_3}$ from K.

To create the k^{th} key for some attribute set S, $\mathcal{B}$ proceeds as follows. It chooses random values $\tilde{u}_1, \tilde{u}_2, \tilde{u}_3 \in \mathbb{Z}_p$. It implicitly sets $t_1 = \eta \tau_1$, $t_2 = \beta \tau_2$,

$u_1 = \eta(\tau_1\tilde{u}_3 + \tilde{u}_1)$, and $u_2 = \beta(\tau_2\tilde{u}_3 + \tilde{u}_2)$. We note that these values are independently random because $\tau_1, \tau_2, \tilde{u}_1, \tilde{u}_2$ are independently random. The key is formed as:

$$K = \left(g^{\eta\vec{b}_1^*}\right)^{\tilde{\alpha_1}+\tilde{u}_1} \left(g^{\beta\vec{b}_2^*}\right)^{\tilde{\alpha}_2+\tilde{u}_2} T_1(T_1)^{\tilde{u}_3},$$

$$K_0 = \left(g^{\eta\vec{b}_{1,0}^*}\right)^{\tilde{u}_1} \left(g^{\beta\vec{b}_{2,0}^*}\right)^{\tilde{u}_2} T_{1,0}^{\tilde{u}_3},$$

$$K_i = T_{1,i}T_{2,i} \; \forall i \in S.$$

If the exponents of the T terms here *do not* include the τ_3 terms, then this is a properly distributed normal key. If they *do* include the τ_3 terms, then this is a properly distributed nominal semi-functional key with $t_3 = \tau_3$ and $u_3 = \tau_3\tilde{u}_3$. (Note that these values are random and independent of t_1, t_2, u_1, u_2.)

To create the semi-functional ciphertext for some $n \times \ell$ access matrix (A, ρ), $\mathcal{B}$ chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$ for each j from 1 to ℓ. It also chooses a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to 1, and random vectors $\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}_p^n$ with first entries equal to 0. It implicitly sets $s_1 = \mu_1$, $s_2 = \mu_2$, $s_3 = \mu_3$, $v_1 = s_1 v + \tilde{v}_1$, $v_2 = s_2 v + \tilde{v}_2$, $v_3 = s_3 v$, $r_j^1 = \mu_1\tilde{r}_j^3 + \tilde{r}_j^1$, $r_j^2 = \mu_2\tilde{r}_j^3 + \tilde{r}_j^2$, and $r_j^3 = \mu_3\tilde{r}_j^3$. We note that these values are properly distributed. The ciphertext is formed as:

$$M' = M_b e_3(U_1, g^{\eta\vec{b}_1^*})^{\tilde{\alpha}_1} e_3(U_1, g^{\beta\vec{b}_2^*})^{\tilde{\alpha}_2}, \; C = U_1, \; C_0 = U_{1,0},$$

$$C_j = \left(g^{\vec{b}_{1,\rho(j)}}\right)^{A_j\cdot\tilde{v}_1+\tilde{r}_j^1} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{-\tilde{r}_j^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{A_j\cdot\tilde{v}_2+\tilde{r}_j^2} \left(g^{\vec{b}_{4,\rho(j)}}\right)^{-\tilde{r}_j^2} U_{1,\rho(j)}^{A_j\cdot v+\tilde{r}_j^3} U_{2,\rho(j)}^{-\tilde{r}_j^3},$$

for all j from 1 to ℓ.

Thus, when the τ_3 terms are absent, $\mathcal{B}$ properly simulates Game_{k-1}, and when the τ_3 terms are present, $\mathcal{B}$ properly simulates Game_k^N. As a result, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to gain a non-negligible advantage against the subspace assumption. $\square$

Lemma 21. Under the three party Diffie-Hellman assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^N and Game_k^T for any k from 1 to Q_1 (recall these are all the Phase I queries).

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_k^N and Game_k^T for some k between 1 and Q_1, we will create a PPT algorithm $\mathcal{B}$ to break the three party Diffie-Hellman assumption. $\mathcal{B}$ is given g, g^x, g^y, g^z, T, where T is either g^{xyz} or a random element of G. $\mathcal{B}$ will simulate either Game_k^N or Game_k^T with $\mathcal{A}$ depending on the nature of T.

$\mathcal{B}$ chooses random dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*), (\mathbb{D}_0, \mathbb{D}_0^*)$ of dimension 3 and $(\mathbb{D}_1, \mathbb{D}_1^*)$, ..., $(\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*)$ of dimension 6. It then implicitly sets $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{B}_0, \mathbb{B}_0^*)$ as follows:

$$\vec{b}_1 = \vec{d}_1,\ \vec{b}_2 = \vec{d}_2,\ \vec{b}_3 = (xy)^{-1}\vec{d}_3,\ \vec{b}_1^* = \vec{d}_1^*,\ \vec{b}_2^* = \vec{d}_2^*,\ \vec{b}_3^* = xy\vec{d}_3^*,$$

$$\vec{b}_{1,0} = \vec{d}_{1,0},\ \vec{b}_{2,0} = \vec{d}_{2,0},\ \vec{b}_{3,0} = (xy)^{-1}\vec{d}_{3,0},\ \vec{b}_{1,0}^* = \vec{d}_{1,0}^*,\ \vec{b}_{2,0}^* = \vec{d}_{2,0}^*,\ \vec{d}_{3,0}^* = xy\vec{d}_{3,0}^*.$$

We note $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{B}_0, \mathbb{B}_0^*)$ are properly distributed.

$\mathcal{B}$ sets the *normal* portions of $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*)$ as follows:

$$\vec{b}_{1,i} = \vec{d}_{1,i},\ \vec{b}_{2,i} = \vec{d}_{2,i},\ \vec{b}_{3,i} = \vec{d}_{3,i},\ \vec{b}_{4,i} = \vec{d}_{4,i}\ \forall i = 1, \ldots, \mathcal{U},$$

$$\vec{b}^*_{1,i} = \vec{d}^*_{1,i},\ \vec{b}^*_{2,i} = \vec{d}^*_{2,i},\ \vec{b}^*_{3,i} = \vec{d}^*_{3,i},\ \vec{b}^*_{4,i} = \vec{d}^*_{4,i}\ \forall i = 1, \ldots, \mathcal{U}.$$

The semi-functional portions of these bases will be set later (at which point we may verify that all of $(\mathbb{B}_1, \mathbb{B}^*_1)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}^*_\mathcal{U})$ are properly distributed).

$\mathcal{B}$ chooses $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ randomly. We observe that $\mathcal{B}$ can now produce the public parameters, and also knows the master secret key (enabling it to create normal keys). It gives the public parameters to $\mathcal{A}$. To create the first $k-1$ semi-functional keys in response to $\mathcal{A}$'s key requests, $\mathcal{B}$ first creates a normal key, then raises $g^{\vec{d}^*_3}$ to a random exponent in $\mathbb{Z}_p$ and multiplies this by K. We are using here that $\mathcal{B}$ does not need to know $g^{\vec{b}^*_3}$ precisely in order to create well-distributed semi-functional keys - it suffices for $\mathcal{B}$ to know $g^{c\vec{b}^*_3}$ for some (nonzero) $c \in \mathbb{Z}_p$.

$\mathcal{A}$ requests the k^{th} key for some attribute set $S \subset [\mathcal{U}]$. At this point, $\mathcal{B}$ implicitly defines the semi-functional parts of the bases $(\mathbb{B}_1, \mathbb{B}^*_1)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}^*_\mathcal{U})$ as follows (note that these have not been involved in the game before this):

$$\vec{b}_{5,i} = x^{-1}\vec{d}_{5,i},\ \vec{b}_{6,i} = \vec{d}_{6,i}\ \vec{b}^*_{5,i} = x\vec{d}^*_{5,i},\ \vec{b}^*_{6,i} = \vec{d}^*_{6,i}\ \forall i \notin S,$$

$$\vec{b}_{5,i} = \vec{d}_{5,i},\ \vec{b}_{6,i} = \vec{d}_{6,i},\ \vec{b}^*_{5,i} = \vec{d}^*_{5,i},\ \vec{b}^*_{6,i} = \vec{d}^*_{6,i}\ \forall i \in S.$$

We observe that all of $(\mathbb{B}, \mathbb{B}^*)$, $(\mathbb{B}_0, \mathbb{B}^*_0)$, $(\mathbb{B}_1, \mathbb{B}^*_1)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}^*_\mathcal{U})$ are properly distributed, and their distribution is independent of x, y, and S (the involvement of x, y, and S is only present in $\mathcal{B}$'s view and is information-theoretically hidden from $\mathcal{A}$, see Lemma 2).

To create the k^{th} key, $\mathcal{B}$ first creates a normal key with components $K, K_0, \{K_i\}_{i \in S}$. To create the semi-functional components, it chooses a ran-

dom value $\tilde{u}_3$ and implicitly sets $t_3 = z$, $u_3 = (xy)^{-1}\tilde{u}_3$. It then forms the semi-functional component for K_0 as

$$g^{u_3 \vec{b}^*_{3,0}} = g^{\tilde{u}_3 \vec{d}^*_{3,0}}$$

and the semi-functional component for each K_i as

$$g^{t_3 \vec{b}^*_{5,i} + t_3 \vec{b}^*_{6,i}} = (g^z)^{\vec{d}^*_{5,i} + \vec{d}^*_{6,i}} \ \forall i \in S.$$

It forms the semi-functional component for K as:

$$T^{\vec{d}^*_3} g^{\tilde{u}_3 \vec{d}^*_3}.$$

If $T = g^{xyz}$, then the exponent vector here is $xyz\vec{d}^*_3 + \tilde{u}_3\vec{d}^*_3 = (z + u_3)\vec{b}^*_3$, as required for a nominal semi-functional key. Otherwise, this exponent vector is distributed as a random multiple of $\vec{b}^*_3$, as required for a temporary semi-functional key. $\mathcal{B}$ multiplies these semi-functional components with the normal $K, K_0, \{K_i\}_{i \in S}$ to produce the key it gives to $\mathcal{A}$. It can respond to the rest of $\mathcal{A}$'s key queries by calling the normal key generation algorithm.

At some *later* point, $\mathcal{A}$ requests the challenge ciphertext for some $\ell \times n$ access matrix (A, ρ) that is *not* satisfied by the attribute set S. $\mathcal{B}$ first creates a normal ciphertext with components $M', C, C_0, \{C_j\}_{j=1}^{\ell}$. To create the semi-functional components, $\mathcal{B}$ first computes a vector $\nu \in \mathbb{Z}_p^n$ that has first entry equal to 1 and is orthogonal to all of the rows A_j of A such that $\rho(j) \in S$ (such a vector must exist since S fails to satisfy A, and it is efficiently computable). $\mathcal{B}$ also chooses a random vector $\tilde{v}_3 \in \mathbb{Z}_p^n$ subject to the constraint that the

first entry is zero. It implicitly sets $s_3 = xy$ and sets $v_3 = xy\nu + x\tilde{v}_3$. We note that s_3 is random because all of the dual orthonormal bases are distributed independently of x, y, and v_3 is distributed as a random vector with first entry equal to s_3. $\mathcal{B}$ also chooses random values $r_j^3 \in \mathbb{Z}_p$ for all j such that $\rho(j) \in S$ and random values $\tilde{r}_j^3 \in \mathbb{Z}_p$ for all j such that $\rho(j) \notin S$. For values of j such that $\rho(j) \notin S$, it implicitly sets $r_j^3 = x\tilde{r}_j^3$. $\mathcal{B}$ can then produce the semi-functional components of the ciphertext as:

$$g^{s_3 \vec{b}_3} = g^{\vec{d}_3},\ g^{s_3 \vec{b}_{3,0}} = g^{\vec{d}_{3,0}},$$

$$g^{(A_j \cdot v_3 + r_j^3)\vec{b}_{5,\rho(j)} - r_j^3 \vec{b}_{6,\rho(j)}} = (g^y)^{A_j \cdot \nu \vec{d}_{5,\rho(j)}}\, g^{(A_j \cdot \tilde{v}_3 + \tilde{r}_j^3)\vec{d}_{5,\rho(j)}}\, (g^x)^{-\tilde{r}_j^3 \vec{d}_{6,\rho(j)}}\ \forall j \text{ s.t. } \rho(j) \notin S,$$

$$g^{(A_j \cdot v_3 + r_j^3)\vec{b}_{5,\rho(j)} - r_j^3 \vec{b}_{6,\rho(j)}} = (g^x)^{A_j \cdot \tilde{v}_3 \vec{d}_{5,\rho(j)}}\, g^{r_j^3 \vec{d}_{5,\rho(j)} - r_j^3 \vec{d}_{6,\rho(j)}}\ \forall j \text{ s.t. } \rho(j) \in S.$$

Here we have used the fact that $A_j \cdot \nu = 0$ modulo p to avoid needing to produce a multiple of $g^{xy\vec{d}_{5,\rho(j)}}$ for j such that $\rho(j) \in S$.

$\mathcal{B}$ multiplies these semi-functional components by the normal components to form the semi-functional ciphertext, which it gives to $\mathcal{A}$. If $T = g^{xyz}$, then $\mathcal{B}$ has properly simulated Game_K^N, and if T is a random group element, then $\mathcal{B}$ has properly simulated Game_k^T. Thus, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to gain a non-negligible advantage against the three party Diffie-Hellman assumption. □

Lemma 22. Under the source group q-parallel BDHE assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^N and Game_k^T for a $k > Q_1$ using an access matrix (A, ρ) of size $\ell \times n$ where $\ell, n \leq q$.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_k^N and Game_k^T for some k such that $Q_1 < k \leq Q$ using an access matrix with dimensions $\leq q$, we will create a PPT algorithm $\mathcal{B}$ to break the source group q-parallel BDHE assumption. Our $\mathcal{B}$ is given: $g, g^f, g^{df}, g^{c^i}\ \forall i \in [2q] \setminus \{q+1\}$, $g^{c^i/b_j}\ \forall i \in [2q] \setminus \{q+1\}, j \in [q]$, $g^{dfb_j}\ \forall j \in [q]$, $g^{dfc^i b_{j'}/b_j}\ \forall i \in [q], j, j' \in [q]$ such that $j \neq j'$, and T, where T is either equal to $g^{dc^{q+1}}$ or is a random element of G. $\mathcal{B}$ will simulate either Game_k^N or Game_k^T with $\mathcal{A}$, depending on T.

$\mathcal{B}$ chooses random dual orthonormal bases $(\mathbb{D}, \mathbb{D}^*), (\mathbb{D}_0, \mathbb{D}_0^*)$ of dimension 3 and $(\mathbb{D}_1, \mathbb{D}_1^*)$, ..., $(\mathbb{D}_{\mathcal{U}}, \mathbb{D}_{\mathcal{U}}^*)$ of dimension 6. It then implicitly sets $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{B}_0, \mathbb{B}_0^*)$ as follows:

$$\vec{b}_1 = \vec{d}_1,\ \vec{b}_2 = \vec{d}_2,\ \vec{b}_3 = (cd)^{-1}\vec{d}_3,\ \vec{b}_1^* = \vec{d}_1^*,\ \vec{b}_2^* = \vec{d}_2^*,\ \vec{b}_3^* = (cd)\vec{d}_3^*,$$

$$\vec{b}_{1,0} = \vec{d}_{1,0},\ \vec{b}_{2,0} = \vec{d}_{2,0},\ \vec{b}_{3,0} = (c)^{-1}\vec{d}_{3,0},\ \vec{b}_{1,0}^* = \vec{d}_{1,0}^*,\ \vec{b}_{2,0}^* = \vec{d}_{2,0}^*,\ \vec{b}_{3,0}^* = (c)\vec{d}_{3,0}^*.$$

We note that $(\mathbb{B}, \mathbb{B}^*)$ and $(\mathbb{B}_0, \mathbb{B}_0^*)$ are properly distributed.

$\mathcal{B}$ sets the *normal* portions of $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_{\mathcal{U}}, \mathbb{B}_{\mathcal{U}}^*)$ as follows:

$$\vec{b}_{1,i} = \vec{d}_{1,i},\ \vec{b}_{2,i} = \vec{d}_{2,i},\ \vec{b}_{3,i} = \vec{d}_{3,i},\ \vec{b}_{4,i} = \vec{d}_{4,i}\ \forall i = 1, \ldots, \mathcal{U},$$

$$\vec{b}_{1,i}^* = \vec{d}_{1,i}^*,\ \vec{b}_{2,i}^* = \vec{d}_{2,i}^*,\ \vec{b}_{3,i}^* = \vec{d}_{3,i}^*,\ \vec{b}_{4,i}^* = \vec{d}_{4,i}^*\ \forall i = 1, \ldots, \mathcal{U}.$$

The semi-functional portions of these bases will be set later (at which point we may verify that all of $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_{\mathcal{U}}, \mathbb{B}_{\mathcal{U}}^*)$ are properly distributed).

$\mathcal{B}$ chooses $\alpha_1, \alpha_2 \in \mathbb{Z}_p$ randomly. We observe that $\mathcal{B}$ can now produce the public parameters, and also knows the master secret key (enabling it to

create normal keys). It gives the public parameters to $\mathcal{A}$. To create the first $k-1$ semi-functional keys in response to $\mathcal{A}$'s key requests, $\mathcal{B}$ first creates a normal key, then raises $g^{\vec{d}_3^*}$ to a random exponent in $\mathbb{Z}_p$ and multiplies this by K. As in the proof of the previous lemma, we note here that $\mathcal{B}$ does not need to know $g^{\vec{b}_3^*}$ precisely in order to create well-distributed semi-functional keys.

Before requesting the k^{th} key, $\mathcal{A}$ will request the challenge ciphertext for some access matrix (A, ρ) of size $\ell \times n$, where both $\ell, n \leq q$. For each attribute i, we let J_i denote the set of indices $j \in [\ell]$ such that $\rho(j) = i$. For each i, $\mathcal{B}$ chooses a random value $\tilde{\eta}_i$ and defines a value $\eta_i \in \mathbb{Z}_p$ by

$$\eta_i = \tilde{\eta}_i + \sum_{j \in J_i} cA_{j,1}/b_j + \ldots + c^n A_{j,n}/b_j.$$

At this point, $\mathcal{B}$ implicitly sets the semi-functional portions of the bases $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*)$ as follows (note that these have played no role in the game before this point):

$$\vec{b}_{5,i} = \vec{d}_{5,i},\ \vec{b}_{6,i} = \eta_i^{-1}\vec{d}_{6,i}\ \vec{b}_{5,i}^* = \vec{d}_{5,i}^*,\ \vec{b}_{6,i}^* = \eta_i \vec{d}_{6,i}^*\ \forall i.$$

We observe that $(\mathbb{B}_1, \mathbb{B}_1^*)$, ..., $(\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*)$ are properly distributed.

To create the challenge ciphertext, $\mathcal{B}$ first creates a normal ciphertext using the normal encryption algorithm. To create the semi-functional components, it implicitly sets $s_3 = cdf$. It also chooses random values $y_2, \ldots, y_n \in \mathbb{Z}_p$ and random values $\tilde{r}_j^3 \in \mathbb{Z}_p$ for each $j \in [\ell]$. It implicitly sets $v_3 = (cdf, dfc^2 + y_2, \ldots, dfc^n + y_n)$. This is distributed as a random vector with first entry equal to s_3. For each $j \in [\ell]$, $\mathcal{B}$ implicitly sets $r_3^j = -dfb_j\eta_{\rho(j)} + \tilde{r}_3^j\eta_{\rho(j)}$. These

are distributed as uniformly random elements because each $\tilde{r}_j^3$ is random and $\eta_{\rho(j)} \neq 0$ with all but negligible probability. We observe:

$$\begin{aligned} A_j \cdot v_3 + r_j^3 &= df(cA_{j,1} + c^2 A_{j,2} + \ldots + c^n A_{j,n}) + A_{j,2} y_2 + \ldots + A_{j,n} y_n \\ &\quad -df b_j \left(\tilde{\eta}_{\rho(j)} + \sum_{j' \in J_{\rho(j)}} cA_{j',1}/b_{j'} + \ldots + c^n A_{j',n}/b_{j'} \right) + \tilde{r}_j^3 \eta_{\rho(j)} \end{aligned}$$

By definition, $j \in J_{\rho(j)}$, so we have some cancelation here:

$$\begin{aligned} A_j \cdot v_3 + r_j^3 &= A_{j,2} y_2 + \ldots + A_{j,n} y_n + \tilde{r}_j^3 \eta_{\rho(j)} \\ &\quad -df b_j \left(\tilde{\eta}_{\rho(j)} + \sum_{j' \in J_{\rho(j)} \setminus \{j\}} cA_{j',1}/b_{j'} + \ldots + c^n A_{j'n}/b_{j'} \right). \end{aligned}$$

We now see that $\mathcal{B}$ can compute $g^{A_j \cdot v_3 + r_j^3}$ using the terms it is given in the assumption, enabling it to produce $g^{(A_j \cdot v_3 + r_j^3)\vec{d}_{5,\rho(j)}} = g^{(A_j \cdot v_3 + r_j^3)\vec{b}_{5,i}}$. We also see that

$$-r_j^3 \vec{b}_{6,\rho(j)} = -r_3^j \eta_{\rho(j)}^{-1} \vec{d}_{6,i} = (df b_j - \tilde{r}_3^j) \vec{d}_{6,i},$$

so $\mathcal{B}$ can also produce $g^{-r_j^3 \vec{b}_{6,\rho(j)}}$. In this way, $\mathcal{B}$ produces the semi-functional component of C_j for each j with the proper distribution.

$\mathcal{B}$ also produces the semi-functional components of C and C_0 as:

$$g^{s_3 \vec{b}_3} = \left(g^f\right)^{\vec{d}_3}, \; g^{s_3 \vec{b}_{3,0}} = \left(g^{df}\right)^{\vec{d}_{3,0}}.$$

It gives the resulting properly distributed semi-functional ciphertext to $\mathcal{A}$.

At some *later* point in the game, $\mathcal{A}$ requests the k^{th} key for some attribute set S. $\mathcal{B}$ can create the normal parts of the key using the normal key generation algorithm. To create the semi-functional parts, $\mathcal{B}$ proceeds as

follows. Since S does not satisfy (A, ρ), $\mathcal{B}$ can (efficiently) compute a vector $w \in \mathbb{Z}_p^n$ such that its first entry is non-zero and w is orthogonal (modulo p) to all rows A_j of A such that $\rho(j) \in S$. We may assume the first entry of w is randomized. $\mathcal{B}$ implicitly sets $t_3 = w_1 c^q + \ldots + w_n c^{q-n+1}$, which is properly distributed because w_1 is random (and c is nonzero with all but negligible probability). $\mathcal{B}$ also chooses a random value $\tilde{u}_3$ and implicitly sets $u_3 = -w_2 c^{q-1} - \ldots - w_n c^{q-n+1} + f c^{-1} \tilde{u}_3$. This is properly distributed because $\tilde{u}_3$ is random (and fc^{-1} is nonzero with all but negligible probability).

We observe that

$$(t_3 + u_3)\vec{b}_3^* = (w_1 d c^{q+1} + df\tilde{u}_3)\vec{d}_3^*.$$

$\mathcal{B}$ forms the semi-functional part of K as: $T^{w_1 \vec{d}_3^*} \left(g^{df}\right)^{\tilde{u}_3 \vec{d}_3^*}$. If $T = g^{dc^{q+1}}$, this is equal to $g^{(t_3+u_3)\vec{b}_3^*}$, as required for a nominal semi-functional key. Otherwise, this exponent is distributed as a random multiple of $\vec{b}_3^*$, as required for a temporary semi-functional key. We also have

$$u_3 \vec{b}_{3,0}^* = (-w_2 c^q - \ldots - w_n c^{q-n+2} + f\tilde{u}_3)\vec{d}_{3,0}^*,$$

enabling $\mathcal{B}$ to produce $g^{u_3 \vec{b}_{3,0}^*}$ using the terms given in the assumption.

Now, $\mathcal{B}$ can also produce g^{t_3}, and hence can compute $g^{t_3 \vec{b}_{5,i}^*} = g^{t_3 \vec{d}_{5,i}^*}$ for each $i \in S$. We observe

$$t_3 \vec{b}_{6,i}^* = t_3 \eta_i \vec{d}_{6,i}^*,$$

and

$$t_3 \eta_i = \left(w_1 c^q + \ldots + w_n c^{q-n+1}\right) \left(\tilde{\eta}_i + \sum_{j \in J_i} c A_{j,1}/b_j + \ldots + c^n A_{j,n}/b_j\right).$$

For each $j \in J_i$, we have $\rho(j) = i$. So for $i \in S$, we have $A_j \cdot w = 0$ modulo p for every $j \in J_i$. Thus, all of the terms involving c^{q+1} cancel, and we are left with terms that can be created in the exponent from the group elements given in the assumption (note that $n \leq q$, so $2q$ is an upper bound on the powers of c involved here). This shows that $\mathcal{B}$ can create $g^{t_3 \vec{b}_{6,i}}$ for all $i \in S$, and hence can produce properly distributed semi-functional components for each K_i of the k^{th} key.

$\mathcal{B}$ can respond to the rest of $\mathcal{A}$'s key requests by producing normal keys via the normal key generation algorithm. If $T = g^{dc^{q+1}}$, then $\mathcal{B}$ has properly simulated Game_k^N. If T is distributed randomly, then $\mathcal{B}$ has properly simulated Game_k^T. Thus, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the source group q-parallel BDHE assumption. □

Lemma 23. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_k^T and Game_k for any k from 1 to Q.

Proof. This proof is almost identical to the proof of Lemma 20, except that $\mathcal{B}$ adds an additional term of $g^{\gamma \vec{b}_3^*}$ to K for the k^{th} key (where it chooses $\gamma \in \mathbb{Z}_p$ randomly). This ensures that when the τ_3 terms are not present, the k^{th} key will be a properly distributed semi-functional key. □

Lemma 24. Under the subspace assumption, no polynomial time attacker can achieve a non-negligible difference in advantage between Game_Q and Game_{final}.

Proof. Given a PPT attacker $\mathcal{A}$ achieving a non-negligible difference in advantage between Game_Q and Game_{final}, we will create a PPT algorithm $\mathcal{B}$ to break the subspace assumption. We will employ the subspace assumption with parameters $m = \mathcal{U} + 2$, $n_i = 3, k_i = 1$ for two values of i, and $n_i = 6, k_i = 2$ for the rest of the values of i. To coincide with our notation for the construction, we will denote the bases involved in the assumption by $(\mathbb{B}, \mathbb{B}^*), (\mathbb{B}_0, \mathbb{B}_0^*) \in Dual(\mathbb{Z}_p^3)$ and $(\mathbb{B}_1, \mathbb{B}_1^*), \ldots, (\mathbb{B}_\mathcal{U}, \mathbb{B}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^6)$. $\mathcal{B}$ is given (we will ignore μ_3, g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and $T_{1,0}, \{T_{1,i}, T_{2,i}\}_{i\in[\mathcal{U}]}$ because they will not be needed):

$$G, p, g, g^{\vec{b}_1}, g^{\vec{b}_2}, g^{\vec{b}_{1,0}}, g^{\vec{b}_{2,0}}, \{g^{\vec{b}_{1,i}}, \ldots, g^{\vec{b}_{4,i}}\}_{i\in[\mathcal{U}]},$$

$$g^{\eta\vec{b}_1^*}, g^{\beta\vec{b}_2^*}, g^{\vec{b}_3^*}, g^{\eta\vec{b}_{1,0}^*}, g^{\beta\vec{b}_{2,0}^*}, g^{\vec{b}_{3,0}^*}, \{g^{\eta\vec{b}_{1,i}^*}, g^{\eta\vec{b}_{2,i}^*}, g^{\beta\vec{b}_{3,i}^*}, g^{\beta\vec{b}_{4,i}^*}, g^{\vec{b}_{5,i}^*}, g^{\vec{b}_{6,i}^*}\}_{i\in[\mathcal{U}]},$$

$$U_1 = g^{\mu_1\vec{b}_1+\mu_2\vec{b}_2+\mu_3\vec{b}_3}, U_{1,0} = g^{\mu_1\vec{b}_{1,0}+\mu_2\vec{b}_{2,0}+\mu_3\vec{b}_{3,0}},$$

$$\{U_{1,i} = g^{\mu_1\vec{b}_{1,i}+\mu_2\vec{b}_{3,i}+\mu_3\vec{b}_{5,i}}, U_{2,i} = g^{\mu_1\vec{b}_{2,i}+\mu_2\vec{b}_{4,i}+\mu_3\vec{b}_{6,i}}\}_{i\in[\mathcal{U}]},\ T_1.$$

The exponent of the unknown term T_1 is distributed either as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$, or as $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*$. It is $\mathcal{B}$'s task to determine if this τ_3 contribution is present or not.

$\mathcal{B}$ sets $(\mathbb{B}, \mathbb{B}^*)$, $(\mathbb{B}_0, \mathbb{B}_0^*)$, $\{(\mathbb{B}_i, \mathbb{B}_i^*)\}$ as the bases for the construction. It will implicitly set $\alpha_1 = \eta\tau_1$ and $\alpha_2 = \beta\tau_2$. It forms $e(g,g)^{\alpha_1}$ as $e_3(T_1, g^{\vec{b}_1})$ and $e(g,g)^{\alpha_2}$ as $e_3(T_1, g^{\vec{b}_2})$. It gives the public parameters to $\mathcal{A}$.

To create a semi-functional key for an attribute set S, $\mathcal{B}$ proceeds as follows. It chooses random values $\tilde{t}_1, \tilde{t}_2, \tilde{u}_1, \tilde{u}_2, \tilde{\gamma} \in \mathbb{Z}_p$. It implicitly sets

$t_1 = \eta\tilde{t}_1$, $t_2 = \beta\tilde{t}_2$, $u_1 = \eta\tilde{u}_1$, and $u_2 = \beta\tilde{u}_2$. It creates the key as:

$$K = T_1 \left(g^{\eta\vec{b}^*_1}\right)^{\tilde{t}_1+\tilde{u}_1} \left(g^{\beta\vec{b}^*_2}\right)^{\tilde{t}_2+\tilde{u}_2} \left(g^{\vec{b}^*_3}\right)^{\tilde{\gamma}},$$

$$K_0 = \left(g^{\eta\vec{b}^*_{1,0}}\right)^{\tilde{u}_1} \left(g^{\beta\vec{b}^*_{2,0}}\right)^{\tilde{u}_2},$$

$$K_i = \left(g^{\eta\vec{b}^*_{1,i}}\right)^{\tilde{t}_1} \left(g^{\eta\vec{b}^*_{2,i}}\right)^{\tilde{t}_1} \left(g^{\beta\vec{b}^*_{3,i}}\right)^{\tilde{t}_2} \left(g^{\beta\vec{b}^*_{4,i}}\right)^{\tilde{t}_2} \ \forall i \in S.$$

We note that the multiple of $\vec{b}^*_3$ appearing in the exponent of K is either equal to $\tilde{\gamma}$ or $\tilde{\gamma} + \tau_3$, depending on the nature of T_1. Either way, this is a properly distributed semi-functional key (whose distribution is independent of τ_3 even if it is present).

To create the semi-functional ciphertext for some $n \times \ell$ access matrix (A, ρ), $\mathcal{B}$ can use the same procedure employed in the proof of Lemma 20 to use the U terms to provide the semi-functional components. We repeat the description of this procedure below for the reader's convenience. The only difference here comes in computing the blinding factor for M'.

$\mathcal{B}$ chooses random values $\tilde{r}^1_j, \tilde{r}^2_j, \tilde{r}^3_j \in \mathbb{Z}_p$ for each j from 1 to ℓ. It also chooses a random vector $v \in \mathbb{Z}^n_p$ with first entry equal to 1, and random vectors $\tilde{v}_1, \tilde{v}_2 \in \mathbb{Z}^n_p$ with first entries equal to 0. It implicitly sets $s_1 = \mu_1$, $s_2 = \mu_2$, $s_3 = \mu_3$, $v_1 = s_1 v + \tilde{v}_1$, $v_2 = s_2 v + \tilde{v}_2$, $v_3 = s_3 v$, $r^1_j = \mu_1 \tilde{r}^3_j + \tilde{r}^1_j$, $r^2_j = \mu_2 \tilde{r}^3_j + \tilde{r}^2_j$, and $r^3_j = \mu_3 \tilde{r}^3_j$. We note that these values are properly distributed. The ciphertext is formed as:

$$M' = M_b e_3(U_1, T_1),\ C = U_1,\ C_0 = U_{1,0},$$

$$C_j = \left(g^{\vec{b}_{1,\rho(j)}}\right)^{A_j \cdot \tilde{v}_1 + \tilde{r}^1_j} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{-\tilde{r}^1_j} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{A_j \cdot \tilde{v}_2 + \tilde{r}^2_j} \left(g^{\vec{b}_{4,\rho(j)}}\right)^{-\tilde{r}^2_j} U^{A_j \cdot v + \tilde{r}^3_j}_{1,\rho(j)} U^{-\tilde{r}^3_j}_{2,\rho(j)},$$

for all j from 1 to ℓ.

If the exponent of T_1 is equal to $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^*$, then we have

$$e_3(U_1, T_1) = e(g, g)^{(\alpha_1 s_1 + \alpha_2 s_2)},$$

and hence we have a properly distributed semi-functional encryption of M_b, as required in Game_Q. If instead the exponent of T_1 is equal to $\tau_1\eta\vec{b}_1^* + \tau_2\beta\vec{b}_2^* + \tau_3\vec{b}_3^*$, then we have

$$e_3(U_1, T_1) = e(g, g)^{(\alpha_1 s_1 + \alpha_2 s_2 + \mu_3\tau_3)}.$$

Since τ_3 is random (and independent of the semi-functional keys and the rest of the ciphertext), this blinding factor is distributed as a freshly random group element of G_T (note that μ_3 is nonzero with all but negligible probability). Therefore the ciphertext is distributed as a semi-functional encryption of a random message, as required in Game_{final}. Thus, $\mathcal{B}$ can leverage $\mathcal{A}$'s non-negligible difference in advantage between these games to achieve a non-negligible advantage against the subspace assumption. □

Combining Lemmas 4 and 19 - 24, Theorem 18 follows.

Chapter 7

A Multi-Authority CP-ABE System

We now present a multi-authority attribute-based encryption system. In our system, any party can become an authority and there is no requirement for any global coordination other than the creation of an initial set of common reference parameters. A party can simply act as an authority by creating a public key and issuing private keys to different users that reflect their attributes. Different authorities need not even be aware of each other. We use the Chase [22] concept of global identifiers to "link" private keys together that were issued to the same user by different authorities. A user can encrypt data in terms of any LSSS matrix over attributes issued from any chosen set of authorities.

Finally, our system does not require any central authority. We thus avoid the performance bottleneck incurred by relying on a central authority, which makes our system more scalable. We also avoid placing absolute trust in a single designated entity which must remain active and uncorrupted throughout the lifetime of the system. This is a crucial improvement for efficiency as well as security, since even a central authority that remains uncorrupted may occasionally fail for benign reasons, and a system that constantly relies

on its participation will be forced to remain stagnant until it can be restored. In our system, authorities can function entirely independently, and the failure or corruption of some authorities will not affect the operation of functioning, uncorrupted authorities. This makes our system more robust than previous approaches.

Remark 25. The original version of this scheme in [51] was constructed in composite order bilinear groups, and as a result it required the common reference parameters to be created during a trusted setup. This was needed in order for the factorization of the group order to remain secret. Translating the system into prime order groups using the DPVS framework avoids this feature. This was first observed by Okamoto and Takashima, who also provide a prime order analog of our multi-authority scheme in [61].

Challenges and Our Techniques In the multi-authority setting, we want to satisfy the simultaneous goals of autonomous key generation and collusion resistance. The requirement of autonomous key generation means that established techniques for key randomization cannot be applied since there is no one party to compile all the pieces together. Furthermore, in our system each component may come from a different authority, where such authorities have no coordination and are possibly not even aware of each other and there is no preset access structure.[1]

[1]Prior works [22, 23] assumed coordination ahead of time between different authorities and required a limited access structure.

In constructing our system, our central technical hurdle is to make it collusion resistant. Our single authority ABE systems above achieved collusion resistance when the ABE system authority "tied" together different components (representing different attributes) of a user's private key by randomizing the key. Such randomization would make the different key components compatible with each other, but not with the parts of a key issued to another user.

To replace this, we develop a novel technique for tying a user's key components together and preventing collusion attacks between users with different global identifiers. At a high level, instead of relying on one key generation call to tie all key components together, we will use a hash function on the user's global identity, GID, to manage collusion resistance across multiple key generations issued by different authorities.

In our system, we define a hash function H (modeled as a random oracle) that hashes each identity to a pair of (bilinear) group elements. We will use the group elements output from the hash function $H(\text{GID})$ as the linchpin to tie keys together. Our main idea is to structure the decryption mechanism at each satisfied row A_j in the access matrix (A, ρ) such that a user will recover a target group element of the form $e(g,g)^{A_j \cdot v} \cdot e(g, H^1_{\text{GID}})^{A_j \cdot w^1} e(g, H^2_{\text{GID}})^{A_j \cdot w^2}$. Here, $(H^1_{\text{GID}}, H^2_{\text{GID}})$ denote the pair of elements output by $H(\text{GID})$, v will be a vector with first entry equal to the exponent of the blinding factor on the message (denoted by s), and each of w^1, w^2 will be a vector with first entry equal to 0. This structure allows for the decryption algorithm to both reconstruct

the main secret s and to unblind it in parallel. If a user with a particular identifier GID satisfies the access matrix, he can reconstruct s in the exponent by raising the group elements to the proper exponents. This operation will simultaneously reconstruct the shares of 0 and thus the $e(g, H^1_{\text{GID}})$, $e(g, H^2_{\text{GID}})$ terms will cancel out. Intuitively, if two users with different global identifiers GID, GID' attempt to collude, the cancelation will not work since the w^1, w^2 shares in the exponents will have different bases.

Remark 26. The reason we have H output a random pair of group elements and have two vectors sharing 0 is that it allows us to embed 2-dimensional spaces from the subspace assumption into these terms. In the original composite order scheme, this duplication was not needed.

7.1 Construction

For simplicity, we will conflate notation for authorities and attributes and assume that each authority controls exactly one attribute (we will retain this simplification in our security proof as well). In practice, we would allow a single authority to control several attributes, and the authority would run the setup algorithm below for each attribute (so each attribute would have its own public key and secret key). We assume that messages to be encrypted are elements of the target group G_T.

Global Setup$(\lambda) \rightarrow$ GP The global setup algorithm chooses a bilinear group G of prime order p and a generator g. It also defines a hash function H

mapping global identifiers to pairs of elements in G, so $H : \{0,1\}^* \to G \times G$. The global parameters are:

$$\text{GP} := \{G, p, g, H\}.$$

Authority Setup(GP$)$ $\to$ SK, PK Each authority (indexed by i) chooses random dual orthonormal bases $(\mathbb{B}_i, \mathbb{B}_i^*) \xleftarrow{R} \mathit{Dual}(\mathbb{Z}_p^{12})$ and two uniformly random exponents $\alpha_i^1, \alpha_i^2 \in \mathbb{Z}_p$. It publishes the public parameters:

$$\text{PK} := \{e(g,g)^{\alpha_i^1},\ e(g,g)^{\alpha_i^2},\ g^{\vec{b}_{1,i}},\ g^{\vec{b}_{2,i}},\ g^{\vec{b}_{3,i}},\ g^{\vec{b}_{4,i}}\}.$$

Its corresponding secret key is:

$$\text{SK} := \{g^{\alpha_i^1 \vec{b}_{1,i}^*},\ \vec{b}_{1,i}^*,\ \vec{b}_{2,i}^*,\ \vec{b}_{3,i}^*,\ g^{\alpha_i^2 \vec{b}_{3,i}^*},\ \vec{b}_{4,i}^*\}.$$

Encrypt($M, (A,\rho),$ GP, $\{\text{PK}\})$ $\to$ CT The encryptor chooses a uniformly random exponent $s \in \mathbb{Z}_p$. For an $\ell \times n$ access matrix (A, ρ), the encryptor chooses three random vectors $v, w^1, w^2 \in \mathbb{Z}_p^n$ subject to the constraints that the first entry of v is equal to s and the first entries of w^1 and w^2 are equal to 0. For each j from 1 to ℓ, it also chooses random values $r_j^1, r_j^2 \in \mathbb{Z}_p$. It then computes:

$$C := Me(g,g)^s,\ D_j := e(g,g)^{A_j \cdot v} e(g,g)^{\alpha_{\rho(j)}^1 r_j^1} e(g,g)^{\alpha_{\rho(j)}^2 r_j^2}\ \forall j \in [\ell],$$

$$C_j := g^{r_j^1 \vec{b}_{1,\rho(j)} + (r_j^1 + A_j \cdot w^1)\vec{b}_{2,\rho(j)} + r_j^2 \vec{b}_{3,\rho(j)} + (r_j^2 + A_j \cdot w^2)\vec{b}_{4,\rho(j)}},\ \forall j \in [\ell].$$

The ciphertext CT consists of $C, \{D_j\}, \{C_j\}$ (as well as (A, ρ)).

KeyGen$(\mathrm{GID}, \mathrm{GP}, i, \mathrm{SK}) \to \mathrm{K}_{i,\mathrm{GID}}$ To generate a secret key for a user with identity GID, authority i computes $H(\mathrm{GID}) = (H^1_{\mathrm{GID}}, H^2_{\mathrm{GID}}) \in G^2$. It forms the key as:

$$\mathrm{K}_{i,\mathrm{GID}} := g^{\alpha_i^1 \vec{b}^*_{1,i}} g^{\alpha_i^2 \vec{b}^*_{3,i}} \left(H^1_{\mathrm{GID}}\right)^{\vec{b}^*_{1,i} - \vec{b}^*_{2,i}} \left(H^2_{\mathrm{GID}}\right)^{\vec{b}^*_{3,i} - \vec{b}^*_{4,i}}.$$

Decrypt$(\mathrm{CT}, \mathrm{GP}, \{\mathrm{K}_{i,\mathrm{GID}}\}) \to M$ To decrypt a ciphertext encrypted under a $\ell \times n$ access matrix (A, ρ), an authorized user chooses constants $\omega_j \in \mathbb{Z}_p$ for j from 1 to ℓ such that $\sum_j \omega_j A_j = (1, 0, \ldots, 0)$ and ω_j is only non-zero when $\rho(j)$ is an attribute the user has a secret key for. For each j such that $\omega_j \neq 0$, the user computes:

$$F_j := D_j / e_{12}(K_{\rho(j),\mathrm{GID}}, C_j).$$

It then recovers the message as:

$$M = C / \prod_{j \text{ s.t. } \omega_j \neq 0} F_j^{\omega_j}.$$

Correctness We observe that

$$e_{12}(K_{\rho(j),\mathrm{GID}}, C_j) = e(g,g)^{\alpha^1_{\rho(j)} r^1_j} e(g,g)^{\alpha^2_{\rho(j)} r^2_j} e(g, H^1_{\mathrm{GID}})^{-A_j \cdot w^1} e(g, H^2_{\mathrm{GID}})^{-A_j \cdot w^2}.$$

Thus, each $F_j = e(g,g)^{A_j \cdot v} e(g, H^1_{\mathrm{GID}})^{A_j \cdot w^1} e(g, H^2_{\mathrm{GID}})^{A_j \cdot w^2}$, and hence

$$\prod_{j \text{ s.t. } \omega_j \neq 0} F_j^{\omega_j} = e(g,g)^{\sum_j \omega_j A_j \cdot v} = e(g,g)^s.$$

(This follows because $\sum_j \omega_j A_j$ is orthogonal to w^1 and w^2 modulo p.)

7.2 Security Proof

We will assume that the row labeling function ρ for an access matrix must be injective, the same restriction we imposed in Chapter 5. One can then employ the encoding technique described in Section 5.2 to obtain a system allowing bounded reuse of attributes.

As in the proof in Chapter 5, we will argue that nominal semi-functionality is information-theoretically hidden from the attacker. It will again be crucial to arrange for our information-theoretic argument to take place in a part of the semi-functional space that is only occupied by one key at a time. However, we no longer have a "header" term in the key to act as a structural linchpin and provide the part of the semi-functional space where all semi-functional keys simultaneously reside. To accommodate our now decentralized keys, we design our semi-functional space as having two halves: semi-functional ciphertexts will have components in both halves, while semi-functional keys will first have components in one half and then will be "switched" to having components in the other half. In this way, we can think of the semi-functional space for keys as being divided into a temporary space and a permanent space. Throughout our hybrid argument, at most one key will have components in the temporary space. This enables us to execute our information-theoretic argument in the temporary space, and then switch and save our progress key by key in the permanent space. Concretely, our permanent semi-functional space in the keys will be spanned by $\vec{b}^*_{5,i}, \ldots, \vec{b}^*_{8,i}$ for each attribute i, while our temporary semi-functional space will be spanned by $\vec{b}^*_{9,i}, \ldots, \vec{b}^*_{12,i}$.

We prove:

Theorem 27. Under the decisional linear assumption, the multi-authority CP-ABE system constructed in Section 7.1 is fully secure in the sense of Definition 10 when H is a random oracle, under the restriction that for an access matrix (A, ρ), the row labeling function ρ must be injective.

We must first define our semi-functional objects.

Semi-functional Ciphertexts A semi-functional ciphertext for a $\ell \times n$ LSSS matrix (A, ρ) is created as follows. First, a normal ciphertext $C, \{D_j, C_j\}$ is created by running the normal encryption algorithm. We let $B \subseteq [\ell]$ denote the set of indices j such that $\rho(j)$ is an attribute belonging to a corrupted authority, and we let $\overline{B} \subseteq [\ell]$ denote the complement. We note that for a valid matrix (A, ρ) in the security game, the rows indexed by B cannot include the vector $(1, 0, \ldots, 0)$ in their span.

Then, random values $\gamma_j^1, \gamma_j^2, \gamma_j^3, \gamma_j^4 \in \mathbb{Z}_p$ are chosen for each $j \in \overline{B}$, along with vectors $\theta^1, \theta^2, \theta^3, \theta^4 \in \mathbb{Z}_p^n$ chosen randomly up to the constraint that they are orthogonal to all rows A_j such that $j \in B$ (note their first entries will be random). The semi-functional ciphertext is formed by retaining the values of C and $\{D_j\}$ and multiplying each C_j for $j \in \overline{B}$ by

$$g^{\gamma_j^1 \vec{b}_{5,\rho(j)} + (\gamma_j^1 + A_j \cdot \theta^1)\vec{b}_{6,\rho(j)} + \gamma_j^2 \vec{b}_{7,\rho(j)} + (\gamma_j^2 + A_j \cdot \theta^2)\vec{b}_{8,\rho(j)}}$$

and also by

$$g^{\gamma_j^3 \vec{b}_{9,\rho(j)} + (\gamma_j^3 + A_j \cdot \theta^3)\vec{b}_{10,\rho(j)} + \gamma_j^4 \vec{b}_{11,\rho(j)} + (\gamma_j^4 + A_j \cdot \theta^4)\vec{b}_{12,\rho(j)}}.$$

This essentially creates two additional copies of the normal system in the spaces spanned by $\vec{b}_{5,\rho(j)}, \ldots, \vec{b}_{8,\rho(j)}$ and $\vec{b}_{9,\rho(j)}, \ldots, \vec{b}_{12,\rho(j)}$ for each $j \in \overline{B}$. Note that for these copies, $\theta^1, \ldots, \theta^4$ share *random* values, not zero. The terms C_j for $j \in B$ are left unchanged.

Permanent Semi-functional Keys A permanent semi-functional key for an attribute set S is created as follows. First, a normal key $\{K_{i,\mathrm{GID}}\}$ is created by running the normal key generation algorithms for each attribute $i \in S$. Then random exponents $\xi^1_{\mathrm{GID}}, \xi^2_{\mathrm{GID}} \in \mathbb{Z}_p$ are chosen. Each $K_{i,\mathrm{GID}}$ is multiplied by:

$$g^{\xi^1_{\mathrm{GID}}\vec{b}^*_{5,i} - \xi^1_{\mathrm{GID}}\vec{b}^*_{6,i} + \xi^2_{\mathrm{GID}}\vec{b}^*_{7,i} - \xi^2_{\mathrm{GID}}\vec{b}^*_{8,i}}.$$

This creates an additional copy of the normal system in the permanent semi-functional space, with $g^{\xi^1_{\mathrm{GID}}}$ playing the the role of H^1_{GID} and $g^{\xi^2_{\mathrm{GID}}}$ playing the role of H^2_{GID}.

Temporary Semi-functional Keys A temporary semi-functional key for an attribute set S is created as follows. First, a normal key $\{K_{i,\mathrm{GID}}\}$ is created by running the normal key generation algorithms for each attribute $i \in S$. Then random exponents $\xi^3_{\mathrm{GID}}, \xi^4_{\mathrm{GID}} \in \mathbb{Z}_p$ are chosen. Each $K_{i,\mathrm{GID}}$ is multiplied by:

$$g^{\xi^3_{\mathrm{GID}}\vec{b}^*_{9,i} - \xi^3_{\mathrm{GID}}\vec{b}^*_{10,i} + \xi^4_{\mathrm{GID}}\vec{b}^*_{11,i} - \xi^4_{\mathrm{GID}}\vec{b}^*_{12,i}}.$$

This creates an additional copy of the normal system in the temporary semi-functional space, with $g^{\xi^3_{\mathrm{GID}}}$ playing the the role of H^1_{GID} and $g^{\xi^4_{\mathrm{GID}}}$ playing the

role of H^2_{GID}.

We let Game_{real} denote the real security game defined in Section 4.4.1. We let Q denote the total number of GID's that are queried by the attacker. For each such GID, the attacker may query for several attributes (these queries may be arbitrarily interspersed with other queries). For each k from 1 to Q, we refer to the "k^{th} key" as the set of all attribute keys given to the attacker corresponding to the k^{th} queried GID value. In all of the games below, keys will be grouped in this way by GID value (so when we say "the first k keys", we mean keys corresponding to the first k GID values). We also define the following games:

Game$_k$ For each k from 0 to Q, Game_k is like Game_{real}, except that the ciphertext given to the attacker is semi-functional and the first k keys given to the attacker are permanent semi-functional. The remaining keys are normal.

Game$_k^T$ For each k from 1 to Q, Game_k^T is like Game_k, except that the k^{th} key is a temporary semi-functional key.

Game$_{final}$ This is like Game_Q (everything given to the attacker is semi-functional), except that the ciphertext is a semi-functional encryption of a *random* message.

Our hybrid argument will proceed from Game_{real} to Game_0, then to Game_1^T, then to Game_1, then to Game_2^T, and so on, until arriving at Game_Q

and finally transitioning to Game_{final}. This is accomplished in the following lemmas. Though we will not repeat this in the statement of each lemma, we remind the reader that our proof takes places in the random oracle model.

Lemma 28. Under the subspace assumption, no PPT attacker can attain a non-negligible difference in advantage between Game_{real} and Game_0.

Proof. We will actually perform this transition in four (nearly) identical stages. First, we will use the subspace assumption to expand the C_j pieces (corresponding to good authorities) in the ciphertext into the span of $\vec{b}_{5,\rho(j)}, \vec{b}_{6,\rho(j)}$ in the exponent. We then repeat this process to expand into the span of $\vec{b}_{7,\rho(j)}, \vec{b}_{8,\rho(j)}$, and then into the span of $\vec{b}_{9,\rho(j)}, \vec{b}_{10,\rho(j)}$, and then finally into the span of $\vec{b}_{11,\rho(j)}, \vec{b}_{12,\rho(j)}$. By the end of this process, the ciphertext will be properly distributed as a semi-functional ciphertext. We will describe here the first stage, and the others follow analogously.

We assume we have a PPT attacker $\mathcal{A}$ who obtains a non-negligible difference in advantage between Game_{real} and a version of Game_0 where γ_j^1, θ^1 are random (for all $j \in \overline{B}$), but $\gamma_j^2, \gamma_j^3, \gamma_j^4, \theta^2, \theta^3, \theta^4$ are all zero in the challenge semi-functional ciphertext. We will create a PPT algorithm $\mathcal{B}$ attaining a non-negligible advantage against the subspace assumption with $m = \mathcal{U}$ (where $1, \ldots, \mathcal{U}$ form the universe of attributes), $n_i = 12$, and $k_i = 2$ for each i.

In order to reconcile the notation of the assumption with the notation of our construction as conveniently as possible, we will denote the bases involved in the assumption by $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^{12})$. $\mathcal{B}$ is given (we will

ignore the U terms, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, g^{\eta}, g^{\beta}, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}, g^{\vec{d}_{7,i}}, \ldots, g^{\vec{d}_{12,i}}\}_{i\in[\mathfrak{U}]},$$

$$\{g^{\eta\vec{d}^*_{1,i}}, g^{\eta\vec{d}^*_{2,i}}, g^{\beta\vec{d}^*_{3,i}}, g^{\beta\vec{d}^*_{4,i}}, g^{\vec{d}^*_{5,i}}, g^{\vec{d}^*_{6,i}}, \ldots, g^{\vec{d}^*_{12,i}}\}_{i\in[\mathfrak{U}]},$$

$$\{T_{1,i}, T_{2,i}\}_{i\in[\mathfrak{U}]}.$$

The exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i}$ respectively, or as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i} + \tau_3\vec{d}^*_{5,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i} + \tau_3\vec{d}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ sets the global parameters as G, p, g, and H will be modeled as a random oracle. We let $\mathcal{S}$ denote the set of all authorities, and $\mathcal{A}$ declares $\mathcal{S}' \subseteq \mathcal{S}$, the set of corrupted authorities. For each good authority, $\mathcal{B}$ will set its public key as follows.

For each attribute i controlled by a good authority, $\mathcal{B}$ implicitly sets the bases as:

$$\vec{b}_{1,i} = \eta\vec{d}^*_{1,i},\ \vec{b}_{2,i} = \eta\vec{d}^*_{2,i},\ \vec{b}_{3,i} = \beta\vec{d}^*_{3,i},\ \vec{b}_{4,i} = \beta\vec{d}^*_{4,i},$$

$$\vec{b}_{5,i} = \vec{d}^*_{5,i},\ \vec{b}_{6,i} = \vec{d}^*_{6,i},\ \ldots,\ \vec{b}_{12,i} = \vec{d}^*_{12,i}\ \ \forall i,$$

$$\vec{b}^*_{1,i} = \eta^{-1}\vec{d}_{1,i},\ \vec{b}^*_{2,i} = \eta^{-1}\vec{d}_{2,i},\ \vec{b}^*_{3,i} = \beta^{-1}\vec{d}_{3,i},\ \vec{b}^*_{4,i} = \beta^{-1}\vec{d}_{4,i},$$

$$\vec{b}^*_{5,i} = \vec{d}_{5,i},\ \vec{b}^*_{6,i} = \vec{d}_{6,i},\ \ldots,\ \vec{b}^*_{12,i} = \vec{d}_{12,i}\ \ \forall i.$$

We note that these are properly distributed because $(\mathbb{D}_1, \mathbb{D}^*_1)$, $(\mathbb{D}_2, \mathbb{D}^*_2)$, etc. are randomly chosen. It chooses $\tilde{\alpha}^1_i, \tilde{\alpha}^2_i \in \mathbb{Z}_p$ at random, and implicitly sets

$\alpha_i^1 = \eta\tilde{\alpha}_i^1$, $\alpha_i^2 = \beta\tilde{\alpha}_i^2$. It can then produce the public key

$$e(g, g^\eta)^{\tilde{\alpha}_i^1}, e(g, g^\beta)^{\tilde{\alpha}_i^2}, g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, g^{\vec{b}_{3,i}}, g^{\vec{b}_{4,i}}$$

using the terms given in the assumption.

$\mathcal{B}$ must now answer random oracle and key queries made by $\mathcal{A}$. When $\mathcal{A}$ first queries for $H(\text{GID})$, $\mathcal{B}$ chooses two random exponents $\tilde{h}^1_{\text{GID}}, \tilde{h}^2_{\text{GID}} \in \mathbb{Z}_p$. It returns $H(\text{GID}) := (H^1_{\text{GID}}, H^2_{\text{GID}}) = ((g^\eta)^{\tilde{h}^1_{\text{GID}}}, (g^\beta)^{\tilde{h}^2_{\text{GID}}})$ to $\mathcal{A}$ and stores the values so that it can respond consistently if $\mathcal{A}$ queries $H(\text{GID})$ again.

When $\mathcal{A}$ queries for a key for an identity, attribute pair (GID, i) where i is controlled by a good authority, $\mathcal{B}$ responds as follows. If $H(\text{GID})$ has not yet been queried, it first defines $H^1_{\text{GID}}, H^2_{\text{GID}}$ as above. We observe that $\alpha_i^1 \vec{b}^*_{1,i} = \tilde{\alpha}_i^1 \vec{d}_{1,i}$ and $\alpha_i^2 \vec{b}^*_{3,i} = \tilde{\alpha}_i^2 \vec{d}_{3,i}$, while $(H^1_{\text{GID}})^{\vec{b}^*_{1,i} - \vec{b}^*_{2,i}} = (g^{\tilde{h}^1_{\text{GID}}})^{\vec{d}_{1,i} - \vec{d}_{2,i}}$ and $(H^2_{\text{GID}})^{\vec{b}^*_{3,i} - \vec{b}^*_{4,i}} = (g^{\tilde{h}^2_{\text{GID}}})^{\vec{d}_{3,i} - \vec{d}_{4,i}}$. Thus, $\mathcal{B}$ computes

$$K_{i,\text{GID}} = \left(g^{\vec{d}_{1,i}}\right)^{\tilde{\alpha}_i^1} \left(g^{\vec{d}_{3,i}}\right)^{\tilde{\alpha}_i^2} \left(g^{\vec{d}_{1,i}}\right)^{\tilde{h}^1_{\text{GID}}} \left(g^{\vec{d}_{2,i}}\right)^{-\tilde{h}^1_{\text{GID}}} \left(g^{\vec{d}_{3,i}}\right)^{\tilde{h}^2_{\text{GID}}} \left(g^{\vec{d}_{4,i}}\right)^{-\tilde{h}^2_{\text{GID}}}.$$

At some point, $\mathcal{A}$ will declare messages M_0, M_1 and an $\ell \times n$ access matrix (A, ρ). We let $B \subseteq [\ell]$ denote the set of indices j such that $\rho(j)$ is an attribute controlled by a corrupt authority, and $\overline{B}$ denote its complement. It must be the case that the span of the rows A_j of A with $j \in B$ does not include $(1, 0, \ldots, 0)$. For all $j \in B$, $\mathcal{A}$ also provides the public parameters $e(g, g)^{\alpha^1_{\rho(j)}}, e(g, g)^{\alpha^2_{\rho(j)}}, g^{\vec{b}_{1,\rho(j)}}, \ldots, g^{\vec{b}_{4,\rho(j)}}$.

$\mathcal{B}$ forms the challenge ciphertext as follows. It first chooses a random $s \in \mathbb{Z}_p$ and a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to s. $\mathcal{B}$ chooses a

random vector $\tilde{\theta} \in \mathbb{Z}_p^n$ up to the constraint that is orthogonal to all the rows A_j such that $j \in B$ and its first entry is 0. It also chooses random vectors $\tilde{w}^1, \tilde{w}^2 \in \mathbb{Z}_p^n$ up to the constraint that their first entries are 0. It will implicitly set $w^1 = \tau_1\tilde{\theta} + \tilde{w}^1$ and $w^2 = \tau_2\tilde{\theta} + \tilde{w}^2$. Note that w^1, w^2 are properly distributed.

$\mathcal{B}$ also chooses random values $r_j^1, r_j^2 \in \mathbb{Z}_p$ for each $j \in B$. For each $j \in \overline{B}$, it chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$ and implicitly sets $r_j^1 = \tau_1\tilde{r}_j^3 + \tilde{r}_j^1$, $r_j^2 = \tau_2\tilde{r}_j^3 + \tilde{r}_j^2$. We note that r_j^1, r_j^2 are properly distributed for all values of j.

The ciphertext is computed as:

$$\begin{aligned}
C &= M_b e(g,g)^s, \\
D_j &= e(g,g)^{A_j \cdot v} e_{12}(T_{1,\rho(j)}, g^{\vec{d}_{1,\rho(j)}})^{\tilde{r}_j^3 \tilde{\alpha}_{\rho(j)}^1} e(g, g^{\eta})^{\tilde{r}_j^1 \tilde{\alpha}_{\rho(j)}^1} \\
&\quad \cdot e_{12}(T_{1,\rho(j)}, g^{\vec{d}_{3,\rho(j)}})^{\tilde{r}_j^3 \tilde{\alpha}_{\rho(j)}^2} e(g, g^{\beta})^{\tilde{r}_j^2 \tilde{\alpha}_{\rho(j)}^2} \quad \forall j \in \overline{B}, \\
C_j &= T_{1,\rho(j)}^{\tilde{r}_j^3} T_{2,\rho(j)}^{\tilde{r}_j^3 + A_j \cdot \tilde{\theta}} \left(g^{\eta \vec{d}^*_{1,\rho(j)}}\right)^{\tilde{r}_j^1} \left(g^{\eta \vec{d}^*_{2,\rho(j)}}\right)^{\tilde{r}_j^1 + A_j \cdot \tilde{w}^1} \\
&\quad \cdot \left(g^{\beta \vec{d}^*_{3,\rho(j)}}\right)^{\tilde{r}_j^2} \left(g^{\beta \vec{d}^*_{4,\rho(j)}}\right)^{\tilde{r}_j^2 + A_j \cdot \tilde{w}^2} \quad \forall j \in \overline{B},
\end{aligned}$$

$$D_j = e(g,g)^{A_j \cdot v} \left(e(g,g)^{\alpha_{\rho(j)}}\right)^{r_j^1} \left(e(g,g)^{\alpha_{\rho(j)}^2}\right)^{r_j^2} \quad \forall j \in B,$$

$$C_j = \left(g^{\vec{b}_{1,\rho(j)}}\right)^{r_j^1} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{r_j^1 + A_j \cdot \tilde{w}^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r_j^2} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r_j^2 + A_j \cdot \tilde{w}^2} \quad \forall j \in B.$$

In creating C_j for $j \in B$, we have used the fact that $A_j \cdot \tilde{\theta} = 0$ for these values of j.

If the τ_3 terms are absent from the $T_{1,\rho(j)}, T_{2,\rho(j)}$ values for $\rho(j) \in \overline{B}$, this is a properly distributed normal ciphertext. If the τ_3 are present, this is a semi-functional ciphertext with $\theta^1 = \tau_3\tilde{\theta}$, $\gamma_j^1 = \tau_3\tilde{r}_j^3$ for each $j \in \overline{B}$, and $\theta^2, \theta^3, \theta^4, \gamma_j^2, \gamma_j^3, \gamma_j^4$ all equal to 0.

We finally argue that θ^1 is properly distributed in $\mathcal{A}$'s view, even though $\tilde{\theta}$ has its first entry equal to 0. This is because for each $j \in \overline{B}$, $\rho(j)$ appears exactly once, and $\gamma_j^1 \vec{b}_{5,\rho(j)} + (\gamma_j^1 + A_j \cdot \theta^1)\vec{b}_{6,\rho(j)}$ is distributed *independently* of $A_j \cdot \theta^1$. This is because $\vec{b}^*_{5,\rho(j)}$ and $\vec{b}^*_{6,\rho(j)}$ are never revealed. Hence, by Lemma 2, each of $\vec{b}_{5,\rho(j)}$, $\vec{b}_{6,\rho(j)}$ can be multiplied by a random scalar in $\mathbb{Z}_p$, and the resulting values are identically distributed. Thus, no information about $A_j \cdot \theta^1$ is revealed for $j \in \overline{B}$, so the distribution is unaffected by our restriction that $\tilde{\theta}$ have its first entry equal to 0.

The remaining phases to bring in nonzero values $\theta^2, \theta^3, \theta^4, \gamma_j^2, \gamma_j^3, \gamma_j^4$ are nearly identical (note that in these arguments, $g^{\vec{b}_{5,\rho(j)}}, g^{\vec{b}_{6,\rho(j)}}$ will be known to $\mathcal{B}$). □

Lemma 29. Under the subspace assumption, no PPT attacker can attain a non-negligible difference in advantage between Game_{k-1} and Game_k^T for any k from 1 to Q.

Proof. We will accomplish this transition in two stages: first, we will move from a k^{th} key that is normal to a temporary semi-functional key that has a random value of ξ^3_{GID} and $\xi^4_{\text{GID}} = 0$. Then we will randomize ξ^4_{GID} to obtain a properly distributed temporary semi-functional key.

We describe only the first stage here, as the second stage follows analogously. We suppose we have a PPT attacker $\mathcal{A}$ that achieves a non-negligible difference in advantage between Game_{k-1} and a game where the k^{th} key has

a random value of ξ^3_{GID} and $\xi^4_{\text{GID}} = 0$ for some fixed k. We will build a PPT algorithm $\mathcal{B}$ that breaks the subspace assumption.

We employ the subspace assumption with $m = \mathcal{U}$, $n_i = 12$, and $k_i = 2$ for all i. We will denote the bases involved in the assumption by $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*) \in Dual(\mathbb{Z}_p^{12})$. $\mathcal{B}$ is given (we will ignore μ_3 because it will not be needed):

$$G, p, g, g^\eta, g^\beta, g^{\eta\tau_1}, g^{\beta\tau_2}, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}, g^{\vec{d}_{7,i}}, \ldots, g^{\vec{d}_{12,i}}\}_{i\in[\mathcal{U}]},$$

$$\{g^{\eta\vec{d}^*_{1,i}}, g^{\eta\vec{d}^*_{2,i}}, g^{\beta\vec{d}^*_{3,i}}, g^{\beta\vec{d}^*_{4,i}}, g^{\vec{d}^*_{5,i}}, g^{\vec{d}^*_{6,i}}, \ldots, g^{\vec{d}^*_{12,i}}\}_{i\in[\mathcal{U}]},$$

$$\{U_{1,i}, U_{2,i}\}_{i\in[\mathcal{U}]},\ \{T_{1,i}, T_{2,i}\}_{i\in[\mathcal{U}]}.$$

The exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i}$ respectively, or as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i} + \tau_3\vec{d}^*_{5,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i} + \tau_3\vec{d}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ sets the global parameters as G, p, g, and H will be modeled as a random oracle. We let $\mathcal{S}$ denote the set of all authorities, and $\mathcal{A}$ declares $\mathcal{S}' \subseteq \mathcal{S}$, the set of corrupted authorities. For each good authority i, $\mathcal{B}$ will set its public key as follows.

It implicitly sets the bases $\mathbb{B}_i, \mathbb{B}_i^*$ as:

$$\vec{b}_{1,i} = \vec{d}_{1,i}, \ldots, \vec{b}_{4,i} = \vec{d}_{4,i},\ \vec{b}_{5,i} = \vec{d}_{9,i}, \ldots, \vec{b}_{8,i} = \vec{d}_{12,i},\ \vec{b}_{9,i} = \vec{d}_{5,i}, \ldots, \vec{b}_{12,i} = \vec{d}_{8,i},$$

$$\vec{b}^*_{1,i} = \vec{d}^*_{1,i}, \ldots, \vec{b}^*_{4,i} = \vec{d}^*_{4,i},\ \vec{b}^*_{5,i} = \vec{d}^*_{9,i}, \ldots, \vec{b}^*_{8,i} = \vec{b}^*_{12,i},\ \vec{b}^*_{9,i} = \vec{d}^*_{5,i}, \ldots, \vec{b}^*_{12,i} = \vec{d}^*_{8,i}.$$

We note that $(\mathbb{B}_i, \mathbb{B}_i^*)$ is properly distributed, and is a permutation of $(\mathbb{D}_i, \mathbb{D}_i^*)$ (where vectors 5 through 8 and vectors 9 through 12 have switched places).

$\mathcal{B}$ chooses $\tilde{\alpha}_i^1, \tilde{\alpha}_i^2 \in \mathbb{Z}_p$ at random. It implicitly sets $\alpha_i^1 = \eta\tilde{\alpha}_i^1$ and $\alpha_i^2 = \beta\tilde{\alpha}_i^2$. It can then produce the public key

$$e(g,g)^{\alpha_i^1} = e(g,g^\eta)^{\tilde{\alpha}_i^1}, e(g,g)^{\alpha_i^2} = e(g,g^\beta)^{\tilde{\alpha}_i^2}, g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, g^{\vec{b}_{3,i}}, g^{\vec{b}_{4,i}}$$

using the terms given in the assumption. We note that $\mathcal{B}$ also knows $g^{\eta\vec{b}_{1,i}^*}, g^{\eta\vec{b}_{2,i}^*}$, $g^{\beta\vec{b}_{3,i}^*}, g^{\beta\vec{b}_{4,i}^*}$, and $g^{\vec{b}_{5,i}^*}, \ldots, g^{\vec{b}_{12,i}^*}$.

$\mathcal{B}$ must now answer random oracle and key queries made by $\mathcal{A}$. When $\mathcal{A}$ first queries for $H(\text{GID})$ where GID is *not* the k^{th} queried identity, $\mathcal{B}$ chooses two random exponents $\tilde{h}_{\text{GID}}^1, \tilde{h}_{\text{GID}}^2 \in \mathbb{Z}_p$. It returns $H(\text{GID}) := (H_{\text{GID}}^1, H_{\text{GID}}^2) = ((g^\eta)^{\tilde{h}_{\text{GID}}^1}, (g^\beta)^{\tilde{h}_{\text{GID}}^2})$ to $\mathcal{A}$ and stores the values so that it can respond consistently if $\mathcal{A}$ queries $H(\text{GID})$ again. If $\mathcal{A}$ submits a query for $H(\text{GID})$ where GID is the k^{th} queried identity, then $\mathcal{B}$ returns $H(\text{GID}) := (H_{\text{GID}}^1, H_{\text{GID}}^2) = (g^{\eta\tau_1}, g^{\beta\tau_2})$.

When $\mathcal{A}$ queries for a key for an identity, attribute pair (GID, i) where i is controlled by a good authority and GID is *not* among the first k queried identities, $\mathcal{B}$ produces a normal key as follows. If $H(\text{GID})$ has not yet been queried, it first defines $H_{\text{GID}}^1, H_{\text{GID}}^2$ as above. It then computes:

$$K_{i,\text{GID}} = \left(g^{\eta\vec{d}_{1,i}^*}\right)^{\tilde{\alpha}_i^1+\tilde{h}_{\text{GID}}^1} \left(g^{\eta\vec{d}_{2,i}^*}\right)^{-\tilde{h}_{\text{GID}}^1} \left(g^{\beta\vec{d}_{3,i}^*}\right)^{\tilde{\alpha}_i^2+\tilde{h}_{\text{GID}}^2} \left(g^{\beta\vec{d}_{4,i}^*}\right)^{-\tilde{h}_{\text{GID}}^2}.$$

When $\mathcal{A}$ queries for a key for an identity, attribute pair (GID, i) where i is controlled by a good authority and GID is among the first $k-1$ queried

identities, $\mathcal{B}$ produces a permanent semi-functional key as follows. If $H(\mathrm{GID})$ has not yet been queried, it first defines $H^1_{\mathrm{GID}}, H^2_{\mathrm{GID}}$ as above. Upon the first key query for GID, it also chooses random values $\xi^1_{\mathrm{GID}}, \xi^2_{\mathrm{GID}} \in \mathbb{Z}_p$. It then computes:

$$\left(g^{\eta \vec{d}^*_{1,i}}\right)^{\tilde{\alpha}^1_i + \tilde{h}^1_{\mathrm{GID}}} \left(g^{\eta \vec{d}^*_{2,i}}\right)^{-\tilde{h}^1_{\mathrm{GID}}} \left(g^{\beta \vec{d}^*_{3,i}}\right)^{\tilde{\alpha}^2_i + \tilde{h}^2_{\mathrm{GID}}} \left(g^{\beta \vec{d}^*_{4,i}}\right)^{-\tilde{h}^2_{\mathrm{GID}}}$$

(for the normal components of $K_{i,\mathrm{GID}}$) and

$$\left(g^{\vec{d}^*_{9,i}}\right)^{\xi^1_{\mathrm{GID}}} \left(g^{\vec{d}^*_{10,i}}\right)^{-\xi^1_{\mathrm{GID}}} \left(g^{\vec{d}^*_{11,i}}\right)^{\xi^2_{\mathrm{GID}}} \left(g^{\vec{d}^*_{12,i}}\right)^{-\xi^2_{\mathrm{GID}}}$$

(for the permanent semi-functional components). It multiplies these two quantities to together to produce $K_{i,\mathrm{GID}}$.

When $\mathcal{A}$ makes a key query for (GID, i) where GID is the k^{th} queried identity, $\mathcal{B}$ creates the key as follows. If $H(\mathrm{GID})$ has not yet been queried, it first defines $H^1_{\mathrm{GID}}, H^2_{\mathrm{GID}}$ as above. It then computes:

$$K_{i,\mathrm{GID}} = \left(g^{\eta \vec{d}^*_{1,i}}\right)^{\tilde{\alpha}^1_i} \left(g^{\beta \vec{d}^*_{3,i}}\right)^{\tilde{\alpha}^2_i} T_{1,i} T^{-1}_{2,i}.$$

If $T_{1,i}$ and $T_{2,i}$ do not have the τ_3 terms in their exponents, then this is a normal key. If the τ_3 terms are present, then this is a temporary semi-functional key with $\xi^3_{\mathrm{GID}} = \tau_3$ and $\xi^4_{\mathrm{GID}} = 0$.

At some point, $\mathcal{A}$ will declare M_0, M_1 and request the challenge ciphertext for some $\ell \times n$ access matrix (A, ρ). We let $B \subseteq [\ell]$ denote the set of indices j such that $\rho(j)$ is controlled by a corrupted authority, and we let $\overline{B} \subseteq [\ell]$ denote its complement. For each $\rho(j) \in B$, $\mathcal{A}$ also supplies $e(g,g)^{\alpha^1_{\rho(j)}}$,

$e(g,g)^{\alpha^2_{\rho(j)}}$, $g^{\vec{b}_{1,\rho(j)}}, \ldots, g^{\vec{b}_{4,\rho(j)}}$. It must be the case that the span of the rows A_j of A with $j \in B$ does not include $(1, 0, \ldots, 0)$.

$\mathcal{B}$ creates the semi-functional challenge ciphertext as follows. (This will be quite similar to the creation of the ciphertext in the proof of the previous lemma, except that $U_{1,i}$ and $U_{2,i}$ will be used in place of $T_{1,i}$, $T_{2,i}$ and the remaining semi-functional components can be added easily.) It first chooses a random $s \in \mathbb{Z}_p$ and a random vector $v \in \mathbb{Z}_p^n$ with first entry equal to s. $\mathcal{B}$ chooses a random vector $\tilde{\theta} \in \mathbb{Z}_p^n$ up to the constraint that is orthogonal to all the rows A_j such that $j \in B$ and its first entry is 0. It also chooses random vectors $\tilde{w}^1, \tilde{w}^2 \in \mathbb{Z}_p^n$ up to the constraint that their first entries are 0. It will implicitly set $w^1 = \mu_1 \tilde{\theta} + \tilde{w}^1$ and $w^2 = \mu_2 \tilde{\theta} + \tilde{w}^2$. We note that w^1, w^2 are properly distributed.

$\mathcal{B}$ also chooses random values $r_j^1, r_j^2 \in \mathbb{Z}_p$ for each $j \in B$. For each $j \in \overline{B}$, it chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \tilde{r}_j^3 \in \mathbb{Z}_p$ and implicitly sets $r_j^1 = \mu_1 \tilde{r}_j^3 + \tilde{r}_j^1$, $r_j^2 = \mu_2 \tilde{r}_j^3 + \tilde{r}_j^2$. $\mathcal{B}$ chooses random vectors $\theta^1, \theta^2, \theta^4 \in \mathbb{Z}_p^n$ subject to the constraint that each is orthogonal to A_j for all $j \in B$, and it chooses random values $\gamma_j^1, \gamma_j^2, \gamma_j^4 \in \mathbb{Z}_p$ for each $j \in \overline{B}$. It will implicitly set $\theta^3 = \mu_3 \tilde{\theta}$ and $\gamma_j^3 = \mu_3 \tilde{r}_j^3$. It is clear that all of r_j^1, r_j^2, $\gamma_j^1, \ldots, \gamma_j^4$, $\theta^1, \theta^2, \theta^4$ are properly distributed.

The semi-functional ciphertext is computed as:

$$
\begin{aligned}
C &= M_b e(g,g)^s, \\
D_j &= e(g,g)^{A_j \cdot v} e_{12}(U_{1,\rho(j)}, g^{\eta \vec{d}^*_{1,\rho(j)}})^{\tilde{r}^3_j \tilde{\alpha}^1_{\rho(j)}} e(g, g^{\eta})^{\tilde{r}^1_j \tilde{\alpha}^1_{\rho(j)}} \\
&\quad \cdot e_{12}(U_{1,\rho(j)}, g^{\beta \vec{d}^*_{3,\rho(j)}})^{\tilde{r}^3_j \tilde{\alpha}^2_{\rho(j)}} e(g, g^{\beta})^{\tilde{r}^2_j \tilde{\alpha}^2_{\rho(j)}} \quad \forall j \in \overline{B}, \\
C_j &= U^{\tilde{r}^3_j}_{1,\rho(j)} U^{\tilde{r}^3_j + A_j \cdot \tilde{\theta}}_{2,\rho(j)} \left(g^{\vec{d}_{1,\rho(j)}}\right)^{\tilde{r}^1_j} \left(g^{\vec{d}_{2,\rho(j)}}\right)^{\tilde{r}^1_j + A_j \cdot \tilde{w}^1} \left(g^{\vec{d}_{3,\rho(j)}}\right)^{\tilde{r}^2_j} \\
&\quad \cdot \left(g^{\vec{d}_{4,\rho(j)}}\right)^{\tilde{r}^2_j + A_j \cdot \tilde{w}^2} \left(g^{\vec{d}_{9,\rho(j)}}\right)^{\gamma^1_j} \left(g^{\vec{d}_{10,\rho(j)}}\right)^{\gamma^1_j + A_j \cdot \theta^1} \left(g^{\vec{d}_{11,\rho(j)}}\right)^{\gamma^2_j} \\
&\quad \cdot \left(g^{\vec{d}_{12,\rho(j)}}\right)^{\gamma^2_j + A_j \cdot \theta^2} \left(g^{\vec{d}_{7,\rho(j)}}\right)^{\gamma^4_j} \left(g^{\vec{d}_{8,\rho(j)}}\right)^{\gamma^4_j + A_j \cdot \theta^4} \quad \forall j \in \overline{B}, \\
D_j &= e(g,g)^{A_j \cdot v} \left(e(g,g)^{\alpha^1_{\rho(j)}}\right)^{r^1_j} \left(e(g,g)^{\alpha^2_{\rho(j)}}\right)^{r^2_j} \quad \forall j \in B, \\
C_j &= \left(g^{\vec{b}_{1,\rho(j)}}\right)^{r^1_j} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{r^1_j + A_j \cdot \tilde{w}^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r^2_j} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r^2_j + A_j \cdot \tilde{w}^2} \quad \forall j \in B.
\end{aligned}
$$

In creating C_j for $j \in B$, we have used the fact that $A_j \cdot \tilde{\theta} = 0$ for these values of j.

Finally, we must argue that $\theta^3 = \mu_3 \tilde{\theta}$ is properly distributed in the attacker's view. In other words, we claim that the first entry of $\tilde{\theta}$ is information-theoretically hidden from $\mathcal{A}$. This follows from the same argument given in the proof of Lemma 15 since the collection of rows A_j such that $j \in B$ or $\rho(j)$ is queried for the k^{th} GID value cannot include the vector $(1, 0, \ldots, 0)$ in its span. For all $j \in \overline{B}$ such that $\rho(j)$ is *not* queried for the k^{th} GID value, the share $A_j \cdot \theta^3$ is information theoretically hidden since the basis vectors $\vec{b}^*_{9,\rho(j)}, \vec{b}^*_{10,\rho(j)}$ are never revealed. As in the previous formulation of this argument, we rely on the fact that ρ is an injective function. $\square$

Lemma 30. Under the subspace assumption, no PPT attacker can attain a non-negligible difference in advantage between Game_k^T and Game_k for any k from 1 to Q.

Proof. We will accomplish the transition from a temporary semi-functional key to a permanent semi-functional key in four stages: first, we will move from a key that has random values of $\xi_{\text{GID}}^3, \xi_{\text{GID}}^4$ and $\xi_{\text{GID}}^1 = \xi_{\text{GID}}^2 = 0$ to a key that has random values of $\xi_{\text{GID}}^1, \xi_{\text{GID}}^3, \xi_{\text{GID}}^4$ and only $\xi_{\text{GID}}^2 = 0$. Next, we randomize ξ_{GID}^2, obtaining a key with both temporary and permanent semi-functional components. We then keep $\xi_{\text{GID}}^1, \xi_{\text{GID}}^2$ random while zeroing out ξ_{GID}^3 and ξ_{GID}^4 one at a time.

We begin with the first stage. We suppose that we have a PPT attacker $\mathcal{A}$ that achieves a non-negligible difference in advantage between Game_k^T and a game where the k^{th} key has random values of $\xi_{\text{GID}}^1, \xi_{\text{GID}}^3, \xi_{\text{GID}}^4$ while $\xi_{\text{GID}}^2 = 0$ (for some fixed k). We will build a PPT algorithm $\mathcal{B}$ that breaks the subspace assumption.

We employ the subspace assumption with $m = \mathcal{U}$, $n_i = 12$, and $k_i = 2$ for all i. We will denote the bases involved in the assumption by $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*) \in \mathit{Dual}(\mathbb{Z}_p^{12})$. $\mathcal{B}$ is given (we will ignore g^η, g^β, $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}, g^{\vec{d}_{7,i}}, \ldots, g^{\vec{d}_{12,i}}\}_{i \in [\mathcal{U}]},$$

$$\{g^{\eta\vec{d}^*_{1,i}}, g^{\eta\vec{d}^*_{2,i}}, g^{\beta\vec{d}^*_{3,i}}, g^{\beta\vec{d}^*_{4,i}}, g^{\vec{d}^*_{5,i}}, g^{\vec{d}^*_{6,i}}, \ldots, g^{\vec{d}^*_{12,i}}\}_{i \in [\mathcal{U}]},$$

$$\{U_{1,i}, U_{2,i}\}_{i\in[\mathfrak{U}]},\ \{T_{1,i}, T_{2,i}\}_{i\in[\mathfrak{U}]}.$$

The exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i}$ respectively, or as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i} + \tau_3\vec{d}^*_{5,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i} + \tau_3\vec{d}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ sets the global parameters as G, p, g, and H will be modeled as a random oracle. We let $\mathcal{S}$ denote the set of all authorities, and $\mathcal{A}$ declares $\mathcal{S}' \subseteq \mathcal{S}$, the set of corrupted authorities. For each good authority i, $\mathcal{B}$ will set its public key as follows.

It implicitly sets the bases $\mathbb{B}_i, \mathbb{B}^*_i$ as:

$$\vec{b}_{1,i} = \vec{d}_{9,i}, \ldots, \vec{b}_{4,i} = \vec{d}_{12,i},\ \vec{b}_{5,i} = \vec{d}_{5,i}, \ldots, \vec{b}_{8,i} = \vec{d}_{8,i},\ \vec{b}_{9,i} = \vec{d}_{1,i}, \ldots, \vec{b}_{12,i} = \vec{d}_{4,i},$$

$$\vec{b}^*_{1,i} = \vec{d}^*_{9,i}, \ldots, \vec{b}^*_{4,i} = \vec{d}^*_{12,i},\ \vec{b}^*_{5,i} = \vec{d}^*_{5,i}, \ldots, \vec{b}^*_{8,i} = \vec{d}^*_{8,i},\ \vec{b}^*_{9,i} = \vec{d}^*_{1,i}, \ldots, \vec{b}^*_{12,i} = \vec{d}^*_{4,i}.$$

Observe that $(\mathbb{B}_i, \mathbb{B}^*_i)$ is just a reordering of the bases $(\mathbb{D}_i, \mathbb{D}^*_i)$, where the first four and last four vectors have been interchanged within each basis. We note that $(\mathbb{B}_i, \mathbb{B}^*_i)$ are properly distributed.

$\mathcal{B}$ chooses $\alpha^1_i, \alpha^2_i \in \mathbb{Z}_p$ at random. It can then produce the public key

$$e(g,g)^{\alpha^1_i}, e(g,g)^{\alpha^2_i}, g^{\vec{b}_{1,i}}, g^{\vec{b}_{2,i}}, g^{\vec{b}_{3,i}}, g^{\vec{b}_{4,i}}$$

using the terms given in the assumption. We note that $\mathcal{B}$ also knows $g^{\vec{b}^*_{1,i}}, \ldots, g^{\vec{b}^*_{8,i}}$.

$\mathcal{B}$ must now answer random oracle and key queries made by $\mathcal{A}$. When $\mathcal{A}$ first queries for $H(\mathrm{GID})$, $\mathcal{B}$ chooses two random exponents $h^1_{\mathrm{GID}}, h^2_{\mathrm{GID}} \in \mathbb{Z}_p$.

It returns $H(\text{GID}) := (H^1_{\text{GID}}, H^2_{\text{GID}}) = (g^{h^1_{\text{GID}}}, g^{h^2_{\text{GID}}})$ to $\mathcal{A}$ and stores the values so that it can respond consistently if $\mathcal{A}$ queries $H(\text{GID})$ again.

To respond to queries (GID, i) where GID is *not* among the first k queried identities, $\mathcal{B}$ produces a normal key as follows. If $H(\text{GID})$ has not been queried previously, it first sets this as above. It then computes:

$$K_{i,\text{GID}} = \left(g^{\vec{b}^*_{1,i}}\right)^{\alpha^1_i + h^1_{\text{GID}}} \left(g^{\vec{b}^*_{2,i}}\right)^{-h^1_{\text{GID}}} \left(g^{\vec{b}^*_{3,i}}\right)^{\alpha^2_i + h^2_{\text{GID}}} \left(g^{\vec{b}^*_{4,i}}\right)^{-h^2_{\text{GID}}}.$$

To respond to queries (GID, i) where GID is among the first $k-1$ queried identities, $\mathcal{B}$ first produces a normal key as above, and then produces the permanent semi-functional components as follows. The first time a secret key for GID is requested, $\mathcal{B}$ chooses random exponents $\xi^1_{\text{GID}}, \xi^2_{\text{GID}} \in \mathbb{Z}_p$. Then for each request (GID, i), it computes

$$\left(g^{\vec{b}^*_{5,i}}\right)^{\xi^1_{\text{GID}}} \left(g^{\vec{b}^*_{6,i}}\right)^{-\xi^1_{\text{GID}}} \left(g^{\vec{b}^*_{7,i}}\right)^{\xi^2_{\text{GID}}} \left(g^{\vec{b}^*_{8,i}}\right)^{-\xi^2_{\text{GID}}}$$

and multiplies this by the normal key to produce a permanent semi-functional key.

To respond to a query (GID, i) where GID is the k^{th} queried identity, $\mathcal{B}$ again first produces a normal key as above. It implicitly sets $\xi^3_{\text{GID}} = \eta\tau_1$ and $\xi^4_{\text{GID}} = \beta\tau_2$. We note that these values are random, hence properly distributed. It multiplies the normal key by $T_{1,i}\left(T_{2,i}\right)^{-1}$ to form the semi-functional key.

If the exponents of $T_{1,i}$ and $T_{2,i}$ *do not* include τ_3 terms, then this produces a properly distributed temporary semi-functional key (with $\xi^3_{\text{GID}}, \xi^4_{\text{GID}}$ being random and $\xi^1_{\text{GID}}, \xi^2_{\text{GID}} = 0$). If the exponents of $T_{1,i}$ and $T_{2,i}$ *do* include

τ_3 terms, then this is a semi-functional key with $\xi^1_{\text{GID}} = \tau_3$, and hence all of $\xi^1_{\text{GID}}, \xi^3_{\text{GID}}, \xi^4_{\text{GID}}$ are random, while $\xi^2_{\text{GID}} = 0$.

At some point, $\mathcal{A}$ will declare M_0, M_1 and request the challenge ciphertext for some $\ell \times n$ access matrix (A, ρ). We let $B \subseteq [\ell]$ denote the set of indices j such that $\rho(j)$ is controlled by a corrupted authority, and we let $\overline{B} \subseteq [\ell]$ denote its complement. For each $\rho(j) \in B$, $\mathcal{A}$ also supplies $e(g,g)^{\alpha^1_{\rho(j)}}$, $e(g,g)^{\alpha^2_{\rho(j)}}$, $g^{\vec{b}_{1,\rho(j)}}, \ldots, g^{\vec{b}_{4,\rho(j)}}$. It must be the case that the span of the rows A_j of A with $j \in B$ does not include $(1, 0, \ldots, 0)$.

$\mathcal{B}$ creates the semi-functional challenge ciphertext as follows. It first forms the normal components of the ciphertext using the normal encryption algorithm. To form the semi-functional components, it chooses random vectors $\tilde{\theta}, \theta^2, \tilde{\theta}^3, \tilde{\theta}^4 \in \mathbb{Z}_p^n$ subject to the constraint that they are orthogonal to all rows A_j of A such that $\rho(j) \in B$ (note that the first entry of each will be randomly distributed). It will implicity set $\theta^1 = \mu_3 \tilde{\theta}$, $\theta^3 = \mu_1 \tilde{\theta} + \tilde{\theta}^3$, and $\theta^4 = \mu_2 \tilde{\theta} + \tilde{\theta}^4$. We note that these are properly distributed.

For each $j \in \overline{B}$, $\mathcal{B}$ chooses random exponents $\tilde{\gamma}_j, \gamma^2_j, \tilde{\gamma}^3_j, \tilde{\gamma}^4_j \in \mathbb{Z}_p$. It implicitly sets $\gamma^1_j = \mu_3 \tilde{\gamma}_j$, $\gamma^3_j = \mu_1 \tilde{\gamma}_j + \tilde{\gamma}^3_j$, and $\gamma^4_j = \mu_2 \tilde{\gamma}_j + \tilde{\gamma}^4_j$. We note that $\mathcal{B}$ has the values $g^{\vec{b}_{7,\rho(j)}}, \ldots, g^{\vec{b}_{12,\rho(j)}}$, but it only has access to $\vec{b}_{5,\rho(j)}, \vec{b}_{6,\rho(j)}$ in the exponents of the $U_{1,\rho(j)}, U_{2,\rho(j)}$ terms. It forms the semi-functional components of C_j as:

$$U^{\tilde{\gamma}_j}_{1,\rho(j)} U^{\tilde{\gamma}_j + A_j \cdot \tilde{\theta}}_{2,\rho(j)} \left(g^{\vec{b}_{7,\rho(j)}}\right)^{\gamma^2_j} \left(g^{\vec{b}_{8,\rho(j)}}\right)^{\gamma^2_j + A_j \cdot \theta^2} \left(g^{\vec{b}_{9,\rho(j)}}\right)^{\tilde{\gamma}^3_j}$$
$$\cdot \left(g^{\vec{b}_{10,\rho(j)}}\right)^{\tilde{\gamma}^3_j + A_j \cdot \tilde{\theta}^3} \left(g^{\vec{b}_{11,\rho(j)}}\right)^{\tilde{\gamma}^4_j} \left(g^{\vec{b}_{12,\rho(j)}}\right)^{\tilde{\gamma}^4_j + A_j \cdot \tilde{\theta}^4}.$$

Multiplying the normal components of each C_j for $j \in \overline{B}$ by these values produces a properly distributed semi-functional ciphertext.

Hence, if the τ_3 terms are not present, then $\mathcal{B}$ has properly simulated Game_k^T. If they are present, $\mathcal{B}$ has simulated a game where the k^{th} key has properly distributed components in the temporary semi-functional space as well as the first half of the permanent space. This completes the first stage of our argument.

The other three stages are nearly identical - in the second stage, we may similarly use the subspace assumption to expand the k^{th} key's semi-functional components from the span of $\vec{b}^*_{9,i}, \ldots, \vec{b}^*_{12,i}$ into the span of $\vec{b}^*_{7,i}, \ldots, \vec{b}^*_{12,i}$. Here, the simulator $\mathcal{B}$ will know the terms $g^{\vec{b}^*_{5,i}}, g^{\vec{b}^*_{6,i}}$, so the semi-functional terms in this space can be added easily. The third and fourth stages are then essentially reversals of the first two stages, with the roles of the permanent semi-functional space and temporary semi-functional space interchanged. □

Lemma 31. Under the subspace assumption, no PPT attacker can attain a non-negligible difference in advantage between Game_Q and Game_{final}.

Proof. We suppose there is a PPT algorithm $\mathcal{A}$ that attains a non-negligible difference in advantage between these two games. We will build a PPT algorithm $\mathcal{B}$ to break the subspace assumption with parameters $m = \mathcal{U}$, $n_i = 12$, and $k_i = 2$ for all i. We will denote the bases involved in the assumption by $(\mathbb{D}_1, \mathbb{D}_1^*), \ldots, (\mathbb{D}_\mathcal{U}, \mathbb{D}_\mathcal{U}^*)$.

$\mathcal{B}$ is given (we will ignore $g^{\eta\tau_1}$, $g^{\beta\tau_2}$, and μ_3 because they will not be needed):

$$G, p, g, g^{\eta}, g^{\beta}, \{g^{\vec{d}_{1,i}}, \ldots, g^{\vec{d}_{4,i}}, g^{\vec{d}_{7,i}}, \ldots, g^{\vec{d}_{12,i}}\}_{i\in[\mathfrak{U}]},$$

$$\{g^{\eta\vec{d}^*_{1,i}}, g^{\eta\vec{d}^*_{2,i}}, g^{\beta\vec{d}^*_{3,i}}, g^{\beta\vec{d}^*_{4,i}}, g^{\vec{d}^*_{5,i}}, g^{\vec{d}^*_{6,i}}, \ldots, g^{\vec{d}^*_{12,i}}\}_{i\in[\mathfrak{U}]},$$

$$\{U_{1,i}, U_{2,i}\}_{i\in[\mathfrak{U}]},\ \{T_{1,i}, T_{2,i}\}_{i\in[\mathfrak{U}]}.$$

The exponents of the unknown terms $T_{1,i}, T_{2,i}$ are distributed either as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i}$ respectively, or as $\tau_1\eta\vec{d}^*_{1,i} + \tau_2\beta\vec{d}^*_{3,i} + \tau_3\vec{d}^*_{5,i}$ and $\tau_1\eta\vec{d}^*_{2,i} + \tau_2\beta\vec{d}^*_{4,i} + \tau_3\vec{d}^*_{6,i}$ respectively. It is $\mathcal{B}$'s task to determine if these τ_3 contributions are present or not.

$\mathcal{B}$ sets the global parameters as G, p, g, and H will be modeled as a random oracle. We let $\mathcal{S}$ denote the set of all authorities, and $\mathcal{A}$ declares $\mathcal{S}' \subseteq \mathcal{S}$, the set of corrupted authorities. For each attribute i controlled by a good authority, $\mathcal{B}$ implicitly sets the bases as follows.

$\mathcal{B}$ chooses a random scalar $\psi \in \mathbb{Z}_p$ and defines the 4×4 matrix P as

$$P = \begin{pmatrix} -1 & 1 & 0 & 0 \\ 0 & 0 & -\psi^{-1} & \psi^{-1} \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix},\ P^{-1} = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & \psi & 0 & 1 \end{pmatrix},$$

We define another orthonormal bases pair $\mathbb{D}_{P,i}$, $\mathbb{D}^*_{P,i}$ as in Section 3.2.1: we let D_i be the 12×4 matrix whose columns correspond to the vectors $\vec{d}_{5,i}, \ldots, \vec{d}_{8,i}$ respectively. Then D_iP is also a 12×4 matrix, and we replace the vectors $\vec{d}_{5,i}, \ldots, \vec{d}_{8,i}$ with its columns to form $\mathbb{D}_{P,i}$. For $\mathbb{D}^*_{P,i}$, we do the

same using $(P^{-1})^t$. By Lemma 2, $\mathbb{D}_{P,i}$ and $\mathbb{D}^*_{P,i}$ are a properly distributed orthonormal bases pair. $\mathcal{B}$ implicitly sets $\mathbb{B}_i = \mathbb{D}_{P,i}$ and $\mathbb{B}^*_i = \mathbb{D}^*_{P,i}$.

To set the public keys, $\mathcal{B}$ chooses random values $\tilde{\alpha}^1_i, \tilde{\alpha}^2_i \in \mathbb{Z}_p$ and implicitly sets $\alpha^1_i = \tau_1\eta + \eta\tilde{\alpha}^1_i$ and $\alpha^2_i = \tau_2\beta + \beta\tilde{\alpha}^2_i$. It forms the public keys as:

$$\{e(g,g)^{\alpha^1_i} = e_{12}(T_{1,i}, g^{\vec{d}_{1,i}})e(g^\eta, g)^{\tilde{\alpha}^1_i}, e(g,g)^{\alpha^2_i} = e_{12}(T_{1,i}, g^{\vec{d}_{3,i}})e(g^\beta, g)^{\tilde{\alpha}^2_i},$$
$$g^{\vec{b}_{1,i}} = g^{\vec{d}_{1,i}}, \ldots, g^{\vec{b}_{4,i}} = g^{\vec{d}_{4,i}}\}$$

using the terms given in the assumption (for each attribute i controlled by a good authority). It gives these to $\mathcal{A}$.

$\mathcal{B}$ must now answer random oracle and key queries made by $\mathcal{A}$. When $\mathcal{A}$ first queries for $H(\text{GID})$, $\mathcal{B}$ chooses two random exponents $\tilde{h}^1_{\text{GID}}, \tilde{h}^2_{\text{GID}} \in \mathbb{Z}_p$. It returns $H(\text{GID}) := (H^1_{\text{GID}}, H^2_{\text{GID}}) = ((g^\eta)^{\tilde{h}^1_{\text{GID}}}, (g^\beta)^{\tilde{h}^2_{\text{GID}}})$ to $\mathcal{A}$ and stores the values so that it can respond consistently if $\mathcal{A}$ queries $H(\text{GID})$ again.

When $\mathcal{A}$ queries for a key for an identity, attribute pair (GID, i) where i is controlled by a good authority, $\mathcal{B}$ responds as follows. The first time GID is queried, $\mathcal{B}$ chooses random values $\tilde{\xi}^1_{\text{GID}}, \tilde{\xi}^2_{\text{GID}} \in \mathbb{Z}_p$. It forms the secret key as:

$$\begin{aligned} K_{i,\text{GID}} &= T_{1,i}\left(g^{\eta\vec{d}^*_{1,i}}\right)^{\tilde{\alpha}^1_i + \tilde{h}^1_{\text{GID}}}\left(g^{\eta\vec{d}^*_{2,i}}\right)^{-\tilde{h}^1_{\text{GID}}}\left(g^{\beta\vec{d}^*_{3,i}}\right)^{\tilde{\alpha}^2_i + \tilde{h}^2_{\text{GID}}} \\ &\quad \cdot \left(g^{\beta\vec{d}^*_{4,i}}\right)^{-\tilde{h}^2_{\text{GID}}}\left(g^{\vec{d}^*_{5,i}}\right)^{-\tilde{\xi}^1_{\text{GID}}}\left(g^{\vec{d}^*_{6,i}}\right)^{-\tilde{\xi}^2_{\text{GID}}}. \end{aligned}$$

Recalling that $\mathbb{B}^*_i = \mathbb{D}^*_{P,i}$, we can calculate the coefficients of $\vec{b}^*_{5,i}, \ldots, \vec{b}^*_{8,i}$ in the exponent vector of $K_{i,\text{GID}}$. We let $-\xi^1_{\text{GID}}$ denote the coefficient of $\vec{d}^*_5$:

this ξ^1_{GID} will be equal to $\tilde{\xi}^1_{\mathrm{GID}}$ when the τ_3 term is absent from $T_{1,i}$ and will be equal to $\tilde{\xi}^1_{\mathrm{GID}} - \tau_3$ otherwise. (Either way, ξ^1_{GID} is properly distributed.) The coefficients of $\vec{b}^*_{5,i}, \ldots, \vec{b}^*_{8,i}$ can then be computed as:

$$P^t \cdot \begin{pmatrix} -\xi^1_{\mathrm{GID}} \\ -\tilde{\xi}^2_{\mathrm{GID}} \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} \xi^1_{\mathrm{GID}} \\ -\xi^1_{\mathrm{GID}} \\ \xi^2_{\mathrm{GID}} \\ -\xi^2_{\mathrm{GID}} \end{pmatrix},$$

where $\xi^2_{\mathrm{GID}} = \psi^{-1}\tilde{\xi}^2_{\mathrm{GID}}$. (We note that $(-\xi^1_{\mathrm{GID}}, -\tilde{\xi}^2_{\mathrm{GID}}, 0, 0)$ are the coefficients of the exponent vector in terms of the basis $\mathbb{D}^*_i$. To obtain the coefficients in terms of the basis $\mathbb{D}^*_{P,i}$, we multiply by P^t.) This shows that the keys produced by $\mathcal{B}$ are properly distributed as permanent semi-functional keys.

At some point, $\mathcal{A}$ declares messages M_0, M_1 and a $\ell \times n$ access matrix (A, ρ). We let $B \subseteq [\ell]$ denote the set of indices j such that $\rho(j)$ is controlled by a corrupt authority and $\overline{B} \subseteq [\ell]$ denote its complement. $\mathcal{A}$ also declares the public keys for all $\rho(j)$ such that $j \in B$. To encrypt M_b, $\mathcal{B}$ proceeds as follows. It chooses a random $s \in \mathbb{Z}_p$ and a random vector $v \in \mathbb{Z}_p^n$ with s as its first entry. It chooses random vectors $\tilde{w}^1, \tilde{w}^2 \in \mathbb{Z}_p^n$ with first entries equal to 0. It chooses random vectors $\tilde{\theta}, \theta^3, \theta^4 \in \mathbb{Z}_p^n$ subject to the constraint that each is orthogonal to all rows A_j of A with $j \in B$. It chooses a random vector $\tilde{w} \in \mathbb{Z}_p^n$ with first entry equal to 0 also subject to the constraint that it is orthogonal to all rows A_j of A with $j \in B$. It will implicitly set $w^1 = \mu_1\tilde{w} + \tilde{w}^1$, $w^2 = \mu_2\tilde{w} + \tilde{w}^2$, $\theta^1 = \mu_3\tilde{\theta}$, and $\theta^2 = \psi\mu_3(\tilde{\theta} + \tilde{w})$. We note that w^1, w^2 are distributed as random vectors with first entry equal to 0, while $\theta^1, \ldots, \theta^4$ are distributed as random vectors orthogonal to all A_j such that $j \in B$ (note that ψ, μ_3 are nonzero with

all but negligible probability).

For all $j \in B$, $\mathcal{B}$ chooses random values $r_j^1, r_j^2 \in \mathbb{Z}_p$. For all $j \in \overline{B}$, $\mathcal{B}$ chooses random values $\tilde{r}_j^1, \tilde{r}_j^2, \gamma_j^1, \ldots, \gamma_j^4 \in \mathbb{Z}_p$ and implicitly sets $r_j^1 = \tilde{r}_j^1 + \mu_1 A_j \cdot \tilde{\theta}$ and $r_j^2 = \tilde{r}_j^2 + \mu_2 A_j \cdot \tilde{\theta}$. It computes the ciphertext as:

$$
\begin{aligned}
C &= M_b e(g,g)^s, \\
D_j &= e(g,g)^{A_j \cdot v} e(g^\eta, g)^{\tilde{r}_j^1 \tilde{\alpha}_{\rho(j)}^1} e_{12}(T_{1,\rho(j)}, g^{\vec{d}_{1,\rho(j)}})^{\tilde{r}_j^1} \\
&\quad \cdot e_{12}(U_{1,\rho(j)}, g^{\eta \vec{d}^*_{1,\rho(j)}})^{\tilde{\alpha}_{\rho(j)}^1 A_j \cdot \tilde{\theta}} e(g^\beta, g)^{\tilde{r}_j^2 \tilde{\alpha}_{\rho(j)}^2} e_{12}(T_{1,\rho(j)}, g^{\vec{d}_{2,\rho(j)}})^{\tilde{r}_j^2} \\
&\quad \cdot e_{12}(U_{1,\rho(j)}, g^{\beta \vec{d}^*_{2,\rho(j)}})^{\tilde{\alpha}_{\rho(j)}^2 A_j \cdot \tilde{\theta}} e_{12}(U_{1,\rho(j)}, T_{1,\rho(j)})^{A_j \cdot \tilde{\theta}} \ \forall j \in \overline{B}, \\
C_j &= U_{1,\rho(j)}^{A_j \cdot \tilde{\theta}} \left(g^{\vec{d}_{1,\rho(j)}}\right)^{\tilde{r}_j^1} U_{2,\rho(j)}^{A_j \cdot \tilde{\theta} + A_j \cdot \tilde{w}} \left(g^{\vec{d}_{2,\rho(j)}}\right)^{\tilde{r}_j^1 + A_j \cdot \tilde{w}^1} \left(g^{\vec{d}_{3,\rho(j)}}\right)^{\tilde{r}_j^2} \\
&\quad \cdot \left(g^{\vec{d}_{4,\rho(j)}}\right)^{\tilde{r}_j^2 + A_j \cdot \tilde{w}^2} \left(g^{\vec{d}_{7,\rho(j)}}\right)^{\gamma_j^1} \left(g^{\vec{d}_{8,\rho(j)}}\right)^{\gamma_j^2} \left(g^{\vec{d}_{9,\rho(j)}}\right)^{\gamma_j^3} \\
&\quad \cdot \left(g^{\vec{d}_{10,\rho(j)}}\right)^{\gamma_j^3 + A_j \cdot \theta^3} \left(g^{\vec{d}_{11,\rho(j)}}\right)^{\gamma_4^j} \left(g^{\vec{d}_{12,\rho(j)}}\right)^{\gamma_j^4 + A_j \cdot \theta^4} \ \forall j \in \overline{B} \\
D_j &= e(g,g)^{A_j \cdot v} \left(e(g,g)^{\alpha_{\rho(j)}^1}\right)^{r_j^1} \left(e(g,g)^{\alpha_{\rho(j)}^2}\right)^{r_j^2} \ \forall j \in B, \\
C_j &= \left(g^{\vec{b}_{1,\rho(j)}}\right)^{r_j^1} \left(g^{\vec{b}_{2,\rho(j)}}\right)^{r_j^1 + A_j \cdot \tilde{w}^1} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r_j^2} \left(g^{\vec{b}_{3,\rho(j)}}\right)^{r_j^2 + A_j \cdot \tilde{w}^2} \ \forall j \in B.
\end{aligned}
$$

In creating C_j for $j \in B$, we have used the fact that $A_j \cdot \tilde{w} = 0$ for these values of j.

We observe that the coefficients of the exponent vector of C_j for $j \in \overline{B}$ in terms of the basis vectors $\vec{b}_{5,\rho(j)}, \ldots, \vec{b}_{8,\rho(j)}$ can be computed as:

$$
P^{-1} \cdot \begin{pmatrix} \mu_3 A_j \cdot \tilde{\theta} \\ \mu_3 A_j \cdot (\tilde{\theta} + \tilde{w}) \\ \gamma_j^1 \\ \gamma_j^2 \end{pmatrix} = \begin{pmatrix} \gamma_j^1 \\ \gamma_j^1 + A_j \cdot (\mu_3 \tilde{\theta}) \\ \gamma_j^2 \\ \gamma_j^2 + A_j \cdot (\psi \mu_3 (\tilde{\theta} + \tilde{w})) \end{pmatrix}.
$$

This demonstrates the implicit assignments for θ^1, θ^2 claimed above.

Now, we examine the exponents of the D_j values for $j \in \overline{B}$. If the τ_3 term is not present on $T_{1,\rho(j)}$, D_j is equal to

$$e(g,g)^{A_j \cdot v + \alpha^1_{\rho(j)} r^1_j + \alpha^2_{\rho(j)} r^2_j},$$

as required in Game_Q. If the τ_3 term is present, then we have an extra contribution of $\tau_3 \mu_3 A_j \cdot \tilde{\theta}$. This is equivalent to replacing the sharing vector v by $v + \tau_3 \mu_3 \tilde{\theta}$. Crucially, the value of τ_3 is only involved elsewhere in ξ^1_{GID}, but its contribution is masked by the random value $\tilde{\xi}^1_{\text{GID}}$ and $\tilde{\theta}$ *has a nonzero first entry*, since the collection of rows A_j for $j \in B$ is not allowed to include the vector $(1, 0, \ldots, 0)$ in its span. Thus, $v + \tau_3 \mu_3 \tilde{\theta}$ is distributed as a truly random vector, *independent of* s. In this case, the ciphertext is properly distributed as a semi-functional encryption of a random element of G_T, independent of the value of M_b, as required in Game_{final}. Thus, $\mathcal{B}$ can leverage $\mathcal{A}$'s difference in advantage between these games to break the subspace assumption. □

Combining Lemmas 4, 28, 29, 30, and 31, Theorem 27 follows.

Chapter 8

Further Work and Future Directions

We have now presented three CP-ABE schemes and accompanying security proofs. These results demonstrate the flexibility and breadth of the dual system encryption paradigm, which can be used to prove full security under simple, standard assumptions and can also leverage more complex, parameterized assumptions to prove full security for more efficient systems. We have additionally demonstrated how it can be adapted to the functionality of multi-authority schemes. Here, we briefly discuss further work on functional encryption that we have presented elsewhere and open problems for future work.

8.1 Further Work

Unbounded ABE Schemes All of the schemes we have presented here have public parameters that scale linearly with the size of the attribute universe. This is often called "small universe ABE." There has also been prior work on "large universe ABE," which allows the size of the attribute universe to be exponential in comparison to the size of the public parameters. The first large universe ABE construction [39] was a KP-ABE scheme that was proven

selectively secure and required ciphertexts to obey a preset bound on the size of the associated attribute sets.

Requiring such a bound is an undesirable feature and is counter to the intention of allowing greater freedom by enlarging the attribute universe. In joint work with Waters [52], we provide a selectively secure large universe KP-ABE scheme without any such bounds. We expect that our techniques presented here could also be used in the large universe setting to obtain fully secure unbounded CP-ABE and KP-ABE schemes under q-type assumptions. Such schemes would allow an exponentially large attribute universe and not require any restrictions on the use of attributes or the size of attribute sets - this would maintain full security without requiring any compromises on flexibility.

Leakage Resilience Motivated by side-channel attacks on real world cryptosystems, many recent works have studied the problem of extending security proofs to instances where an attacker has access to limited side-channel information about underlying secret keys. Such information is called "leakage," and schemes that are proven to remain secure in such contexts are called "leakage-resilient." There have been many different models proposed for formalizing leakage-resilience. In the "only computation leaks" model introduced by Micali and Reyzin [56] (further studied in [28, 29, 37, 43], for example), it is assumed that leakage obtained during a computation only depends on whatever portions of the secret state are directly involved in that computation. In the bounded leakage model introduced by Akavia, Goldwasser, and Vaikun-

tanathan [4] (further studied in [5, 6, 18, 45, 57], for example), it is assumed that all memory can leak, but that the entire leakage obtained over the lifetime of a system is suitably bounded.

In the continual memory leakage model introduced concurrently by Dodis, Haralambiev, Lopez-Alt, and Wichs [26] and Brakerski, Kalai, Katz, and Vaikuntanathan [19], the most conservative approach is taken, as it is supposed that all memory can leak, all the time - the amount of leakage is only bounded "locally" within discrete time intervals. Working in this model provides the most complete defense against the variety of potential leakage attack scenarios. In this challenging setting, one must continually update secret keys to avoid an attacker leaking an entire key.

In joint work with Rouselakis and Waters [49], we show that the dual system encryption methodology provides convenient tools for proving leakage resilience alongside full security. We demonstrated this by presenting fully secure IBE, HIBE, and ABE systems resilient to bounded leakage from each of many secret keys per user, as well as master keys. This can be realized as resilience against continual leakage if we assume keys are periodically updated and no (or logarithmic) leakage is allowed during the update process.

The assumption that little or no leakage is allowed during a secret key update was shared by all early works in the continual memory leakage model [17, 19, 26, 49, 54]. Though this assumption was necessary to implement the proof strategies in these works, it is a very artificial and undesirable restriction to place on the attacker. One goal of the continual memory leakage model is

to avoid the "only computation leaks" assumption and hence obtain a more robust result. However, the intuition behind the "only computation leaks" assumption is that it is usually computation itself that causes leakage, not storage. Since an update of a secret key is a computation, assuming that this cannot leak is counter to this intuition, and merely replaces one potentially unrealistic assumption with an opposite one.

In joint work with M. Lewko and Waters [47], we present the first schemes (signatures and PKE) secure in the continual memory leakage model while allowing significant amounts of leakage on the entire secret state at all times, including *during secret key updates*. As long as the amount of leakage per key update remains bounded by a certain constant fraction of the secret key size, our schemes remain provably secure. The main technical innovation we introduce in this work is a strategy for embedding a computational challenge in an initial key generation and then designing a sequence of updates so that its effect is realized only after a certain number of updates. This strategy circumvents the primary obstacle encountered by prior proof techniques, which required embedding a computational challenge into each update process, hence preventing the simulator from performing the update "honestly" and knowing all of the values it would need to compute leakage on during an update. We extend these results in joint work with Dodis, Waters, and Wichs [27].

8.2 Future Directions

Achieving Optimal Efficiency from Simple Assumptions The open problem most directly suggested by our work here is the task of obtaining fully secure ABE systems from simple (i.e. non-q-type) assumptions, without making significant sacrifices on efficiency (we consider our encoding technique to be a significant sacrifice). Our results suggest that accomplishing this for CP-ABE in the selective setting will likely suffice, as a new selective proof technique that can be leveraged within our dual system framework would ultimately produce a proof of full security. It may also be possible to further leverage our key isolation mechanism and take more advantage of the fact that in some sense we need consider only a single key at a time.

More Expressive Functionalities The schemes we have presented here allow LSSS access structures, which can efficiently express monotonic boolean formulas, for example. These are quite versatile, but one could imagine allowing even more expressive access structures, such as circuits. Allowing circuits would enable some access policies to be expressed much more efficiently than current systems can achieve. However, moving to structures as general as circuits poses many technical challenges, and we suspect that such an advance would require many new ideas - perhaps in a different setting than bilinear groups.

Alternate Complexity Settings Given the several examples of lattice-based cryptographic systems that have built upon ideas first developed in bilinear group cryptography (e.g. [1, 21]), lattice-based cryptography is a good target for new instantiations of dual system encryption techniques. Obtaining dual system proofs in the lattice-based setting would likely extend the variety of functional encryption systems available in that realm, which are currently limited to IBE [1, 21, 35], HIBE [1, 21], and predicate encryption for inner product predicates [3]. It is also possible that functionality in the lattice-based setting could eventually surpass what is achieved (or even what is achievable) in the bilinear setting. This is the case, for instance, in the area of homomorphic encryption, as fully homomorphic encryption was achieved in the lattice setting by Gentry [33] and is not known to be possible in any other context. However, expressive functional encryption primitives like ABE present some distinct technical challenges in the lattice-based setting that have yet to be overcome.

Appendix

Appendix 1

Proof of Our q-Based Assumption in the Generic Group Model

We prove a lower bound for the complexity of our source group q-parallel BDHE assumption in the generic group model. In the generic group model [66], an adversary is not given direct access to the group, but rather only receives "handles" representing elements. It must interact with an oracle to perform the group operation (multiplication and division are both enabled) and obtain handles for new elements. It is assumed that it can only use handles which it has previously received from the oracle. We consider an experiment where an adversary is given handles for the group elements given out in the assumption as well as a handle for the challenge term T_β (here, β is a uniformly random bit). The adversary may then interact with the oracle to perform group operations and pairings and it is given the handles for the group elements resulting from these operations. Finally, the adversary must guess the bit β. The difference between the adversary's success probability and one half is defined to be its advantage. For other examples of uses of the generic group model to justify assumptions in bilinear groups, see [12, 44].

We now develop some convenient notation. We consider $c, d, f, b_1, \ldots, b_q$

as variables over $\mathbb{Z}_p$, and we define $\mathcal{M}$ to be the following set of rational functions over these variables:

$$\mathcal{M} := \{1, f, df, c^1, \ldots, c^q, c^{q+2}, \ldots, c^{2q},\ c^i/b_j\ \forall i \in [2q] \setminus \{q+1\},\ \forall j \in [q],$$

$$df b_j\ \forall j \in [q],\ df c^i b_{j'}/b_j\ \forall i \in [q],\ \forall j, j' \in [q] \text{ s.t. } j \neq j'\}.$$

These are the exponents of the group elements given out in our source group q-parallel BDHE assumption. We let $E(\mathcal{M})$ denote the set of all pairwise products of functions in $\mathcal{M}$. This set of rational functions represents the exponents of elements in G_T that can be obtained by pairing elements with exponents in $\mathcal{M}$.

We say a function T is *dependent* on a set of functions $\mathcal{S} = \{S_1, \ldots, S_k\}$ if there exist constants $r_1, \ldots, r_k \in \mathbb{Z}_p$ such that $T = r_1 S_1 + \cdots + r_k S_k$. This is an equality of functions over $\mathbb{Z}_p$, and hence must hold for *all* settings of the variables. If no such constants exist, we say that T is *independent* of $\mathcal{S}$. We begin by establishing the following lemma.

Lemma 32. For each function $M \in \mathcal{M} \cup \{dc^{q+1}\}$, the product $M \cdot dc^{q+1}$ is independent of $E(\mathcal{M}) \cup dc^{q+1}(\mathcal{M} \setminus M)$. (Here, $dc^{q+1}(\mathcal{M} \setminus M)$ denotes the set formed by removing M from $\mathcal{M}$ and then multiplying all remaining elements by dc^{q+1}.)

Proof. We note that every element of $\mathcal{M} \cup \{dc^{q+1}\}$ and every element of $E(\mathcal{M}) \cup dc^{q+1}(\mathcal{M} \setminus M)$ is a ratio of monomials. Hence, the only way $M(dc^{q+1})$ can be dependent on $E(\mathcal{M}) \cup dc^{q+1}(\mathcal{M} \setminus M)$ is if it is in fact *contained* in the

set $E(\mathcal{M}) \cup dc^{q+1}(\mathcal{M} \setminus M)$. First, we note that d^2c^{2q+2} is not contained in $E(\mathcal{M}) \cup dc^{q+1}\mathcal{M}$. For any $M \in \mathcal{M}$, it is clear that $dc^{q+1}M \notin dc^{q+1}(\mathcal{M} \setminus M)$. Thus it suffices to prove that for each M, $dc^{q+1}M \notin E(\mathcal{M})$. In other words, we must show that $E(\mathcal{M})$ does not intersect with the set $dc^{q+1}\mathcal{M}$ (the set formed by multiplying each element of $\mathcal{M}$ by dc^{q+1}. To see this, we examine the set $dc^{q+1}\mathcal{M}$.

By definition, we have that

$$dc^{q+1}\mathcal{M} = \{dc^{q+1}, dfc^{q+1}, d^2fc^{q+1}, dc^{q+2}, \ldots, dc^{2q+1}, dc^{2q+3}, \ldots, dc^{3q+1},$$

$$dc^i/b_j \ \forall j \in [q], \ \forall i \in \{q+2, \ldots, 3q+1\} \setminus \{2q+2\}, \ d^2fb_jc^{q+1} \ \forall j \in [q],$$

$$d^2fc^ib_{j'}/b_j \ \forall i \in \{q+2, \ldots, 2q+1\}, \ \forall j, j' \in [q] \text{ s.t. } j \neq j'\}.$$

We now must check if any of these are in $E(\mathcal{M})$, which is the set of pairwise products of things in $\mathcal{M}$. We observe that in $\mathcal{M}$, every occurrence of d is accompanied by f and f^{-1} never appears: so it is impossible for $E(\mathcal{M})$ to contain any elements which have higher powers of d than f. This rules out all of the elements in $dc^{q+1}\mathcal{M}$ above except for dfc^{q+1}.

To also rule out dfc^{q+1}, we consider all the possible ways it might be formed as a product of two elements of $\mathcal{M}$. Since this term contains f, one of the two factors in $\mathcal{M}$ must be a term containing f. We note that neither f or df can be one of the factors, since $dc^{q+1}, c^{q+1} \notin \mathcal{M}$. We note that an element of the form dfb_j cannot be one of the two factors, since $c^{q+1}/b_j \notin \mathcal{M}$. Similarly, an element of the form $dfc^ib_{j'}/b_j$ cannot be one of the two factors,

since $c^{q+1-i}b_j/b_{j'} \notin \mathcal{M}$. We have thus dismissed all ways the f could be obtained, and we conclude that $dfc^{q+1} \notin E(\mathcal{M})$. □

We now proceed similarly to the proof strategy in [12, 44] to establish the following theorem:

Theorem 33. For any adversary $\mathcal{A}$ that makes Q queries to the oracles computing the group operations in G, G_T and the bilinear map $e : G \times G \to G_T$, the advantage of $\mathcal{A}$ against the source group q-parallel BDHE assumption in the generic group model is at most $O\left(\frac{Q^2q}{p}\right)$.

Proof. In the real experiment, the variables $c, d, f, b_1, \ldots, b_q$ are first set randomly, and then the adversary is given handles for the group elements corresponding to the terms given out in the assumption and to T_β. $\mathcal{A}$ can then issue oracle queries for handles corresponding to products, divisions, or pairings of elements that it already has handles for. We define a new experiment in which the variables are never concretely instantiated, but instead the handles correspond to formal functions. Two elements are now given the same handle if and only if they are equal as formal functions over the variables.

This differs only from the real experiment when it happens that two formal functions are unequal, but happen to coincide for the particular choice of the variable settings. All of the functions created in the course of the experiment are linear combinations of rational functions whose numerators are polynomials of degree $\leq 4q$ and whose denominators are always among $\{b_1, \ldots, b_q\}$. Multiplying such a rational function by the product $b_1 \cdots b_q$ will

thus yield a polynomial of degree $\leq 5q$. The probability that two formal polynomials of degree $\leq 5q$ are unequal but happen to be equal for a random setting of the variables modulo p is upper bounded by $\frac{5q}{p}$ by the Schwartz-Zippel Lemma. Thus, the probability of the particular setting of the variables causing a difference between the two experiments is $O\left(\frac{Q^2 q}{p}\right)$.

Now, in this new experiment where formal variables are maintained, the only way for the adversary to have any advantage in guessing β is for it to generate a two formal functions during the course of the experiment that are the same only when β takes a particular value. In our case, this must mean that the attacker generates two functions that are equal only when the challenge term is $g^{dc^{q+1}}$. We can rearrange terms and express this equality as $dc^{q+1}h_1 = h_2$ for functions h_1 and h_2 satisfying the following constraints: h_1 must be non-zero and generated in G (so without using pairings). Thus, h_1 must be a linear combination of elements of $\mathcal{M} \cup \{dc^{q+1}\}$. Also, h_2 must be a linear combination of elements in $E(\mathcal{M})$ (note that this set includes $\mathcal{M}$ since $1 \in \mathcal{M}$). But this means that for some $M \in \mathcal{M} \cup \{dc^{q+1}\}$, $dc^{q+1}M$ is dependent on $E(\mathcal{M}) \cup dc^{q+1}(\mathcal{M} \setminus M)$, contradicting Lemma 32. $\square$

Bibliography

[1] S. Agrawal, D. Boneh, and X. Boyen. Efficient lattice (h)ibe in the standard model. In *EUROCRYPT*, pages 553–572, 2010.

[2] S. Agrawal, D. Boneh, and X. Boyen. Lattice basis delegation in fixed dimension and shorter-ciphertext hierarchical ibe. In *CRYPTO*, pages 98–115, 2010.

[3] S. Agrawal, D. M. Freeman, and V. Vaikuntanathan. Functional encryption for inner product predicates from learning with errors. In *ASIACRYPT*, pages 21–40, 2011.

[4] A. Akavia, S. Goldwasser, and V. Vaikuntanathan. Simultaneous hardcore bits and cryptography against memory attacks. In *TCC*, pages 474–495, 2009.

[5] J. Alwen, Y. Dodis, M. Naor, G. Segev, S. Walfish, and D. Wichs. Public-key encryption in the bounded-retrieval model. In *EUROCRYPT*, pages 113–134, 2010.

[6] J. Alwen, Y. Dodis, and D. Wichs. Leakage-resilient public-key cryptography in the bounded-retrieval model. In *CRYPTO*, pages 36–54, 2009.

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

[8] M. Bellare, B. Waters, and S. Yilek. Identity-based encryption secure against selective opening attack. In *TCC*, pages 235–252, 2011.

[9] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

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

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

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

[13] D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO*, pages 41–55, 2004.

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

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

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

[17] E. Boyle, G. Segev, and D. Wichs. Fully leakage-resilient signatures. In *EUROCRYPT*, pages 89–108, 2011.

[18] Z. Brakerski and S. Goldwasser. Circular and leakage resilient public-key encryption under subgroup indistinguishability - (or: Quadratic residuosity strikes back). In *CRYPTO*, pages 1–20, 2010.

[19] Z. Brakerski, Y. T. Kalai, J. Katz, and V. Vaikuntanathan. Overcoming the hole in the bucket: Public-key cryptography resilient to continual memory leakage. In *FOCS*, pages 501–510, 2010.

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

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

[22] M. Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.

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

[24] L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *ACM Conference on Computer and Communications Security*, pages 456–465, 2007.

[25] 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.

[26] Y. Dodis, K. Haralambiev, A. López-Alt, and D. Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 2010.

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

[28] S. Dziembowski and K. Pietrzak. Leakage-resilient cryptography. In *FOCS*, pages 293–302, 2008.

[29] S. Faust, E. Kiltz, K. Pietrzak, and G. N. Rothblum. Leakage-resilient signatures. In *TCC*, pages 343–360, 2010.

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

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

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

[33] C. Gentry. Fully homomorphic encryption using ideal lattices. In *STOC*, pages 169–178, 2009.

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

[35] C. Gentry, C. Peikert, and V. Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. In *STOC*, pages 197–206, 2008.

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

[37] S. Goldwasser and G. N. Rothblum. Securing computation against continuous leakage. In *CRYPTO*, pages 59–79, 2010.

[38] V. Goyal, A. Jain, O. Pandey, and A. Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP*, pages 579–591, 2008.

[39] 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.

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

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

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

[43] A. Juma and Y. Vahlis. Protecting cryptographic keys against continual leakage. In *CRYPTO*, pages 41–58, 2010.

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

[45] J. Katz and V. Vaikuntanathan. Signature schemes with bounded leakage resilience. In *ASIACRYPT*, pages 703–720, 2009.

[46] A. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.

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

[48] 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.

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

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

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

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

[53] A. Lewko and B. Waters. New proof methods for attribute-based encryption: Achieving full security through selective techniques, 2012.

[54] T. Malkin, I. Teranishi, Y. Vahlis, and M. Yung. Signatures resilient to continual leakage on memory and computation. In *TCC*, pages 89–106, 2011.

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

[56] S. Micali and L. Reyzin. Physically observable cryptography (extended abstract). In *TCC*, pages 278–296, 2004.

[57] M. Naor and G. Segev. Public-key cryptosystems resilient to key leakage. In *CRYPTO*, pages 18–35, 2009.

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

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

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

[61] T. Okamoto and K. Takashima. Decentralized attribute-based signatures. Cryptology ePrint Archive, Report 2011/701, 2011. `http://eprint.iacr.org/`.

[62] 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.

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

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

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

[66] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, pages 256–266, 1997.

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

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

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

Vita

Allison Bishop Lewko was born in South Bend, Indiana on January 26, 1984, the daughter of Marie D. Bishop and Donald C. Bishop. She received her A.B. degree in Mathematics from Princeton University in 2006 (summa cum laude) and a Certificate of Advanced Study in Mathematics from the University of Cambridge in 2007 (with distinction). She began her doctoral studies at the University of Texas at Austin in the fall of 2007. She married Mark Joseph Lewko in December, 2009.

Permanent address: 2100 Alexander Avenue
Austin, Texas 78722

This dissertation was typeset with LaTeX[†] by the author.

[†]LaTeX is a document preparation system developed by Leslie Lamport as a special version of Donald Knuth's TeX Program.