Decentralizing Attribute-Based Encryption

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

Brent Waters †
University of Texas at Austin
bwaters@cs.utexas.edu

Abstract

We propose a Multi-Authority Attribute-Based Encryption (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 ABE authority by creating a public key and issuing private keys to different users that reflect their attributes. A user can encrypt data in terms of any boolean formula over attributes issued from any chosen set of authorities. Finally, our system does not require any central authority.

In constructing our system, our largest technical hurdle is to make it collusion resistant. Prior Attribute-Based Encryption systems 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. However, in our system each component will come from a potentially different authority, where we assume no coordination between such authorities. We create new techniques to tie key components together and prevent collusion attacks between users with different global identifiers.

We prove our system secure using the recent dual system encryption methodology where the security proof works by first converting the challenge ciphertext and private keys to a semi-functional form and then arguing security. We follow a recent variant of the dual system proof technique due to Lewko and Waters and build our system using bilinear groups of composite order. We prove security under similar static assumptions to the LW paper in the random oracle model.

1 Introduction

Traditionally, we view encryption as a mechanism for a user, Alice, to confidentially encode data to a target recipient, Bob. Alice encrypts the data under the recipient's public key such that only Bob, with knowledge of his private key, can decrypt it.

However, in many applications, we find we need to share data according to an encryption policy without prior knowledge of who will be receiving the data. Suppose an administrator needs to encrypt a junior faculty member's performance review for all senior members of the computer science department or anyone in the dean's office. The administrator will want to encrypt the review with the access policy ("Computer Science" **AND** "Tenured") **OR** "Dean's Office". In this system, only users with attributes (credentials) that match this policy should be able to decrypt the document. The key challenge in building such systems is to realize security against *colluding* users. For instance, the encrypted records should not be accessible to a pair of unauthorized users, where one has the two credentials of "Tenured"

*Supported by a National Defense Science and Engineering Graduate Fellowship.

†Supported by NSF CNS-0915361, and CNS-0952692, the MURI program under AFOSR Grant No: FA9550-08-1-0352. Department of Homeland Security Grant 2006-CS-001-000001-02 (subaward 641), a Google Faculty Research award, and the Alfred P. Sloan Foundation.

and "Chemistry" and the other one has the credential of "Computer Science". Neither user is actually a tenured faculty member of the Computer Science Department.

Sahai and Waters [44] proposed a solution to the above problem that they called Attribute-Based Encryption (ABE). In an ABE system, a party encrypting data can specify access to the data as a boolean formula over a set of attributes. Each user in the system will be issued a private key from an authority that reflects their attributes (or credentials). A user will be able to decrypt a ciphertext if the attributes associated with their private key satisfy the boolean formula ascribed to the ciphertext. A crucial property of ABE systems is that they resist collusion attacks as described above.

Since the introduction of Attribute-Based Encryption, several works [8, 30, 43, 29, 23, 53, 21, 22, 37] have proposed different ABE systems and applications. In almost all ABE proposals, private keys were issued by one central authority that would need to be in a position to verify all the attributes or credentials it issued for each user in the system. These systems can be utilized to share information according a policy over attributes issued within a domain or organization, however, in many applications a party will want to share data according to a policy written over attributes or credentials issued across different trust domains and organizations. For instance, a party might want to share medical data only with a user who has the attribute of "Doctor" issued by a medical organization and the attribute "Researcher" issued by the administrators of a clinical trial. On a commercial application, two corporations such as Boeing and General Electric might both issue attributes as part of a joint project. Using current ABE systems for these applications can be problematic since one needs a *single* authority that is both able to verify attributes across different organizations and issue private keys to every user in the system.

A Simple Approach and Its Limitations We would like to realize an encryption system where a party can encrypt data for a policy written over attributes issued by different authorities. A user in the system should be able to decrypt if their attributes (possibly issued by multiple authorities) satisfy the policy specified by the ciphertext. In addition, the system should be able to express complex policies and not require coordination amongst the authorities.

An initial step towards this goal is to simply "engineer" a system by utilizing existing (Ciphertext-Policy) Attribute-Based Encryption schemes along with standard signature schemes. In this proposal, a designated "central authority" will first create a set of public parameters. Then any party wishing to become an "authority" will create a signature verification key VK that will be associated with them. A user in the system with a globally verifiable identifier GID will collect private keys for attributes that it has from different authorities.

Suppose that a user GID can demonstrate attributes X_1, X_2 to the authority with verification key VK and attribute Y to the authority with verification key VK$'$. The user will obtain his secret key as follows. First, he will obtain a signature of $\text{GID}, (X_1, X_2)$ that verifies under VK and a signature of GID, Y under VK$'$ from the two respective authorities (and any other authorities). Next, the user will present these signature and verification key pairs to the central authority. The central authority will first check that each signature verifies under the claimed verification key and that each signature is on the *same* global identifier. Using an existing ABE algorithm, it will then issue an attribute for each verification key and attribute pair. In the above example, the user will get a key with attributes "VK, X_1", "VK, X_2", and "VK', Y". We note that the operation of the central authority is agnostic to the meaning of these verification keys and attributes; indeed, it will not need to have any a priori relationship with any of the authorities.

This simple system enjoys multiple benefits. Since encryption simply uses a prior ABE system, we can achieve the same level of expressiveness and write a policy in terms of any boolean formula. The system also requires minimum coordination between separate authorities.

Any party can choose to be an authority by creating and publishing a verification key coupled with a list of attributes it will manage. Different authorities will not need to coordinate or even be aware of each other. There are several issues that will need to be dealt with in any larger system, such as the choice of an appropriate global identifier [1] or a party's decision as to which authority it trusts to issue private keys related to certain attributes. For instance, one might encrypt a policy using Experian's verification key to attest for the attribute of a good FICO (credit) score.

The major drawback of this simple engineered approach is that it requires a designated central authority. This authority must be globally trustworthy, since its failure will compromise the entire system. If we aim to build a large or even global scale system, this authority will become a common bottleneck. Spreading a central authority's keys over several machines to alleviate performance pressures might simultaneously increase the risk of key exposure.

A few works have attempted to create new cryptographic solutions to the multi-authority ABE problem. Chase [21] proposed an interesting solution that introduced the concept of using a global identifier as a "linchpin" 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. Therefore a party encrypting would be much more limited than in the simple engineering approach outlined above. Müller, Katzenbeisser, and Eckert [41, 42] give a different system with a centralized authority that realizes any LSSS access structure. Their construction builds on the Waters system [53]; their proof is limited to non-adaptive queries. The system achieves roughly the same functionality as the engineering approach above, except one can still acquire attributes from additional authorities without revisiting the central authority. Chase and Chow [22] 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. Lin et. al. [39] give a threshold based scheme that is also somewhat decentralized. The set of authorities is fixed ahead of time, and they must interact during the system setup. The system is only secure up to collusions of m users, where m is a system parameter chosen at setup such that the cost of operations and key storage scales with m.

Our Contribution We propose a new 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. (These will be created during a trusted setup.) 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 [21] 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 boolean formula [2] 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

[1] The idea of applying a global identifier in the context of multi-authority ABE was first proposed by Chase [21]. Chase adapted the concept from its use in anonymous credential systems [19]. One previously suggested candidate for a global identifier is a user's social security number.

[2] Our system actually generalizes to handle any policy that can be expressed as a Linear Secret Sharing Scheme (LSSS) or equivalently a monotone span program.

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 then the other approaches outlined above.

Challenges and Our Techniques In constructing our system, our central technical hurdle is to make it collusion resistant. Prior Attribute-Based Encryption systems 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.

In our 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.[3]

To overcome 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 (bilinear) group element. We will use the group element output from the hash function $H(\text{GID})$ as the linchpin to tie keys together. Tying keys together in this manner is more challenging than in the single authority case. Our main idea is to structure the decryption mechanism at each satisfied node 'x' in the access tree such that a user will recover a target group element of the form $e(g,g)^{\lambda_x} \cdot e(g, H(\text{GID}))^{w_x}$. This group element first contains a secret share λ_x of a secret s in the exponent, and these shares can be combined to recover the message. However, these will each be "blinded" by a share w_x which is a share of 0 in the exponent with base $e(g, H(\text{GID}))$. This structure allows for the decryption algorithm to both reconstruct the main secret and to unblind it in parallel. If a user with a particular identifier GID satisfies the access tree, he can reconstruct s in the exponent by raising the group elements to the proper exponents. However, this operation will simultaneously reconstruct the share of 0 and thus the $e(g, H(\text{GID}))$ 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_x shares will have different bases.

We prove our system secure using the recent dual system encryption methodology [52], where the security proof works by first converting the challenge ciphertexts and private keys to a semi-functional form and then arguing security. We follow a recent variant of the dual system proof technique due to Lewko and Waters [38] and build our system using bilinear groups of composite order. The absence of coordination between the authorities also introduces a new technical challenge in applying the dual system encryption methodology. Due to the decentralized nature of user's keys, the techniques employed in [37] to achieve full security for single authority ABE using dual system encryption are insufficient. We overcome this by using two semi-functional subgroups instead of one, and switching between these allows us to defeat the information-theoretic problem which is naturally encountered if one simply tries to apply

[3] Prior works [21, 22] assumed coordination ahead of time between different authorities and required a limited access structure.

the previous techniques. We prove security under similar assumptions to the LW paper in the random oracle model.

Related Work Several of the roots of Attribute-Based Encryption can be traced back to Identity Based Encryption (IBE), proposed by Shamir [45]. The first IBE schemes were constructed by Boneh and Franklin [13] and Cocks [24]. These initial systems were proven secure in the random oracle model. Other standard model solutions followed [20, 9, 10, 51, 27], along with extensions to the hierarchical IBE setting [34, 28, 11].

Attribute-based encryption was introduced by Sahai and Waters [44]. Subsequently, Goyal, Pandey, Sahai, and Waters [30] formulated two complimentary forms of ABE: Ciphertext-Policy Attribute-Based Encryption (CP-ABE) and Key-Policy Attribute-Based Encryption (KP-ABE). In a CP-ABE system, keys are associated with sets of attributes and ciphertexts are associated with access policies. In a KP-ABE system, the situation is reversed: keys are associated with access policies and ciphertexts are associated with sets of attributes. Since then, several different ABE systems have been proposed [8, 21, 23, 29, 43, 53, 22], as well as related systems [14, 2]. The problem of building ABE systems with multiple authorities was proposed by Sahai and Waters and first considered by Chase [21] and Chase and Chow [22]. Another interesting direction is the construction of "anonymous" or predicate encryption systems [36, 48, 17, 12, 1, 46, 37] where in addition to the data the encryption policy or other properties are hidden. Other works have discussed similar problems without addressing collusion resistance [3, 4, 5, 18, 40, 50]. In these systems, the data encryptor specifies an access policy such that a set of users can decrypt the data only if the union of their credentials satisfies the access policy.

Until recently, all ABE systems were proven secure in the selective model where an attacker needed to declare the structure of the challenge ciphertext *before* seeing the public parameters. Recently, Lewko, Okamoto, Sahai, Takashima and Waters [37] solved the open problem by giving the first fully secure Attribute-Based Encryption systems. Their system applied the dual system encryption methodology introduced by Waters [52] and techniques used by Lewko and Waters [38]. Our proof uses some techniques from Lewko et. al. [37], but faces new challenges from the multi-authority setting.

Organization In Section 2, we formally define multi-authority CP-ABE systems and their security. In Section 3, we give our complexity assumptions. In Section 4, we present our multi-authority CP-ABE system and outline the proof of its security. In Section 5, we discuss possible extensions of our results. We define access structures, linear secret-sharing schemes (LSSS), and composite order bilinear groups in Appendix A. We give the full details of our security proof in Appendix C.

2 Multi-authority CP-ABE

Here we give the necessary background on multi-authority CP-ABE schemes and their security definition. For background on access structures, linear secret-sharing schemes, and composite order bilinear groups, see Appendix A.

A multi-authority Ciphertext-Policy Attribute-Based Encryption system is comprised of the following five algorithms:

Global Setup(λ) $\rightarrow$ 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($M, (A, \rho), \mathrm{GP}, \{\mathrm{PK}\}) \rightarrow \mathrm{CT}$ The encryption algorithm takes in a message M, an access matrix (A, ρ), 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($\mathrm{CT}, \mathrm{GP}, \{\mathrm{K}_{i,\mathrm{GID}}\}) \rightarrow M$ The decryption algorithm takes in the global parameters, the ciphertext, and a collection of keys corresponding to attribute, identity pairs all with the same fixed identity GID. It outputs either the message M when the collection of attributes i satisfies the access matrix corresponding to the ciphertext. Otherwise, decryption fails.

Definition 1. *A multi-authority CP-ABE system is said to be correct if whenever* GP *is obtained from the global setup algorithm,* CT *is obtained from the encryption algorithm on the message* M*, and* $\{\mathrm{K}_{i,\mathrm{GID}}\}$ *is a set of keys obtained from the key generation algorithm for the same identity* GID *and for a set of attributes satisfying the access structure of the ciphertext,* $\mathit{Decrypt}(\mathrm{CT}, \mathrm{GP}, \{\mathrm{K}_{i,\mathrm{GID}}\}) = M$*.*

2.1 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 [21] and Chase and Chow [22], 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 S denote the set of authorities and 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 $S' \subseteq S$ of corrupt authorities. For good (non-corrupt) authorities in $S - S'$, 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 giving the attacker the corresponding key, $\mathrm{K}_{i,\mathrm{GID}}$.

Challenge Phase The attacker must specify two messages, M_0, M_1, and an access matrix (A, ρ). The access matrix must satisfy the following constraint. We let V denote the subset of rows of A labeled by attributes controlled by corrupt authorities. For each identity GID, we let V_{GID} denote the subset of rows of A labeled by attributes i for which the attacker has queried

(i, GID). For each GID, we require that the subspace spanned by $V \cup V_{\text{GID}}$ must not include $(1, 0, \ldots, 0)$. (In other words, the attacker cannot ask for a set of keys 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 labeling ρ. The challenger flips a random coin $\beta \in \{0, 1\}$ and sends the attacker an encryption of M_β under access matrix (A, ρ).

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 matrix (A, ρ).

Guess The attacker must submit a guess β' for β. The attacker wins if $\beta = \beta'$.

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

Definition 2. *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.*

2.2 Transformation from One-Use Multi-Authority CP-ABE

In Appendix B, we show how to construct a fully secure multi-authority CP-ABE system where attributes are used multiple times in an access matrix from a fully secure multi-authority CP-ABE system where attributes are used only once. We do this with a simple encoding technique. This same transformation was employed by [37] for (single authority) CP-ABE.

3 Our Assumptions

We now state the complexity assumptions that we will rely on to prove security for our system. These assumptions are formulated for a bilinear group G of order $N = p_1 p_2 p_3$, a product of 3 primes. For background on these groups, see Appendix A. We note that these are similar to the assumptions used in [38, 37]. While the fourth assumption is new, the first three are instances of the class of General Subgroup Decision Assumptions described in [7]. This class is defined as follows: in a bilinear group of order $N = p_1 p_2 \ldots p_n$, there is a subgroup of order $\prod_{i \in S} p_i$ for each subset $S \subseteq \{1, \ldots, n\}$. We let S_0, S_1 denote two distinct subsets. We then assume it is hard to distinguish a random element from the subgroup associated with S_0 from a random element of the subgroup associated with S_1, even if one is given random elements from subgroups associated with several subsets Z_i which each satisfy either that $S_0 \cap Z_i = \emptyset = S_1 \cap Z_i$ or $S_0 \cap Z_i \neq \emptyset \neq S_1 \cap Z_i$. We prove our four specific assumptions are generically secure in Appendix F, under the assumption that it is hard to find a nontrivial factor of the group order N.

In the assumptions below, we let $G_{p_1 p_2}$, e.g., denote the subgroup of order $p_1 p_2$ in G. When we write $g_1 \xleftarrow{R} G_{p_1}$, we mean that g_1 is chosen to be a random generator of G_{p_1} (so it is not the identity element). Similarly, when we write $T_1 \xleftarrow{R} G$, we mean that T_1 is chosen to be a random generator of G (this is not quite the same as a uniformly random element, but the distributions are negligibly close).

Assumption 1 (Subgroup decision problem for 3 primes) Given a group generator $\mathcal{G}$, we define the following distribution:

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1},$$
$$D = (\mathbb{G}, g_1),$$
$$T_1 \xleftarrow{R} G,\ T_2 \xleftarrow{R} G_{p_1}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 1 to be:

$$Adv1_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

We note that T_1 can be written (uniquely) as the product of an element of G_{p_1}, an element of G_{p_2}, and an element of G_{p_3}. We refer to these elements as the "G_{p_1} part of T_1", the "G_{p_2} part of T_1", and the "G_{p_3} part of T_1" respectively. We will use this terminology in our proofs.

Definition 3. *We say that $\mathcal{G}$ satisfies Assumption 1 if $Adv1_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

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

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e) \xleftarrow{R} \mathcal{G},$$
$$g_1, X_1 \xleftarrow{R} G_{p_1}, X_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3},$$
$$D = (\mathbb{G}, g_1, g_3, X_1X_2),$$
$$T_1 \xleftarrow{R} G_{p_1},\ T_2 \xleftarrow{R} G_{p_1p_2}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 2 to be:

$$Adv2_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 4. *We say that $\mathcal{G}$ satisfies Assumption 2 if $Adv2_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

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

$$\mathbb{G} = (N = p_1p_2p_3, G, G_T, e), \xleftarrow{R} \mathcal{G},$$
$$g_1, X_1 \xleftarrow{R} G_{p_1}, Y_2 \xleftarrow{R} G_{p_2}, X_3, Y_3 \xleftarrow{R} G_{p_3},$$
$$D = (\mathbb{G}, g_1, X_1X_3, Y_2Y_3),$$
$$T_1 \xleftarrow{R} G_{p_1p_2}, T_2 \xleftarrow{R} G_{p_1p_3}.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 3 to be:

$$Adv3_{\mathcal{G},\mathcal{A}}(\lambda) := \big|Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1]\big|.$$

Definition 5. *We say that $\mathcal{G}$ satisfies Assumption 3 if $Adv3_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

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

$$\mathbb{G} = (N = p_1 p_2 p_3, G, G_T, e), \xleftarrow{R} \mathcal{G},$$

$$g_1 \xleftarrow{R} G_{p_1}, g_2 \xleftarrow{R} G_{p_2}, g_3 \xleftarrow{R} G_{p_3}, a, b, c, d \xleftarrow{R} \mathbb{Z}_N,$$

$$D = (\mathbb{G}, g_1, g_2, g_3, g_1^a, g_1^b g_3^b, g_1^c, g_1^{ac} g_3^d),$$

$$T_1 = e(g_1, g_1)^{abc}, T_2 \xleftarrow{R} G_T.$$

We define the advantage of an algorithm $\mathcal{A}$ in breaking Assumption 4 to be:

$$Adv4_{\mathcal{G},\mathcal{A}}(\lambda) := \big| Pr[\mathcal{A}(D, T_1) = 1] - Pr[\mathcal{A}(D, T_2) = 1] \big|.$$

Definition 6. *We say that $\mathcal{G}$ satisfies Assumption 4 if $Adv4_{\mathcal{G},\mathcal{A}}(\lambda)$ is a negligible function of λ for any polynomial time algorithm $\mathcal{A}$.*

4 Our Multi-Authority CP-ABE System

Overview We now present our one-use multi-authority ciphertext-policy attribute based encryption system. We use a composite order bilinear group G, where the group order is a product of three primes: $N = p_1 p_2 p_3$. Except for the random oracle H which maps identities to random group elements, the entire system is confined to the subgroup G_{p_1} in G. The subgroups G_{p_2} and G_{p_3} are used in our security proof, which employs the dual system encryption technique. In a dual system, keys and ciphertexts can be either normal or semi-functional. Normal keys and ciphertexts in our system will be contained in the subgroup G_{p_1}, while semi-functional keys and ciphertexts will involve elements of the subgroups G_{p_2} and G_{p_3}. In other words, the subgroups G_{p_2} and G_{p_3} form the semi-functional space, which is orthogonal to the subgroup G_{p_1} where the normal keys and ciphertexts reside.

Preventing Collusion To prevent collusion attacks, our system uses the global identity to "tie" together the various attributes belonging to a specific user so that they cannot be successfully combined with another's user's attributes in decryption. More specifically, the encryption algorithm blinds the message M with $e(g_1, g_1)^s$, where g_1 is a generator of the subgroup G_{p_1}, and s is a randomly chosen value in $\mathbb{Z}_N$. The value s is then split into shares λ_x according to the LSSS matrix, and the value 0 is split into shares ω_x. The decryptor must recover the blinding factor $e(g_1, g_1)^s$ by pairing their keys for attribute, identity pairs (i, GID) with the ciphertext elements to obtain the shares of s. In doing so, the decryptor will introduce terms of the form $e(g_1, H(\text{GID}))^{\omega_x}$. If the decryptor has a satisfying set of keys with the same identity GID, these additional terms will cancel from the final result, since the ω_x's are shares of 0. If two users with different identities GID and GID$'$ attempt to collude and combine their keys, then there will be some terms of the form $e(g_1, H(\text{GID}))^{\omega_x}$ and some terms of the form $e(g_1, H(\text{GID}'))^{\omega_{x'}}$, and these will not cancel with each other, thereby preventing the recovery of $e(g_1, g_1)^s$.

4.1 Construction

Global Setup(λ) $\rightarrow$ GP In the global setup, a bilinear group G of order $N = p_1 p_2 p_3$ is chosen. The global public parameters, GP, are N and a generator g_1 of G_{p_1}. In addition, the description of a hash function $H : \{0,1\}^* \rightarrow G$ that maps global identities GID to elements of G is published. We will model H as a random oracle.

Authority Setup$(\mathrm{GP}) \to \mathrm{PK}, \mathrm{SK}$ For each attribute i belonging to the authority, the authority chooses two random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and publishes $\mathrm{PK}_j = \{e(g_1, g_1)^{\alpha_i}, g_1^{y_i} \forall i\}$ as its public key. It keeps $\mathrm{SK} = \{\alpha_i, y_i \forall i\}$ as its secret key.

Encrypt$(M, (A, \rho), \mathrm{GP}, \{\mathrm{PK}\}) \to \mathrm{CT}$ The encryption algorithm takes in a message M, an $n \times \ell$ access matrix A with ρ mapping its rows to attributes, the global parameters, and the public keys of the relevant authorities. It chooses a random $s \in \mathbb{Z}_N$ and a random vector $v \in \mathbb{Z}_N^\ell$ with s as its first entry. We let λ_x denote $A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $w \in \mathbb{Z}_N^\ell$ with 0 as its first entry. We let ω_x denote $A_x \cdot w$. For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_N$. The ciphertext is computed as:

$$C_0 = Me(g_1, g_1)^s,\ C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},\ C_{2,x} = g_1^{r_x},\ C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{\omega_x}\ \forall x.$$

KeyGen$(\mathrm{GID}, i, \mathrm{SK}, \mathrm{GP}) \to \mathrm{K}_{i,\mathrm{GID}}$ To create a key for GID for attribute i belonging to an authority, the authority computes:

$$\mathrm{K}_{i,\mathrm{GID}} = g_1^{\alpha_i} H(\mathrm{GID})^{y_i}.$$

Decrypt$(\mathrm{CT}, \{\mathrm{K}_{i,\mathrm{GID}}\}, \mathrm{GP}) \to M$ We assume the ciphertext is encrypted under an access matrix (A, ρ). To decrypt, the decryptor first obtains $H(\mathrm{GID})$ from the random oracle. If the decryptor has the secret keys $\{\mathrm{K}_{\rho(x),\mathrm{GID}}\}$ for a subset of rows A_x of A such that $(1, 0, \ldots, 0)$ is in the span of these rows, then the decryptor proceeds as follows. For each such x, the decryptor computes:

$$C_{1,x} \cdot e(H(\mathrm{GID}), C_{3,x}) / e(\mathrm{K}_{\rho(x),\mathrm{GID}}, C_{2,x}) = e(g_1, g_1)^{\lambda_x} e(H(\mathrm{GID}), g_1)^{\omega_x}.$$

The decryptor then chooses constants $c_x \in \mathbb{Z}_N$ such that $\sum_x c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$\prod_x \left(e(g_1, g_1)^{\lambda_x} e(H(\mathrm{GID}), g_1)^{\omega_x} \right)^{c_x} = e(g_1, g_1)^s.$$

(We recall that $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot w$, where $v \cdot (1, 0, \ldots, 0) = s$ and $w \cdot (1, 0, \ldots, 0) = 0$.) The message can then be obtained as:

$$M = C_0 / e(g_1, g_1)^s.$$

4.2 Security

We apply a form of the dual system encryption technique to prove security; overcoming the new challenges that arise in the multi-authority setting. In a dual system, keys and ciphertexts can either be normal or semi-functional: normal keys can decrypt semi-functional ciphertexts, semi-functional keys can decrypt normal ciphertexts, but semi-functional keys cannot decrypt semi-functional ciphertexts. The proof proceeds by a hybrid argument over a sequence of games, where we first change the challenge ciphertext to be semi-functional, and then change the keys to be semi-functional one by one. To prove that these games are indistinguishable, we must ensure that the simulator cannot test the form of the key being turned from normal to semi-functional for itself by test decrypting a semi-functional ciphertext. We avoid this problem employing the approach of [38, 37], where the simulator can only make a challenge ciphertext and key pair which is *nominally semi-functional*, meaning that both the key and ciphertext have semi-functional components, but these cancel out upon decryption. Thus, if the simulator attempts to test the form of the key for itself, decryption will succeed unconditionally.

New Challenges The existence of multiple authorities who do not coordinate with each other introduces additional technical challenges in our case. Nominal semi-functionality must be hidden from the attacker's view, which is accomplished in [37] by using temporary "blinding factors" in the semi-functional space that are active for one key at a time. Leaving these blinding factors off for the other keys prevents leakage of information that would information-theoretically reveal nominal semi-functionality in the attacker's view. However, what allows these blinding factors to be turned on and off is the stable presence of a semi-functional term attached to a single element in each key derived from the master secret key. In the multi-authority case, we do not have this sort of structural linchpin to rely on. We still need the blinding factors to hide nominal semi-functionality, but we cannot simply excise them from the other semi-functional keys to prevent their leakage. To overcome this, we use two subgroups for the semi-functional space, and instead of removing the blinding factors from the other keys, we "switch" them from one semi-functional subgroup to the other. This switch preserves semi-functionality of the keys while avoiding leakage of information about the subgroup the semi-functional components have been switched out of.

Hybrid Organization We now formally define our sequence of games. We will assume a one-use restriction on attributes throughout the proof: this means that the row labeling ρ of the challenge ciphertext access matrix (A, ρ) must be injective.

The first game, Game_{Real}, is the real security game. We next define $\text{Game}_{Real'}$, which is like the real security game, except that the random oracle maps identities GID to random elements of G_{p_1} instead of G. We now define semi-functional ciphertexts and keys, which are used only in the proof - not in the real system.

Semi-functional ciphertexts will contain terms from subgroups G_{p_2} and G_{p_3}. Semi-functional keys will be of two types: semi-functional keys of Type 1 will have terms in G_{p_2}, while semi-functional keys of Type 2 will have terms in G_{p_3}. When a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the extra terms from G_{p_2} in the key will be paired with the extra G_{p_2} terms in the ciphertext, which will cause decryption to fail. When a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the extra terms from G_{p_3} in the key will be paired with the extra G_{p_3} terms in the ciphertext, which will cause decryption to fail.

To more precisely describe semi-functional ciphertexts and keys, we first fix random values $z_i, t_i \in \mathbb{Z}_N$ for each attribute i which will be common to semi-functional ciphertexts and keys. These values are fixed per attribute, and do not vary for different users.

Semi-functional Ciphertexts To create a semi-functional ciphertext, we first run the encryption algorithm to obtain a normal ciphertext,

$$C'_0,\ C'_{1,x},\ C'_{2,x},\ C'_{3,x}\ \forall x.$$

We let g_2, g_3 denote generators of G_{p_2} and G_{p_3} respectively. We choose two random vectors $u_2, u_3 \in \mathbb{Z}_N^\ell$ and set $\delta_x = A_x \cdot u_2$, $\sigma_x = A_x \cdot u_3$ for each row A_x of the access matrix A. We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row $A_x \in \overline{B}$, we also choose random exponents $\gamma_x, \psi_x \in \mathbb{Z}_N$. The semi-functional ciphertext is formed as:

$$C_0 = C'_0,\ C_{1,x} = C'_{1,x},\ C_{2,x} = C'_{2,x} g_2^{\gamma_x} g_3^{\psi_x},\ C_{3,x} = C'_{3,x} g_2^{\delta_x + \gamma_x z_{\rho(x)}} g_3^{\sigma_x + \psi_x t_{\rho(x)}}\ \forall x \text{ s.t. } A_x \in \overline{B},$$

$$C_{1,x} = C'_{1,x},\ C_{2,x} = C'_{2,x},\ C_{3,x} = C'_{3,x} g_2^{\delta_x} g_3^{\sigma_x}\ \forall x \text{ s.t. } A_x \in B.$$

We say a ciphertext is *nominally* semi-functional when the values δ_x are shares of 0.

Semi-functional Keys We define the *key for identity* GID to be the collection of $H(\text{GID})$ and all keys $\text{K}_{i,\text{GID}}$ for attributes i belonging to good authorities requested by the attacker throughout the game. (These queries may occur at different times.) Semi-functional keys for an identity GID will be of two types: Type 1 or Type 2. To create a semi-functional key for identity GID, we let $H'(\text{GID})$ be a random element of G_{p_1}, and we choose a random exponent $c \in \mathbb{Z}_N$.

To create a semi-functional key of Type 1, we define the random oracle's output on GID to be:

$$H(\text{GID}) = H'(\text{GID}) g_2^c.$$

We create $\text{K}_{i,\text{GID}}$ (for an attribute i controlled by a good authority) by first creating a normal key $\text{K}'_{i,\text{GID}}$ and setting:

$$\text{K}_{i,\text{GID}} = \text{K}'_{i,\text{GID}} g_2^{cz_i}.$$

To create a semi-functional key of Type 2, we define the random oracle's output on GID to be:

$$H(\text{GID}) = H'(\text{GID}) g_3^c.$$

We create $\text{K}_{i,\text{GID}}$ (for an attribute i controlled by a good authority) by first creating a normal key $\text{K}'_{i,\text{GID}}$ and setting:

$$\text{K}_{i,\text{GID}} = \text{K}'_{i,\text{GID}} g_3^{ct_i}.$$

We note that when a semi-functional key of Type 1 is used to decrypt a semi-functional ciphertext, the additional terms $e(g_2, g_2)^{c\delta_x}$ prevent decryption from succeeding, except when the values δ_x are shares of 0 (i.e. when we have a nominally semi-functional ciphertext). When a semi-functional key of Type 2 is used to decrypt a semi-functional ciphertext, the additional terms $e(g_3, g_3)^{c\sigma_x}$ prevent successful decryption.

We now define Game_0, which is like $\text{Game}_{Real'}$, except that the ciphertext given to the attacker is semi-functional. We let q be the number of identities GID for which the attacker makes key queries $\text{K}_{i,\text{GID}}$. We define $\text{Game}_{j,1}$ and $\text{Game}_{j,2}$ for each j from 1 to q as follows:

$\text{Game}_{j,1}$ This is like Game_0, except that for the first $j-1$ queried identities, the received keys are semi-functional of Type 2, and the received key for the j^{th} queried identity is semi-functional of Type 1. The remaining keys are normal.

$\text{Game}_{j,2}$ This is like Game_0, except that for the first j queried identities, the received keys are semi-functional of Type 2. The remaining keys are normal. We note that in $\text{Game}_{q,2}$, all keys are semi-functional of Type 2.

Game_{Final} In this game, all keys are semi-functional of Type 2, and the ciphertext is a semi-functional encryption of a random message. We note that the attacker has advantage 0 in this game.

We show these games are indistinguishable in the following lemmas, the proofs of which appear in Appendix C.

Lemma 7. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $\text{Game}_{Real}Adv_{\mathcal{A}} - \text{Game}_{Real'}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Lemma 8. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{Real'}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 1.*

Lemma 9. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j-1,2}Adv_{\mathcal{A}} - Game_{j,1}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 2.*

Lemma 10. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j,1}Adv_{\mathcal{A}} - Game_{j,2}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Lemma 11. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{q,2}Adv_{\mathcal{A}} - Game_{Final}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 4.*

5 Discussion

There are multiple ways in which one might extend our work.

Removing the Random Oracle It would be desirable to remove the need for a random oracle and replace it with a concrete function H mapping identities to group elements. One approach would be to fix a degree d polynomial, $P(x)$, and map identities in $\mathbb{Z}_N$ to elements of G by setting $H(\text{GID}) := g^{P(\text{GID})}$, where g denotes a generator of the group G. This approach has previously been employed to obtain large universe constructions for Attributed-Based encryption [30]. The public parameters would then include $\{g^{P(x_i)}\}$ for $d+1$ points x_i so that $H(\text{GID})$ could be computed for any GID by polynomial interpolation. We note that $P(x)$ is a $(d+1)$-wise independent function modulo primes, but this will leave the system vulnerable to collusion attacks when $\geq d+1$ users collude. Clearly, this is far from ideal, and we would prefer a better method with stronger security guarantees.

Prime order groups An interesting direction is create a prime order group variant of our system. Using groups of prime order can potentially lead to more efficient systems (via faster group operations) and security under different assumptions. One approach is to simply use our exact construction except use a group order of one prime (instead of a product of three primes). Applying this setting results in an efficient system that we show to be generically secure in Appendix D. However, this construction does not lend itself (to the best of our knowledge) to a proof under a non-interactive assumption.

Another possible approach is to realize the subspaces needed for dual system encryption proofs using vector spaces over prime order groups instead of subgroups. We note that several systems such as BGN encryption [15], Groth-Ostrovsky-Sahai NIZK proofs [32], traitor tracing [16] , and predicate encryption [17, 36] were originally developed in the composite order setting, but later variants were developed in prime order groups [31, 47, 33, 35, 25, 26, 37]. [4] Ideally, a variant would result in security under a simple assumption such as the decision linear assumption.

[4] Freeman [25] discusses a class of general transformations, although it does not encompass our construction.

References

[1] M. Abdalla, M. Bellare, D. Catalano, E. Kiltz, T. Kohno, T. Lange, J. Malone-Lee, G. Neven, P. Paillier, and H. Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe, and extensions. In *Journal of Cryptology*, volume 21, pages 350–391, 2008.

[2] M. Abdalla, E. Kiltz, and G. Neven. Generalized key delegation for hierarchical identity-based encryption. In *Computer Security ESORICS*, pages 139–154, 2007.

[3] S. Al-Riyami, J. Malone-Lee, and N. Smart. Escrow-free encryption supporting cryptographic workflow. In *Int. J. Inf. Sec.*, volume 5, pages 217–229, 2006.

[4] W. Bagga, R. Molva, and S. Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS*, page 368, 2006.

[5] M. Barbosa and P. Farshim. Secure cryptographic workflow in the standard model. In *INDOCRYPT*, pages 379–393, 2006.

[6] A. Beimel. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

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

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

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

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

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

[12] D. Boneh, G. Di Crescenzo, R. Ostrovsky, and G. Persiano. Public key encryption with keyword search. In *EUROCRYPT*, pages 506–522, 2004.

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

[14] D. Boneh, C. Gentry, and M. Hamburg. Space-efficient identity based encryption without pairings. In *Proceedings of FOCS*, pages 647–657, 2007.

[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] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. In *TCC*, pages 535–554, 2007.

[18] R. Bradshaw, J. Holt, and K. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[19] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT*, 2001.

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

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

[22] 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.

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

[24] 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.

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

[26] 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.

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

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

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

[30] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute Based Encryption for Fine-Grained Access Conrol of Encrypted Data. In *ACM conference on Computer and Communications Security*, pages 89–98, 2006.

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

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

[33] J. Groth and A. Sahai. Efficient non-interactive proof systems for bilinear groups. In *EUROCRYPT*, pages 415–432, 2008.

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

[35] V. Iovino and G. Persiano. Hidden-vector encryption with groups of prime order. In *Pairing*, pages 75–88, 2008.

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

[37] 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.

[38] A. Lewko and B. Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In *TCC*, pages 455–579, 2010.

[39] H. Lin, Z. Cao, X. Liang, and J. Shao. Secure threshold multi authority attribute based encryption without a central authority. In *INDOCRYPT*, pages 426–436, 2008.

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

[41] S. Müller, S. Katzenbeisser, and C. Eckert. Distributed attribute-based encryption. In *ICISC*, pages 20–36, 2008.

[42] S. Müller, S. Katzenbeisser, and C. Eckert. On multi-authority ciphertext-policy attribute-based encryption. In *Bulletin of the Korean Mathematical Society 46, 4*, pages 803–819, 2009.

[43] 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.

[44] A. Sahai and B. Waters. Fuzzy Identity Based Encryption. In *EUROCRYPT*, pages 457–473, 2005.

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

[46] E. Shi, J. Bethencourt, H. Chan, D. Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, 2007.

[47] E. Shi, J. Bethencourt, H. T.-H. Chan, D. Xiaodong Song, and A. Perrig. Multi-dimensional range query over encrypted data. In *IEEE Symposium on Security and Privacy*, pages 350–364, 2007.

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

[49] V. Shoup. Lower bounds for discrete logarithms and related problems. In *EUROCRYPT*, page 256266, 1997.

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

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

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

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

A Background

We now define access structures, linear secret-sharing schemes, and composite order bilinear groups.

A.1 Access Structures

Definition 12. *(Access Structure [6]) Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$. An* access structure *(respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \{\}$. 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 only consider monotone access structures. We observe that more general access structures can be (inefficiently) realized with our techniques by letting the negation of an attribute be a separate attribute (this doubles the total number of attributes).

A.2 Linear Secret-Sharing Schemes

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

Definition 13. *(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 $x = 1, \ldots, \ell$, the x^{th} row of A is labeled by a party $\rho(x)$ (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 $(Av)_x$ belongs to party $\rho(x)$.*

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 $I = \{x | \rho(x) \in S\}$. Then there exist constants $\{\omega_x \in \mathbb{Z}_p\}_{x \in I}$ such that, for any valid shares $\{\lambda\}_x$ of a secret s according to Π, we have: $\sum_{x \in I} \omega_x \lambda_x = s$. These constants $\{\omega_x\}$ can be found in polynomial time with respect to the size of the share-generating matrix A [6].

Boolean Formulas Access policies can also be described in terms of monotonic boolean formulas. LSSS access structures are more general, and can be derived from representations as boolean formulas. There are standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix. We can represent the boolean formula as an access tree, where the interior nodes are AND and OR gates, and the leaf nodes correspond to attributes. The number of rows in the corresponding LSSS matrix will be same as the number of leaf nodes in the access tree. The conversion process is described in Appendix G.

A.3 Composite Order Bilinear Groups

We construct our system using composite order bilinear groups, which were first introduced in [15]. We define a group generator $\mathcal{G}$, an algorithm which takes a security parameter λ as input and outputs a description of a bilinear group G. For our purposes, we will have $\mathcal{G}$ output $(p_1, p_2, p_3, G, G_T, e)$ where p_1, p_2, p_3 are distinct primes, G and G_T are cyclic groups of order $N = p_1 p_2 p_3$, and $e : G^2 \to G_T$ is a map such that:

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

We assume that the group operations in G and G_T as well as the bilinear map e are computable in polynomial time with respect to λ and that the group descriptions of G and G_T include generators of the respective cyclic groups. We let G_{p_1}, G_{p_2}, and G_{p_3} denote the subgroups of order p_1, p_2, and p_3 in G respectively. We note that when $h_i \in G_{p_i}$ and $h_j \in G_{p_j}$ for $i \neq j$, $e(h_i, h_j)$ is the identity element in G_T. To show this, suppose $h_1 \in G_{p_1}$ and $h_2 \in G_{p_2}$. Let g denote a generator of G. Then, $g^{p_1 p_2}$ generates G_{p_3}, $g^{p_1 p_3}$ generates G_{p_2}, and $g^{p_2 p_3}$ generates G_{p_1}. Hence, for some α_1, α_2, $h_1 = (g^{p_2 p_3})^{\alpha_1}$ and $h_2 = (g^{p_1 p_3})^{\alpha_2}$. Then:

$$e(h_1, h_2) = e(g^{p_2 p_3 \alpha_1}, g^{p_1 p_3 \alpha_2}) = e(g^{\alpha_1}, g^{p_3 \alpha_2})^{p_1 p_2 p_3} = 1.$$

This orthogonality property of $G_{p_1}, G_{p_2}, G_{p_3}$ will be used to implement semi-functionality in our constructions.

B Transformation from One-Use Multi-Authority CP-ABE

We suppose that we have a multi-authority CP-ABE system on a universe U of attributes using LSSS access matrices that is secure when we restrict the row labeling ρ of each matrix to be injective (i.e. each attribute is used at most once). To construct a system that is secure when we use an attribute $\leq k$ times in the row labeling of an access matrix, we make k copies of each attribute in the system. For each attribute B, we replace it with k new "attributes" $B : 1, \ldots, B : k$ (all controlled by the same authority that controlled B). When we want to label a row of an access matrix with the attribute B, we label it with $B : i$ for a fresh value of i, so that each new "attribute" $B : i$ is only used once.

To make a key for an identity, attribute pair (GID, B), the authority controlling attribute B makes k keys corresponding to the pairs $(\text{GID}, B : 1), \ldots, (\text{GID}, B : k)$ and gives this collection to the key requester. We let S denote a subset of attributes. If we define $S' := \{B : 1, \ldots, B : k | B \in S\}$, then it follows that S' satisfies an access matrix under our new labeling if and only if S satisfies the access matrix under the original labeling (which used each B at most k times). Security for our system that uses attributes at most k times now follows from the security of the one-use system.

We will obtain a fully secure multi-authority CP-ABE system where attributes are used multiple times by constructing a system which is secure where attributes are used only once in an access matrix and then applying this transformation.

We note that the size of the universe of attributes will increase by a factor of k, as will the size of keys. For our construction, the sizes of the public parameters for each authority depend linearly on the number of attributes controlled by that authority, so these will expand by a factor of k under this transformation. Private keys for an identity, attribute pair will also expand by a factor of k. We note that the size of the access matrix does not change, so our ciphertexts remain the same size.

C Proof of Security

Lemma 7. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{Real}Adv_{\mathcal{A}} - Game_{Real'}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ receives N, g_1, T. $\mathcal{B}$ will simulate either Game_{Real} or $\text{Game}_{Real'}$ with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters $e(g_1, g_1)^{\alpha_i}, g_1^{y_i}$.

When $\mathcal{A}$ first queries the random oracle for $H(\text{GID})$, $\mathcal{B}$ chooses a random exponent $h_{\text{GID}} \in \mathbb{Z}_N$ and sets $H(\text{GID}) = T^{h_{\text{GID}}}$. It stores this value so that it can respond consistently if $H(\text{GID})$ is queried again. If T is a generator of G_{p_1}, these will be random elements of G_{p_1}. If T is a generator of G, these will be random elements of G.

When $\mathcal{A}$ makes a key query (GID, i), $\mathcal{B}$ can generate the corresponding key by using the key generation algorithm, since it knows α_i and y_i. (It sets $H(\text{GID})$ as above if this has not been separately queried yet.) At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages M_0 and M_1 and an access matrix (A, ρ). $\mathcal{A}$ additionally supplies $\mathcal{B}$ with the public parameters for any corrupted attributes appearing in the matrix. $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$. To create the challenge ciphertext for M_β, $\mathcal{B}$ uses the encryption algorithm.

If T is a generator of G, then $\mathcal{B}$ has properly simulated Game_{Real}. If T is a generator of G_{p_1}, then $\mathcal{B}$ has properly simulated $\text{Game}_{Real'}$. Thus, $\mathcal{B}$ can use $\mathcal{A}$ to attain advantage ϵ in breaking Assumption 1. □

Lemma 8. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{Real'}Adv_{\mathcal{A}} - Game_0Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 1.*

Proof. $\mathcal{B}$ receives N, g_1, T. $\mathcal{B}$ will simulate either $\text{Game}_{Real'}$ or Game_0 with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters $e(g_1, g_1)^{\alpha_i}, g_1^{y_i}$.

When $\mathcal{A}$ first queries the random oracle for $H(\text{GID})$, $\mathcal{B}$ chooses a random exponent $h_{\text{GID}} \in \mathbb{Z}_N$ and sets $H(\text{GID}) = g_1^{h_{\text{GID}}}$. It stores this value so that it can respond consistently if $H(\text{GID})$ is queried again. When $\mathcal{A}$ makes a key query (GID, i), $\mathcal{B}$ generates the corresponding key using the key generation algorithm, since it knows α_i and y_i. (If necessary, it sets $H(\text{GID})$ as above.)

At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0, M_1, and an access matrix (A, ρ). $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β as follows. First, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_N$ and sets $C_0 = Me(g_1, g_1)^s$. $\mathcal{B}$ also chooses two vectors, $v = (s, v_2, \ldots, v_\ell), w = (0, w_2, \ldots, w_\ell)$, where $v_2, \ldots, v_\ell, w_2, \ldots, w_\ell$ are chosen randomly from $\mathbb{Z}_N$. We let $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot w$.

$\mathcal{A}$ additionally supplies $\mathcal{B}$ with public parameters $g^{y_i}, e(g_1, g_1)^{\alpha_i}$ for attributes i belonging to corrupt authorities which are included in the access matrix (A, ρ). We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row A_x in B, $\mathcal{B}$ chooses a random value $r_x \in \mathbb{Z}_N$. For each row $A_x \in \overline{B}$, $\mathcal{B}$ chooses a random value $r'_x \in \mathbb{Z}_N$, and will implicitly set $r_x = rr'_x$, where g_1^r is the G_{p_1} part of T. We will also embed T into the terms with shares ω_x.

For each row $A_x \in B$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} \left(e(g_1, g_1)^{\alpha_{\rho(x)}}\right)^{r_x},$$

$$C_{2,x} = g_1^{r_x},\ C_{3,x} = \left(g_1^{y_{\rho(x)}}\right)^{r_x} T^{\omega_x}.$$

For each row $A_x \in \overline{B}$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, T)^{\alpha_{\rho(x)} r'_x},$$

$$C_{2,x} = T^{r'_x},\ C_{3,x} = T^{y_{\rho(x)} r'_x} T^{\omega_x}.$$

We note that the G_{p_1} part of T^{ω_x} is $g_1^{A_x \cdot rw}$, and rw is a random vector with first coordinate equal to 0. Thus, if $T \in G_{p_1}$, this is a properly distributed normal ciphertext. If $T \in G$, then this is a semi-functional ciphertext with parameters $\delta_x = A_x \cdot cw$ modulo p_2 where g_2^c is the G_{p_2} part of T, $\sigma_x = A_x \cdot dw$ modulo p_3 where g_3^d is the G_{p_3} part of T, $g_2^{\gamma_x}$ equals the G_{p_2} part of $T^{r'_x}$, $g_3^{\psi_x}$ equals the G_{p_3} part of $T^{r'_x}$, $z_{\rho(x)} = y_{\rho(x)}$ modulo p_2, and $t_{\rho(x)} = y_{\rho(x)}$ modulo p_3.

To see that this is properly distributed, we note that since $r'_x, y_{\rho(x)}$ are chosen randomly in $\mathbb{Z}_N$, their values modulo p_1, modulo p_2, and modulo p_3 are uncorrelated by the Chinese Remainder Theorem. This means that our $\gamma_x, \psi_x, z_{\rho(x)}, t_{\rho(x)}$ parameters are randomly distributed. The entries $w_2, \ldots, w_\ell$ of w are also randomly distributed modulo p_2, p_3, however both δ_x and σ_x are shares of 0 from the simulator's perspective. We must argue that these appear to be shares of a random exponent in $\mathcal{A}$'s view.

We note that the shares δ_x, σ_x for rows $A_x \in B$ are information-theoretically revealed to $\mathcal{A}$, but the space R spanned by these rows cannot include the vector $(1, 0, \ldots, 0)$. This means there is some vector u such that u is orthogonal to R modulo p_2, but u is not orthogonal to $(1, 0, \ldots, 0)$. We fix a basis including the vector u, and write $cw = w' + au$ for some a modulo p_2 and w' in the span of the other basis elements. We note that w' is uniformly distributed in this space (modulo p_2) and reveals no information about a (modulo p_2). Now, the first coordinate of cw modulo p_2 depends on the value of a, and the shares δ_x for $A_x \in B$ contain no information about a (since u is orthogonal to R). The only information $\mathcal{A}$ receives about the value of a appears in exponents of the form $\delta_x + \gamma_x z_{\rho(x)}$, where the $z_{\rho(x)}$ is a new random value each time that appears nowhere else (recall that ρ is constrained to be injective). As long as γ_x does not equal 0 modulo p_2 ($\gamma_x = 0$ with only negligible probability), this means that any value of δ_x can be explained by $z_{\rho(x)}$ taking on a particular value. Since $z_{\rho(x)}$ is uniformly random, this means that no information about the value of a is revealed. Hence, the value being shared is information-theoretically hidden, and the shares δ_x (and similarly σ_x) are properly distributed in the adversary's view.

Thus, when $T \in G_{p_1}$, $\mathcal{B}$ properly simulates $\text{Game}_{Real'}$. When $T \in G$, $\mathcal{B}$ properly simulates Game_0 with probability negligibly close to 1. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to obtain advantage negligibly close to ϵ in breaking Assumption 1. □

Lemma 9. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j-1,2}Adv_{\mathcal{A}} - Game_{j,1}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage negligibly close to ϵ in breaking Assumption 2.*

Proof. $\mathcal{B}$ receives g_1, g_3, X_1X_2, T. $\mathcal{B}$ will simulate either $\text{Game}_{j-1,2}$ or $\text{Game}_{j,1}$ with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters $e(g_1, g_1)^{\alpha_i}, g_1^{y_i}$.

We let GID_k denote the k^{th} identity queried by $\mathcal{A}$. When $\mathcal{A}$ first queries the random oracle for $H(\text{GID}_k)$, if $k > j$, then $\mathcal{B}$ chooses a random exponent $h_{\text{GID}_k} \in \mathbb{Z}_N$ and sets $H(\text{GID}_k) = g_1^{h_{\text{GID}_k}}$. If $k < j$, then $\mathcal{B}$ chooses a random exponent $h_{\text{GID}_k} \in \mathbb{Z}_N$ and sets $H(\text{GID}_k) = (g_1 g_3)^{h_{\text{GID}_k}}$ (we note that this is a random element of $G_{p_1 p_3}$ since the values of h_{GID_k} modulo p_1 and

modulo p_3 are uncorrelated). When $k = j$, $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_j} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_j) = T^{h_{\mathrm{GID}_j}}$. In all cases, it stores this value so that it can respond consistently if $H(\mathrm{GID}_k)$ is queried again.

When $\mathcal{A}$ makes a key query (i, GID_k), $\mathcal{B}$ responds as follows. If $H(\mathrm{GID}_k)$ has already been fixed, then $\mathcal{B}$ retrieves the stored value. Otherwise, $\mathcal{B}$ creates $H(\mathrm{GID}_k)$ according to k as above. $\mathcal{B}$ forms the key as:

$$\mathrm{K}_{i,\mathrm{GID}_k} = g_1^{\alpha_i} H(\mathrm{GID}_k)^{y_i}.$$

Notice that for $k < j$, $\mathcal{B}$ forms properly distributed semi-functional keys of Type 2, where t_i is congruent to y_i modulo p_3 (these are uncorrelated from the values of y_i modulo p_1 which appear in the public parameters). Also recall that the values t_i are fixed per attribute, and do not vary across different keys. For $k > j$, $\mathcal{B}$ forms properly distributed normal keys. For $k = j$, $\mathcal{B}$ forms a normal key if $T \in G_{p_1}$ and a semi-functional key of Type 1 if $T \in G_{p_1 p_2}$.

At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0, M_1, and an access matrix (A, ρ). $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β as follows. (We note that $\mathcal{B}$ will produce a nominally semi-functional ciphertext, but this will be hidden from $\mathcal{A}$'s view.) First, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_N$ and sets $C_0 = Me(g_1, g_1)^s$. $\mathcal{B}$ also chooses three vectors, $v = (s, v_2, \ldots, v_\ell), w = (0, w_2, \ldots, w_\ell), u = (u_1, \ldots, u_\ell)$, where $v_2, \ldots, v_\ell, w_2, \ldots, w_\ell, u_1, \ldots, u_\ell$ are chosen randomly from $\mathbb{Z}_N$. We let $\lambda_x = A_x \cdot v$, $\omega_x = A_x \cdot w$, and $\sigma_x = A_x \cdot u$.

$\mathcal{A}$ additionally supplies $\mathcal{B}$ with public parameters $g^{y_i}, e(g_1, g_1)^{\alpha_i}$ for attributes i belonging to corrupt authorities which are included in the access matrix (A, ρ). We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row A_x in B, $\mathcal{B}$ chooses a random value $r_x \in \mathbb{Z}_N$. For each row $A_x \in \overline{B}$, $\mathcal{B}$ chooses random values $\psi_x, r'_x \in \mathbb{Z}_N$, and will implicitly set $r_x = r r'_x$, where g_1^r is X_1.

For each row $A_x \in B$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} \left(e(g_1, g_1)^{\alpha_{\rho(x)}}\right)^{r_x},$$

$$C_{2,x} = g_1^{r_x},\ C_{3,x} = \left(g_1^{y_{\rho(x)}}\right)^{r_x} (X_1 X_2)^{\omega_x} g_3^{\sigma_x}.$$

For each row $A_x \in \overline{B}$, the ciphertext is formed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, X_1 X_2)^{\alpha_{\rho(x)} r'_x},$$

$$C_{2,x} = (X_1 X_2)^{r'_x} g_3^{\psi_x},\ C_{3,x} = (X_1 X_2)^{y_{\rho(x)} r'_x} g_3^{y_{\rho(x)} \psi_x} (X_1 X_2)^{\omega_x} g_3^{\sigma_x}.$$

We note that the $X_1^{\omega_x}$ is $g_1^{A_x \cdot rw}$, and rw is a random vector with first coordinate equal to 0. This is a semi-functional ciphertext with parameters $\delta_x = A_x \cdot cw$ modulo p_2 where g_2^c is X_2, $g_2^{\gamma_x}$ equals $X_2^{r'_x}$, $z_{\rho(x)} = y_{\rho(x)}$ modulo p_2, and $t_{\rho(x)} = y_{\rho(x)}$ modulo p_3.

To see that this is properly distributed, we note that since $r'_x, y_{\rho(x)}$ are chosen randomly in $\mathbb{Z}_N$, their values modulo p_1 and modulo p_2 are uncorrelated. This means that our $\gamma_x, \psi_x, z_{\rho(x)}, t_{\rho(x)}$ parameters are randomly distributed. It is clear that σ_x is properly distributed, since it is a share of a random vector. The entries $w_2, \ldots, w_\ell$ of w are also randomly distributed modulo p_2, however the δ_x's are shares of 0 from the simulator's perspective. We must argue that these appear to be shares of a random exponent in $\mathcal{A}$'s view.

We let the space R denote the span of the rows of A whose attributes are in B and the rows whose attributes $\rho(x)$ are queried by the attacker with identity GID_j. This space cannot include the vector $(1, 0, \ldots, 0)$, so there is some vector u' which is orthogonal to R modulo p_2 and not orthogonal to $(1, 0, \ldots, 0)$. We can then write $cw = w' + au'$ for some a modulo p_2 and w' in the span of the other basis vectors. We note that w' is uniformly distributed in this

space, and reveals no information about a. The value of the first coordinate of cw modulo p_2 depends on the value of a, but the shares δ_x for $A_x \in B$ contain no information about a. The only information $\mathcal{A}$ receives about the value of a appears in exponents of the form $\delta_x + \gamma_x z_{\rho(x)}$, where the $z_{\rho(x)}$ is a new random value each time that appears nowhere else (recall that ρ is constrained to be injective). (We note that these $z_{\rho(x)}$ values modulo p_2 do not occur in any keys for identities not equal to GID_j, since these keys are either normal or semi-functional of type 2, and hence do not have components in G_{p_2}.) As long as γ_x does not equal 0 ($\gamma_x = 0$ with only negligible probability), this means that any value of δ_x can be explained by $z_{\rho(x)}$ taking on a particular value. Since $z_{\rho(x)}$ is uniformly random, this means that no information about the value of a modulo p_2 is revealed. Hence, the value being shared is information-theoretically hidden, and the δ_x's are properly distributed in the adversary's view.

Though it is hidden from $\mathcal{A}$, the fact that we can only make δ_x shares of 0 is crucial here (i.e. the simulator can only make a nominally semi-functional ciphertext). If $\mathcal{B}$ tried to test the semi-functionality of the j^{th} key for itself by making a challenge ciphertext the key could decrypt, decryption would succeed regardless of the presence of G_{p_2} components, since the δ_x's are shares of 0. Hence the simulator would not be able to tell whether the j^{th} key was semi-functional of Type 1 or normal.

In summary, when $T \in G_{p_1}$, $\mathcal{B}$ properly simulates $\mathrm{Game}_{j-1,2}$. When $T \in G_{p_1 p_2}$, $\mathcal{B}$ properly simulates $\mathrm{Game}_{j,1}$ with probability negligibly close to 1. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to obtain advantage negligibly close to ϵ in breaking Assumption 2.

□

Lemma 10. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $Game_{j,1}Adv_{\mathcal{A}} - Game_{j,2}Adv_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 3.*

Proof. $\mathcal{B}$ receives $N, g_1, X_1X_3, Y_2Y_3, T$. $\mathcal{B}$ will simulate either $\mathrm{Game}_{j,1}$ or $\mathrm{Game}_{j,2}$ with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i, y_i \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters $e(g_1, g_1)^{\alpha_i}, g_1^{y_i}$.

We let GID_k denote the k^{th} identity queried by $\mathcal{A}$. When $\mathcal{A}$ first queries the random oracle for $H(\mathrm{GID}_k)$, if $k > j$, then $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_k} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_k) = g_1^{h_{\mathrm{GID}_k}}$. If $k < j$, then $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_k} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_k) = (X_1X_3)^{h_{\mathrm{GID}_k}}$ (we note that this is a random element of $G_{p_1p_3}$ since the values of h_{GID_k} modulo p_1 and modulo p_3 are uncorrelated). When $k = j$, $\mathcal{B}$ chooses a random exponent $h_{\mathrm{GID}_j} \in \mathbb{Z}_N$ and sets $H(\mathrm{GID}_j) = T^{h_{\mathrm{GID}_j}}$. In all cases, it stores this value so that it can respond consistently if $H(\mathrm{GID}_k)$ is queried again.

When $\mathcal{A}$ makes a key query (i, GID_k), $\mathcal{B}$ responds as follows. If $H(\mathrm{GID}_k)$ has already been fixed, then $\mathcal{B}$ retrieves the stored value. Otherwise, $\mathcal{B}$ creates $H(\mathrm{GID}_k)$ according to k as above. $\mathcal{B}$ forms the key as:

$$\mathrm{K}_{i,\mathrm{GID}_k} = g_1^{\alpha_i} H(\mathrm{GID}_k)^{y_i}.$$

Notice that for $k < j$, $\mathcal{B}$ forms properly distributed semi-functional keys of Type 2, where t_i is congruent to y_i modulo p_3 (these are uncorrelated from the values of y_i modulo p_1 which appear in the public parameters). For $k > j$, $\mathcal{B}$ forms properly distributed normal keys. For $k = j$, $\mathcal{B}$ forms a properly distributed semi-functional key of Type 1 if $T \in G_{p_1p_2}$ and a semi-functional key of Type 2 if $T \in G_{p_1p_3}$.

At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0, M_1, and an access matrix (A, ρ). $\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β as follows. First, $\mathcal{B}$ chooses a random $s \in \mathbb{Z}_N$ and

sets $C_0 = Me(g_1, g_1)^s$. $\mathcal{B}$ also chooses three vectors, $v = (s, v_2, \ldots, v_\ell), w = (0, w_2, \ldots, w_\ell), u = (u_1, \ldots, u_\ell)$, where $v_2, \ldots, v_\ell, w_2, \ldots, w_\ell, u_1, \ldots, u_\ell$ are chosen randomly from $\mathbb{Z}_N$. We let $\lambda_x = A_x \cdot v$, $\omega_x = A_x \cdot w$, and $\delta_x = A_x \cdot u$.

$\mathcal{A}$ additionally supplies $\mathcal{B}$ with public parameters $g^{y_i}, e(g_1, g_1)^{\alpha_i}$ for attributes i belonging to corrupt authorities which are included in the access matrix (A, ρ). We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities. For each row A_x, $\mathcal{B}$ chooses a random value $r_x \in \mathbb{Z}_N$.

For each row $A_x \in B$, the ciphertext is computed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} \left(e(g_1, g_1)^{\alpha_{\rho(x)}}\right)^{r_x},$$

$$C_{2,x} = g_1^{r_x},\ C_{3,x} = \left(g_1^{y_{\rho(x)}}\right)^{r_x} g_1^{\omega_x} (Y_2 Y_3)^{\delta_x}.$$

For each row $A_x \in \overline{B}$, the ciphertext is computed as:

$$C_{1,x} = e(g_1, g_1)^{\lambda_x} e(g_1, g_1)^{\alpha_{\rho(x)} r_x},$$

$$C_{2,x} = g_1^{r_x} (Y_2 Y_3)^{r_x},\ C_{3,x} = g_1^{y_{\rho(x)} r_x} g_1^{\omega_x} (Y_2 Y_3)^{y_{\rho(x)} r_x} (Y_2 Y_3)^{\delta_x}.$$

We note that this implicitly sets $z_{\rho(x)} \equiv y_{\rho(x)}$ modulo p_2 and $t_{\rho(x)} \equiv y_{\rho(x)}$ modulo p_3. We note that these values are uncorrelated. We note that the sharing vector u is random modulo p_2 and p_3, so this is a properly distributed semi-functional ciphertext.

Thus, when $T \in G_{p_1 p_2}$, $\mathcal{B}$ properly simulates $\text{Game}_{j,1}$. When $T \in G_{p_1 p_3}$, $\mathcal{B}$ properly simulates $\text{Game}_{j,2}$. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to obtain advantage ϵ in breaking Assumption 3.

□

Lemma 11. *Suppose there exists a polynomial time algorithm $\mathcal{A}$ such that $\text{Game}_{q,2}\text{Adv}_{\mathcal{A}} - \text{Game}_{Final}\text{Adv}_{\mathcal{A}} = \epsilon$. Then we can construct a polynomial time algorithm $\mathcal{B}$ with advantage ϵ in breaking Assumption 4.*

Proof. $\mathcal{B}$ first receives $g_1, g_2, g_3, g_1^a, g_1^b g_3^b, g_1^c, g_1^{ac} g_3^d, T$. $\mathcal{B}$ will simulate either $\text{Game}_{q,2}$ or Game_{Final} with $\mathcal{A}$, depending on the value of T. $\mathcal{B}$ outputs g_1 as the public generator of G_{p_1} and N as the group order. $\mathcal{A}$ specifies a set $S' \subseteq S$ of corrupt authorities, where S is the the set of all authorities in the system. For each attribute i belonging to a good authority, $\mathcal{B}$ chooses random exponents $\alpha_i', y_i' \in \mathbb{Z}_N$ and gives $\mathcal{A}$ the public parameters

$$e(g_1, g_1)^{\alpha_i} = e(g_1^a, g_1^b g_3^b) e(g_1, g_1)^{\alpha_i'},\ g_1^{y_i} = g_1^a g_1^{y_i'}.$$

We note that this sets $\alpha_i = ab + \alpha_i'$ and $y_i = a + y_i'$.

When $\mathcal{A}$ queries the random oracle for $H(\text{GID})$, $\mathcal{B}$ chooses random exponents $f, h \in \mathbb{Z}_N$ and sets

$$H(\text{GID}) = (g_1^b g_3^b)^{-1} g_1^f g_3^h.$$

It stores this value.

When $\mathcal{A}$ makes a key query (i, GID), $\mathcal{B}$ responds as follows. If $H(\text{GID})$ has already been fixed, then $\mathcal{B}$ retrieves the stored value. Otherwise, $\mathcal{B}$ creates $H(\text{GID})$ as above. If we consider only the subgroup 1 parts, $\mathcal{B}$ needs to compute:

$$\text{K}_{i,\text{GID}} = g_1^{\alpha_i} (g_1^{-b+f})^{y_i} = g_1^{ab+\alpha_i'} g_1^{-ba+fa-by_i'+fy_i'} = g_1^{\alpha_i'+fy_i'} (g_1^a)^f g_1^{-by_i'}.$$

Notice that the g_1^{ab} and g_1^{-ab} terms cancel. Only the last term, $g_1^{-by_i'}$ is unknown to $\mathcal{B}$, the rest can be easily computed. To form the term $g_1^{-by_i'}$, $\mathcal{B}$ will raise $g_1^b g_3^b$ to the power $-y_i'$. The full key can be formed as:

$$g_1^{\alpha_i' + fy_i'} (g_1^a)^f (g_1^b g_3^b)^{-y_i'} g_3^{hy_i'}.$$

We note that this sets t_i equal to y_i' modulo p_3, and that this value is not correlated with the value of y_i' modulo p_1.

At some point, $\mathcal{A}$ gives $\mathcal{B}$ two messages, M_0, M_1, and an access matrix (A, ρ). $\mathcal{A}$ additionally supplies $\mathcal{B}$ with public parameters $g^{y_i}, e(g_1, g_1)^{\alpha_i}$ for attributes i belonging to corrupt authorities which are included in the access matrix (A, ρ). We let B denote the subset of rows of A whose corresponding attributes belong to corrupted authorities. We let $\overline{B}$ be the subset of rows of A whose corresponding attributes belong to good authorities.

$\mathcal{B}$ flips a random coin $\beta \in \{0, 1\}$, and encrypts M_β as follows. $\mathcal{B}$ sets:

$$C_0 = M_\beta T.$$

We think of this as setting $s = abc$. If $T = e(g_1, g_1)^{abc}$, then this will be an encryption of M_β. If T is random, this is will an encryption of a random message.

$\mathcal{B}$ chooses a random vector u_1 with entries in $\mathbb{Z}_N$, subject to the constraints that the first entry is 1 and u_1 is orthogonal to all the rows in B (such a vector must exist, since otherwise the access matrix is illegal - see [30]). We additionally choose a random vector u_2 with entries in $\mathbb{Z}_N$ such that the first entry is 0 and the rest are randomly chosen. We define the vector $v = abcu_1 + u_2$ (we note that this vector is uniformly random from $\mathcal{A}$'s perspective). We let $\lambda_x := A_x \cdot v = abcA_x \cdot u_1 + A_x \cdot u_2$.

Since $\mathcal{B}$ cannot form the terms $e(g_1, g_1)^{abcA_x \cdot u_1}$ for rows $A_x \in \overline{B}$, it sets $r_x = -cA_x \cdot u_1 + r_x'$, where r_x' is randomly chosen from $\mathbb{Z}_N$. Then we have:

$$\begin{aligned} \lambda_x + \alpha_{\rho(x)} r_x &= abcA_x \cdot u_1 + A_x \cdot u_2 + (ab + \alpha_{\rho(x)}')(-cA_x \cdot u_1 + r_x') \\ &= A_x \cdot u_2 - c\alpha_{\rho(x)}' A_x \cdot u_1 + abr_x' + \alpha_{\rho(x)}' r_x'. \end{aligned}$$

This allows $\mathcal{B}$ to form $C_{1,x}$ for $A_x \in \overline{B}$ as:

$$C_{1,x} = e(g_1, g_1^c)^{-\alpha_{\rho(x)}' A_x \cdot u_1} e(g_1^a, g_1^b g_3^b)^{r_x'} e(g_1, g_1)^{A_x \cdot u_2 + \alpha_{\rho(x)}' r_x'}.$$

For rows $A_x \in B$ corresponding to corrupt authorities, $\mathcal{B}$ chooses $r_x \in \mathbb{Z}_N$ randomly and sets:

$$C_{1,x} = e(g_1, g_1)^{A_x \cdot u_2} \left(e(g_1, g_1)^{\alpha_{\rho(x)}}\right)^{r_x}.$$

We note that $\lambda_x = A_x \cdot u_2$ for these rows because u_1 is orthogonal to A_x.

For rows $A_x \in \overline{B}$, $\mathcal{B}$ can form $C_{2,x}$ by choosing a random value $\gamma_x \in \mathbb{Z}_N$ and setting:

$$C_{2,x} = (g_1^c)^{-A_x \cdot u_1 + r_x'} (g_2 g_3)^{\gamma_x}.$$

We note that the values of γ_x modulo p_2 and p_3 are uncorrelated, so this is properly distributed. For rows $A_x \in B$, $\mathcal{B}$ can simply compute $C_{2,x} = g_1^{r_x}$.

$\mathcal{B}$ now chooses a random vector w with first entry equal to 0 and other entries randomly chosen from $\mathbb{Z}_N$, and a random vector u_3 whose entries are all randomly chosen from $\mathbb{Z}_N$. We let $\omega_x = A_x \cdot w$ and $\delta_x = A_x \cdot u_3$. For rows $A_x \in \overline{B}$, we note that

$$y_{\rho(x)} r_x = (a + y_{\rho(x)}')(-cA_x \cdot u_1 + r_x') = -acA_x \cdot u_1 - cy_{\rho(x)}' A_x \cdot u_1 + r_x' a + y_{\rho(x)}' r_x',$$

so $\mathcal{B}$ can form $C_{3,x}$ as:

$$C_{3,x} = g_1^{\omega_x} (g_1^c)^{-y'_{\rho(x)} A_x \cdot u_1} (g_1^a)^{r'_x} g_1^{y'_{\rho(x)} r'_x} (g_1^{ac} g_3^d)^{-A_x \cdot u_1} (g_2 g_3)^{\delta_x + \gamma_x y'_{\rho(x)}}.$$

(This is consistent with $t_{\rho(x)}$ being congruent to $y'_{\rho(x)}$ modulo p_3 in the keys.) We note that the sharing vector in subgroups G_{p_2} and G_{p_3} is $-du_1 + u_3$, which is random modulo p_2 and modulo p_3.

For rows $A_x \in B$, $\mathcal{B}$ sets:

$$C_{3,x} = (g_1^{y_{\rho(x)}})^{r_x} g_1^{\omega_x} (g_2 g_3)^{A_x \cdot u_3}.$$

The sharing vector is consistent here because u_1 is orthogonal to all of these rows A_x. This is a properly distributed semi-functional ciphertext with $s = abc$. If $T = e(g_1, g_1)^{abc}$, this is a semi-functional encryption of M_β, and $\mathcal{B}$ has simulated $\text{Game}_{q,2}$. If T is random, then this is a semi-functional encryption of a random message, so $\mathcal{B}$ has simulated Game_{Final}. Hence, $\mathcal{B}$ can use $\mathcal{A}$ to obtain advantage ϵ in breaking Assumption 4.

□

D Construction in Prime Order Groups

Our multi-authority CP-ABE system in prime order groups is almost identical to our construction in composite order groups, with the prime order group G now playing the role of the subgroup G_{p_1} in the composite order group construction.

Global Setup$(\lambda) \rightarrow \text{GP}$ In the global setup, a bilinear group G of prime order p is chosen. The global public parameters, GP, are p and a generator g of G. A random oracle H maps global identities GID to elements of G.

Authority Setup$(\text{GP}) \rightarrow \text{PK}, \text{SK}$ For each attribute i belonging to the authority, the authority chooses two random exponents $\alpha_i, y_i \in \mathbb{Z}_p$ and publishes $\text{PK}_j = \{e(g,g)^{\alpha_i}, g^{y_i} \forall i\}$ as its public key. It keeps $\text{SK} = \{\alpha_i, y_i \forall i\}$ as its secret key.

Encrypt$(M, (A, \rho), \text{GP}, \{\text{PK}\}) \rightarrow \text{CT}$ The encryption algorithm takes in a message M, an $n \times \ell$ access matrix A with ρ mapping its rows to attributes, the global parameters, and the public keys of the relevant authorities. It chooses a random $s \in \mathbb{Z}_p$ and a random vector $v \in \mathbb{Z}_p^\ell$ with s as its first entry. We let λ_x denote $A_x \cdot v$, where A_x is row x of A. It also chooses a random vector $w \in \mathbb{Z}_p^\ell$ with 0 as its first entry. We let ω_x denote $A_x \cdot w$. For each row A_x of A, it chooses a random $r_x \in \mathbb{Z}_p$. The ciphertext is computed as:

$$C_0 = M e(g,g)^s,\ C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} r_x},\ C_{2,x} = g^{r_x},\ C_{3,x} = g^{y_{\rho(x)} r_x} g^{\omega_x}\ \forall x.$$

KeyGen$(\text{GID}, i, \text{SK}, \text{GP}) \rightarrow \text{K}_{i,\text{GID}}$ To create a key for GID for attribute i belonging to an authority, the authority computes:

$$\text{K}_{i,\text{GID}} = g^{\alpha_i} H(\text{GID})^{y_i}.$$

Decrypt$(\mathrm{CT}, \{\mathrm{K}_{i,\mathrm{GID}}\}, \mathrm{GP}) \rightarrow M$ We assume the ciphertext is encrypted under an access matrix (A, ρ). To decrypt, the decryptor first obtains $H(\mathrm{GID})$ from the random oracle. If the decryptor has the secret keys $\{\mathrm{K}_{\rho(x),\mathrm{GID}}\}$ for a subset of rows A_x of A such that $(1, 0, \ldots, 0)$ is in the span of these rows, then the decryptor proceeds as follows. For each such x, the decryptor computes:

$$C_{1,x} \cdot e(H(\mathrm{GID}), C_{3,x}) / e(\mathrm{K}_{\rho(x),\mathrm{GID}}, C_{2,x}) = e(g,g)^{\lambda_x} e(H(\mathrm{GID}), g)^{\omega_x}.$$

The decryptor then chooses constants $c_x \in \mathbb{Z}_p$ such that $\sum_x c_x A_x = (1, 0, \ldots, 0)$ and computes:

$$\prod_x \left(e(g,g)^{\lambda_x} e(H(\mathrm{GID}), g)^{\omega_x} \right)^{c_x} = e(g,g)^s.$$

(We recall that $\lambda_x = A_x \cdot v$ and $\omega_x = A_x \cdot w$, where $v \cdot (1, 0, \ldots, 0) = s$ and $w \cdot (1, 0, \ldots, 0) = 0$.) The message can then be obtained as:

$$M = C_0 / e(g,g)^s.$$

E Proof of Security in the Generic Group and Random Oracles Models

We will now prove our prime order group scheme is secure in the generic bilinear group model previously used in [11, 49, 8], modeling H as a random oracle. Security in this model assures us that an adversary cannot break our scheme with only black-box access to the group operations and H.

We describe the generic bilinear model as in [11]. We let ψ_0 and ψ_1 be two random encodings of the additive group $\mathbb{Z}_p$. More specifically, each of ψ_0, ψ_1 is an injective map from $\mathbb{Z}_p$ to $\{0,1\}^m$, for $m > 3\log(p)$. We define the groups $G_0 = \{\psi_0(x) : x \in \mathbb{Z}_p\}$ and $G_1 = \{\psi_1(x) : x \in \mathbb{Z}_p\}$. We assume we have access to oracles which compute the induced group operations in G_0 and G_1 and an oracle which computes a non-degenerate bilinear map $e : G_0 \times G_0 \rightarrow G_1$. We refer to G_0 as a generic bilinear group.

In our security game, the attacker must distinguish between $C_0 = M_0 e(g,g)^s$ and $C_0 = M_1 e(g,g)^s$. We can alternatively consider a modified game, where the attacker must distinguish between $C_0 = e(g,g)^s$ or $C_0 = e(g,g)^t$, for t chosen uniformly randomly from $\mathbb{Z}_p$. This is the same modification employed in [8], and it is justified by a simple hybrid argument.

We will simplify our notation as follows. We let g denote $\psi_0(1)$, and g^x denote $\psi_0(x)$. We let $e(g,g)$ denote $\psi_1(1)$, and $e(g,g)^y$ denote $\psi_1(y)$.

We now simulate the modified security game in the generic bilinear group model where C_0 is set to be $e(g,g)^t$. We let S denote the set of all authorities, and U denote the universe of attributes. The simulator runs the global setup algorithm, and gives g to the attacker. The attacker chooses a set $S' \subset S$ of corrupted authorities, and reveals these to the simulator. The simulator randomly chooses values $\alpha_i, y_i \in \mathbb{Z}_p$ for the attributes $i \in U$ controlled by uncorrupted authorities, and it queries the group oracles for each $g^{y_i}, e(g,g)^{\alpha_i}$ and gives these to the attacker.

When the attacker requests $H(\mathrm{GID})$ for some GID for the first time, the simulator chooses a random value $h_{\mathrm{GID}} \in \mathbb{Z}_p$, queries the group oracle for $g^{h_{\mathrm{GID}}}$, and gives this value to the attacker as $H(\mathrm{GID})$. It stores this value so that it can reply consistently to any subsequent requests for $H(\mathrm{GID})$.

When the attacker requests a key $\mathrm{K}_{i,\mathrm{GID}}$ for some attribute i and identity GID, the simulator computes $g^{\alpha_i} H(\mathrm{GID})^{y_i}$ using the group oracle and supplies this to the attacker. If $H(\mathrm{GID})$ has not been requested before, it is determined as above.

At some point, the attacker specifies an access matrix (A, ρ) for the challenge ciphertext and additionally supplies the simulator with the $g^{y_i}, e(g,g)^{\alpha_i}$ values for any attributes i controlled by corrupt authorities that appear in the image of ρ on the rows of A. The simulator then checks that these are valid group elements by querying the group oracles.

The simulator must now produce the challenge ciphertext. To do so, the simulator chooses random values $s, v_2, \ldots, v_\ell \in \mathbb{Z}_p$ and sets the sharing vector $v = (s, v_2, \ldots, v_\ell)$. It then computes the shares $\lambda_x = A_x \cdot v$. The simulator then chooses a random vector $w = (0, w_2, \ldots, w_\ell)$, where each w_j is chosen randomly from $\mathbb{Z}_p$. It sets $\omega_x = A_x \cdot w$. The simulator also chooses random values $r_x \in \mathbb{Z}_p$ for each row A_x of A, and a random value t from $\mathbb{Z}_p$. Using the group oracles, the simulator can now compute:

$$C_0 = e(g,g)^t,\ C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} r_x},\ C_{2,x} = g^{r_x},\ C_{3,x} = g^{y_{\rho(x)} r_x} g^{\omega_x} \forall x.$$

The challenge ciphertext is given to the attacker.

We will argue that will all but negligible probability, the attacker's view in the simulation is identically distributed to what it's view would have been if C_0 had been set to $e(g,g)^s$ instead of $e(g,g)^t$. This shows that the attacker cannot attain a non-negligible advantage in the modified security game, and hence cannot attain a non-negligible advantage in the real security game.

We condition on the event that each of the attacker's queries to the group oracles have input values that were given to the attacker during the simulation or were received from the oracles in response to previous queries. This event occurs with high probability. Since each ψ_0, ψ_1 is a random injective map from $\mathbb{Z}_p$ into a set of $> p^3$ elements, the probability of the attacker being able to guess an element in the image of ψ_0, ψ_1 which it has not previously obtained is negligible.

Under this condition, we can think of each of the attacker's queries as a multi-variate polynomial in the variables $t, y_i, \alpha_i, \gamma_x, r_x, \omega_x, h_{\mathrm{GID}}$, where i ranges over the attributes controlled by uncorrupted authorities, x ranges over the rows of the challenge access matrix, and GID ranges over the allowed identities. (We can also think of γ_x, ω_x as linear combinations of the variables $s, v_2, \ldots, v_\ell, w_2, \ldots, w_\ell$.)

We now further condition on the event that for each pair of queries the attacker makes corresponding to different polynomials, the attacker receives different answers. In other words, we are conditioning on the event that our random assignment of values to the variables t, y_i, α_i, s, $v_2, \ldots, v_\ell, r_x, w_2, \ldots, w_\ell, h_{\mathrm{GID}}$ does not happen to be a zero of the difference of two query polynomials. (Here, we are treating γ_x as a linear combination of the variables $s, v_2, \ldots, v_\ell$ and ω_x as a linear combination of the variables $w_2, \ldots, w_\ell$.) This event occurs with high probability, which we can see by using the Schwartz-Zippel lemma and a union bound, since our polynomials have degree at most 4 (which we will see below when we enumerate all the types of queries the attacker can make).

Since t only appears as $e(g,g)^t$, the only queries the attacker can make involving t are of the form $ct+$ other terms, where c is a constant. The attacker's view can only differ when $t = s$ if the attacker can make two queries f and f' into G_1 where these are unequal as polynomials but become the same when we substitute s for t. This implies $f - f' = cs - ct$ for some constant c. We may conclude that the attacker can then make the query cs.

We will now show the attacker cannot make a query of the form cs, and therefore arrive at a contradiction. By examining the values given to the attacker during the simulation, we see that the attacker can only form queries which are linear combinations of $1, t$, and the terms appearing in Table 1.

We note that the attacker additionally knows the values of α_i, y_i for attributes i which are controlled by corrupt authorities, so these are known constants which can appear in coefficients of the terms in Table 1 in a linear combination.

Table 1: Possible query terms

α_i	$y_i h_{\mathrm{GID}}$	$y_i y_j$
y_i	$\alpha_i y_j + h_{\mathrm{GID}} y_i y_j$	$h_{\mathrm{GID}} h_{\mathrm{GID}'}$
h_{GID}	$y_i r_x$	$r_x r_{x'}$
$\alpha_i + h_{\mathrm{GID}} y_i$	$y_i y_{\rho(x)} r_x + y_i \omega_x$	$(\alpha_i + h_{\mathrm{GID}} y_i)(y_{\rho(x)} r_x + \omega_x)$
$\lambda_x + \alpha_{\rho(x)} r_x$	$h_{\mathrm{GID}} \alpha_i + h_{\mathrm{GID}} h_{\mathrm{GID}'} y_i$	$r_x y_{\rho(x')} r_{x'} + r_x \omega_{x'}$
r_x	$h_{\mathrm{GID}} r_x$	$(\alpha_i + h_{\mathrm{GID}} y_i)(\alpha_j + h_{\mathrm{GID}'} y_j)$
$y_{\rho(x)} r_x + \omega_x$	$h_{\mathrm{GID}} y_{\rho(x)} r_x + h_{\mathrm{GID}} \omega_x$	$(y_{\rho(x)} r_x + \omega_x)(y_{\rho(x')} r_{x'} + \omega_{x'})$
	$r_x \alpha_i + r_x h_{\mathrm{GID}} y_i$	

We recall that $\lambda_x = A_x \cdot v$, where $v = (s, v_2, \ldots, v_\ell)$. Since these are the only appearances of s in the above table, in order to form a query cs the attacker must choose constants β_x such that $\sum_x \lambda_x = cs$ and form:

$$\sum_x \beta_x (\lambda_x + \alpha_{\rho(x)} r_x).$$

For any terms $\beta_x \alpha_{\rho(x)} r_x$ where $\rho(x)$ is an attribute controlled by a corrupt authority, the attacker knows the value $\alpha_{\rho(x)}$, and so can form the term $-\beta_x \alpha_{\rho(x)} r_x$ in order to cancel this from the above polynomial. For terms $\beta_x \alpha_{\rho(x)} r_x$ where $\rho(x)$ is an attributed controlled by an uncorrupted authority, the attacker must cancel this term by using:

$$-\beta_x (r_x \alpha_{\rho(x)} + r_x h_{\mathrm{GID}} y_{\rho(x)}),$$

which leaves an additional term of $-\beta_x r_x h_{\mathrm{GID}} y_{\rho(x)}$ to be canceled. We also note that the attacker only has access to a term $r_x \alpha_{\rho(x)} + r_x h_{\mathrm{GID}} y_{\rho(x)}$ if it requested a key for the attribute, identity pair $(\rho(x), \mathrm{GID})$.

The extra term $-\beta_x r_x h_{\mathrm{GID}} y_{\rho(x)}$ can only be canceled by using:

$$\beta_x (h_{\mathrm{GID}} y_{\rho(x)} r_x + h_{\mathrm{GID}} \omega_x),$$

which leaves behind the term $\beta_x h_{\mathrm{GID}} \omega_x$. The collection of these terms for each identity GID will only cancel if the length ℓ vector $(1, 0, \ldots, 0)$ is in the span of the rows A_x of A belonging to corrupt authorities or for which the attacker obtained keys for $(\rho(x), \mathrm{GID})$. If this condition is satisfied for some GID, then the attacker has broken the rules of the security game and requested a collection of keys for a single identity that is capable of decrypting the challenge ciphertext.

Hence, we have shown that the attacker cannot construct a query of the form cs for a constant c. Therefore, under conditions that hold with all but negligible probability, the attacker's view when t is random is the same as the attacker's view when $t = s$. This proves that the attacker cannot attain non-negligible advantage in the security game.

F Generic Security of Our Assumptions

We now prove our four complexity assumptions hold in the generic bilinear group model, assuming it is hard to find a nontrivial factor of the group order, N. We use the notation of [36] to express our assumptions. If we fix generators $g_{p_1}, g_{p_2}, g_{p_3}$ of the subgroups $G_{p_1}, G_{p_2}, G_{p_3}$ respectively, every element of G can then be expressed as $g_{p_1}^{a_1} g_{p_2}^{a_2} g_{p_3}^{a_3}$ for some values of a_1, a_2, a_3. We denote an element of G by (a_1, a_2, a_3). The element $e(g_{p_1}, g_{p_1})^{a_1} e(g_{p_2}, g_{p_2})^{a_2} e(g_{p_3}, g_{p_3})^{a_3}$ in G_T

will be denoted by $[a_1, a_2, a_3]$. We use capital letters to denote random variables, and we reuse random variables to denote relationships between elements. For example, $X = (X_1, Y_1, Z_1)$ is a random element of G, and $Y = (X_1, Y_2, Z_2)$ is another random element that shares the same component in the G_{p_1} subgroup.

Given random variables $X, \{A_i\}$ expressed in this form, we say that X is *dependent* on $\{A_i\}$ if there exist values $\lambda_i \in \mathbb{Z}_n$ such that $X = \sum_i \lambda_i A_i$ as formal random variables. Otherwise, we say that X is *independent* of $\{A_i\}$. We note the following two theorems from [36]:

Theorem 14. *(Theorem A.1 of [36]) Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{A_i\}$ be random variables over G, and let $\{B_i\}, T_0, T_1$ be random variables over G_T, where all random variables have degree at most t. Consider the following experiment in the generic group model:*

An algorithm is given $N, \{A_i\}$, and $\{B_i\}$. A random bit b is chosen, and the adversary is given T_b. The algorithm outputs a bits b', and succeeds if $b' = b$. The algorithm's advantage is the absolute value of the difference between its success probability and $\frac{1}{2}$.

Say each of T_0 and T_1 is independent of $\{B_i\} \cup \{e(A_i, A_j)\}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ in the above experiment, $\mathcal{A}$ can be used to find a nontrivial factor of N (in time polynomial in λ and the running time of $\mathcal{A}$) with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.

Theorem 15. *(Theorem A.2 of [36]) Let $N = \prod_{i=1}^m p_i$ be a product of distinct primes, each greater than 2^λ. Let $\{A_i\}, T_0, T_1$ be random variables over G, and let $\{B_i\}$ be random variables over G_T, where all random variables have degree at most t. Consider the same experiment as in the theorem above.*

Let $S := \{i | e(T_0, A_i) \neq e(T_1, A_i)\}$ (where inequality refers to inequality as formal polynomials). Say each of T_0 and T_1 is independent of $\{A_i\}$, and furthermore that for all $k \in S$ it holds that $e(T_0, A_k)$ is independent of $\{B_i\} \cup \{e(A_i, A_j)\} \cup \{e(T_0, A_i)\}_{i \neq k}$, and $e(T_1, A_k)$ is independent of $\{B_i\} \cup \{e(A_i, A_j)\} \cup \{e(T_1, A_i)\}_{i \neq k}$. Then given any algorithm $\mathcal{A}$ issuing at most q instructions and having advantage δ, the algorithm can be used to find a nontrivial factor of N (in time polynomial in λ and the running time of $\mathcal{A}$) with probability at least $\delta - \mathcal{O}(q^2 t/2^\lambda)$.

We apply these theorems to prove the security of our assumptions in the generic group model.

Assumption 1 We apply Theorem 15. We can write this assumption as:

$$A_1 = (1, 0, 0)$$

$$T_1 = (X_1, X_2, X_3),\ T_2 = (X_1, 0, 0).$$

We note that $S = \emptyset$ here. Both of T_1, T_2 are independent of A_1 because X_1 does not appear in A_1, so Assumption 1 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 2 We apply Theorem 15. We can write this assumption as:

$$A_1 = (1, 0, 0),\ A_2 = (0, 0, 1),\ A_3 = (X_1, 1, 0)$$

$$T_1 = (Y_1, 0, 0),\ T_2 = (Y_1, Y_2, 0).$$

We note that $S = \{3\}$. We have that T_1, T_2 are both independent of $\{A_i\}$, because Y_1 does not appear in any of the A_i's. We also have that $e(T_1, A_3)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_1, A_i)\}_{i \neq 3}$ because it is impossible to have $X_1 Y_1$ as the first coordinate of a linear combination of these elements. For the same reason, $e(T_2, A_3)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_2, A_i)\}_{i \neq 3}$, so Assumption 2 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 3 We apply Theorem 15. We can write this assumption as:

$$A_1 = (1, 0, 0),\ A_2 = (X_1, 0, X_3),\ A_3 = (0, Y_2, Y_3)$$

$$T_1 = (Z_1, Z_2, 0),\ T_2 = (Z_1, 0, Z_3).$$

In this case, $S = \{2, 3\}$. We have that T_1, T_2 are independent of $\{A_i\}$ because Z_1 does not appear in the A_i's. We also have that $e(T_1, A_2)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_1, A_i)\}_{i \neq 2}$ because no linear combination of these elements will have $Z_1 X_1$ as a first coordinate. For the same reason, $e(T_2, A_2)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_2, A_i)\}_{i \neq 2}$. Similarly, $e(T_1, A_3)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_1, A_i)\}_{i \neq 3}$ because no linear combination of these terms will have $Z_2 Y_2$ as a second coordinate. Also $e(T_2, A_3)$ is independent of $\{e(A_i, A_j)\} \cup \{e(T_2, A_i)\}_{i \neq 3}$ because no linear combination of these terms will have $Z_3 Y_3$ as its third coordinate. Thus, Assumption 3 is generically secure, assuming it is hard to find a nontrivial factor of N.

Assumption 4 We apply Theorem 14. We can write this assumption as:

$$A_1 = (1, 0, 0),\ A_2 = (0, 1, 0),\ A_3 = (0, 0, 1),\ A_4 = (A, 0, 0),$$

$$A_5 = (B, 0, B),\ A_6 = (C, 0, 0),\ A_7 = (AC, 0, D)$$

$$T_1 = [ABC, 0, 0],\ T_2 = [X_1, X_2, X_3].$$

We note that T_2 is independent of $\{e(A_i, A_j)\}$ because X_1, X_2, X_3 do not appear in the A_i's. T_1 is also independent of $\{e(A_i, A_j)\}$ because the only way to obtain ABC is the first coordinate is to take $e(A_5, A_7)$, which leaves BD in the third coordinate, and this cannot be canceled out. Hence, Assumption 4 is generically secure, assuming it is hard to find a nontrivial factor of N.

G Converting from Boolean Formulas to LSSS Matrices

We now describe a general algorithm for converting a boolean formula into an equivalent LSSS matrix. We consider the boolean formula as an access tree, where interior nodes are AND and OR gates and the leaf nodes correspond to attributes. We will use $(1, 0, \ldots, 0)$ as the sharing vector for the LSSS matrix. We begin by labeling the root node of the tree with the vector (1) (a vector of length 1). We then go down the levels of the tree, labeling each node with a vector determined by the vector assigned to its parent node. We maintain a global counter variable c which is initialized to 1.

If the parent node is an OR gate labeled by the vector v, then we also label its children by v (and the value of c stays the same). If the parent node is an AND gate labeled by the vector v, we pad v with 0's at the end (if necessary) to make it of length c. Then we label one of its children with the vector $v|1$ (where $|$ denotes concatenation) and the other with the vector $(0, \ldots, 0)| - 1$, where $(0, \ldots, 0)$ denotes the zero vector of length c. Note that these two vectors sum to $v|0$. We now increment the value of c by 1. Once we have finished labeling the entire tree, the vectors labeling the leaf nodes form the rows of the LSSS matrix. If these vectors have different lengths, we pad the shorter ones with 0's at the end to arrive at vectors of the same length.

For example, we consider the formula $A\ AND\ (D\ OR\ (B\ AND\ C))$. The root AND node of this tree is labeled (1). Its left child, the leaf node corresponding to A, is labeled $(1, 1)$. Its right child, the OR node, is labeled $(0, -1)$. The left child of the OR node corresponds to D and is labeled $(0, -1)$. Its right child is an AND node and is labeled $(0, -1)$. The left child of

the AND node, which corresponds to B, is labeled $(0, -1, 1)$, and the right child, corresponding to C, is labeled $(0, 0, -1)$. The resulting LSSS matrix is:

$$\begin{pmatrix} 1 & 1 & 0 \\ 0 & -1 & 1 \\ 0 & 0 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

(where the rows correspond to A, B, C, and D respectively). We note that each subset of the rows of this matrix includes $(1, 0, 0)$ in its span if and only if the corresponding attributes satisfy the formula $A\ AND\ (D\ OR\ (B\ AND\ C))$.