Attribute-Based Encryption with Non-Monotonic Access Structures

Rafail Ostrovsky Amit Sahai* Brent Waters †

Abstract

We construct an Attribute-Based Encryption (ABE) scheme that allows a user's private key to be expressed in terms of *any* access formula over attributes. Previous ABE schemes were limited to expressing only monotonic access structures. We provide a proof of security for our scheme based on the Decisional Bilinear Diffie-Hellman (BDH) assumption. Furthermore, the performance of our new scheme compares favorably with existing, less-expressive schemes.

1 Introduction

Several distributed file and information systems require complex access-control mechanisms, where access decisions depend upon attributes of the protected data and access policies assigned to users. Traditionally, such access-control mechanisms have been enforced by a server that acts as a trusted reference monitor; the monitor will allow a user to view data only if his access policy allows it. While the use of trusted servers allows for a relatively straightforward solution, there is a large downside to this approach — both the servers and their storage must be trusted and remain uncompromised. With the increasing number of worm attacks and other forms of intrusion, maintaining the security of any particular host is becoming increasingly difficult. This problem is exacerbated in larger systems where sensitive data must be replicated across several servers because of scalability and survivability concerns.

A natural solution to this problem is to encrypt stored data in order to reduce data vulnerability in the event that a storage server is compromised. However, traditional public-key encryption methods require that data be encrypted to one particular user's public key and are unsuitable for expressing more complex access control policies.[1]

Attribute-Based Encryption. Recently, Sahai and Waters [21] addressed this issue by introducing the concept of Attribute-Based Encryption (ABE). In Attribute-Based Encryption an encryptor will associate encrypted data with a set of attributes. An authority will issue users different private

*This research was supported in part by an Alfred P. Sloan Foundation Research Fellowship, an Intel equipment grant, a Cyber-TA Army grant, and NSF ITR/Cybertrust grants 0205594, 0456717 and 0627781.

†Supported by NSF CNS-0524252 and the US Army Research Office under the CyberTA Grant No. W911NF-06-1-0316.

[1]There have been several proposals for achieving greater access control from public key systems (see, e.g. [24, 9]). However, these systems were unable to achieve the critical property of security against *collusion* attacks, where multiple users share their private key information. Indeed, simple and devastating collusion attacks are easy to mount against the systems of [24, 9] involving as few as two colluding users. In this paper, we focus only on solutions that are able to provide security against collusion attacks.

keys, where a user's private key is associated with an access structure over attributes and reflects the access policy ascribed to the user.

The original ABE construction of Sahai and Waters is somewhat limited in that it only permits an authority to issue private keys that express threshold access policies, in which a certain number of specified attributes need to be present in the ciphertext in order for a user to decrypt. Goyal et al. [16] greatly increased the expressibility of Attribute-Based Encryption systems by creating a new ABE scheme in which users' private keys can express any monotone access formula consisting of **AND**, **OR**, or threshold gates.

While the work of Goyal et al. is a large step forward in the capability of Attribute-Based Encryption systems, one fundamental limitation of their techniques is that there is no satisfactory method to represent *negative* constraints in a key's access formula. This is particularly a problem in scenarios where conflicts of interest naturally arise. Consider the following example. A university is conducting a peer-review evaluation, where each department will be critiqued by a panel of professors from other departments. Bob, who is a member of the panel this year from the Biology department, will need to read (possibly sensitive) comments about other departments and assimilate them for his written review. In an Attribute-Based Encryption system the comments will be labeled with descriptive attributes; for example, a comment on the History department might be encrypted with the attributes: "HISTORY","YEAR=2007","DEPT-REVIEW". In the Goyal et al. scheme Bob might receive a private key for the policy "YEAR=2007" **AND** "DEPT-REVIEW", which would allow him to see all comments from this current year. However, in this setting it is important that Bob should not be able to view comments written about his own department. Therefore, the policy we would *actually* like to ascribe to Bob's key is "YEAR=2007" **AND** "DEPT-REVIEW" **AND** (**NOT** "BIOLOGY").

One way that we might try to handle this issue is to include explicit attributes that indicate the *absence* of attributes in the ciphertext. For example, the attribute "NOT:BIOLOGY" can be included in a ciphertext to indicate that the ciphertext is not related to the Biology department. However, this solution is undesirable for two reasons. First, the ciphertext overhead will become huge in many applications as it needs to explicitly include negative attributes for *everything* that it does not relate to. The feedback about the History department would need to include the attributes "NOT:AERONAUTICS", "NOT:ANTHROPOLOGY", "NOT:ART HISTORY",..., "NOT:WORLD STUDIES" as well as explicit negative attributes for every subject that does not describe the ciphertext. In addition, a user encrypting a message might not be aware of many attributes, and new attributes might come into use in the system after the ciphertext is created. In our example, a user creating a comment on the History department might be unaware of a newly created Otolaryngology[2] department.

The above example illustrates the limitations on system design imposed by the inability of current ABE systems to effectively support negation. Indeed, this limitation appears to be a fundamental characteristic of current ABE systems, which use techniques from secret-sharing schemes as a core component of their design. It is well known that secret-sharing schemes are limited to expressing monotonic access structures because a participating party can always choose not to contribute his share and therefore act like he is not present.

Our Contribution. In this work we present a new Attribute-Based Encryption scheme where private keys can represent *any* access formula over attributes, including non-monotone ones. In

[2]Otolaryngology is the branch of medicine that specializes in ear, nose, throat, head, and neck disorders.

particular, our construction can handle any access structure that can be represented by a boolean formula involving AND, OR, NOT, and threshold operations.

As mentioned above, the main technical obstacle we overcome is finding a way to make use of secret sharing schemes to yield non-monotonic access structures. At a high level, the technical novelty in our work lies in finding a way to (implicitly) make a share "available" to the decryptor only if a given attribute is *not* present among the attributes of the ciphertext. To accomplish this we adapt an idea from the broadcast revocation scheme of Naor and Pinkas [18] to our setting of Attribute-Based Encryption based on bilinear groups. Every negative attribute node in a key is tied to a degree d polynomial (in the exponent) that was created by the authority at setup (where d is the maximum number of attributes used to describe a ciphertext). To access the secret share corresponding to this node, the decryptor will need to make use of at least $d+1$ different points from the polynomial in order to perform an interpolation, where we map attributes to distinct points on the polynomial. The decryption algorithm will be able to gather d different points of the polynomial from the attributes of the ciphertext. To get the remaining point, the decryptor must examine the one point that corresponds to the negative attribute in this particular node of the access formula. If this attribute is distinct from all the attributes in the ciphertext — that is, if the attribute is *not* present — then the decryptor will have $d+1$ points of the polynomial and be able to decrypt; otherwise, if the key's attribute appears in the ciphertext, then the decryption algorithm will have only d points (one particular point will have been given twice) and the decryption algorithm will not be able to interpolate the polynomial and thereby access the secret share corresponding to the node. In designing our construction several challenges arise from adapting these negation techniques while preserving the collusion resistance features that are necessary for Attribute-Based Encryption systems.

1.1 Related Work

Sahai and Waters [21] introduced the concept of Attribute-Based Encryption, as we use the term here (see below for a brief discussion of other related notions). In ABE systems an encrypted ciphertext is associated with a set of attributes, and a user's private key will reflect an access policy over attributes. A user will be able to decrypt if and only if the ciphertext's attributes satisfy the key's policy. Attribute-Based Encryption is closely related to the concept of Identity-Based Encryption (IBE) [7, 23, 15], which was introduced by Shamir in 1984 [23]. One can actually view IBE as a special case of ABE in which ciphertexts are associated with one attribute, the "identity" of the recipient, and a private key's policy demands that one particular attribute, the key holder's identity, be present in the ciphertext for decryption.

The original construction of Sahai and Waters [21] was limited to expressing threshold access policies. Goyal et al. [16] subsequently increased the expressibility of ABE systems by allowing the private key to express any monotonic access structure over attributes.

Other works have examined different variants of ABE. Pirretti et al. [19] examined methods for applying the Sahai-Waters system into practice and gave an implementation of the construction. Chase [13] gave a "multi-authority" construction in which a user's key is constructed by combining components received from different authorities. Bethencourt, Sahai, and Waters [4] gave a construction for "Ciphertext-Policy" Attribute-Based Encryption. In their construction the roles of the ciphertexts and keys are reversed in the sense that attributes are used to describe the features of a key holder, and an encryptor will associate an access policy with the ciphertext.

Attribute-Based Encryption makes use of techniques from secret-sharing schemes [17, 10, 22,

5, 3]. The idea of combining secret-sharing schemes and encryption to achieve access control with respect to policies has a long history (for some recent work in this direction, see [24, 9]). In this previous work, what we call "collusion" was actually seen as a desirable feature – it would be necessary for multiple entities with different attributes/credentials to come together in order to access encrypted data. This is of course problematic in our scenario; indeed, the elusive property of resistance to collusion attacks is considered a defining property of the Sahai-Waters notion of ABE.

1.2 Organization

In Section 2 we give background information on our security definitions and assumptions. Next, we give our construction in Section 3. Then, we prove our scheme secure in Section 4. Finally, we conclude in Section 5.

2 Background

We first give formal definitions for the security of (key-policy) Attribute-Based Encryption (ABE), following [21, 16]. Then we give background information on bilinear maps and our cryptographic assumption. Finally, we give some background on linear secret-sharing schemes.

2.1 Definitions

Definition 1 (Access Structure [2]) *Let $\{P_1, P_2, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$ then $C \in \mathbb{A}$. An* access structure *(respectively, monotonic access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,P_2,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the* authorized sets*, and the sets not in $\mathbb{A}$ are called the* unauthorized sets*.*

A (key-policy) Attribute-Based Encryption scheme consists of four algorithms.

Setup. This is a randomized algorithm that takes no input other than the implicit security parameter. It outputs the public parameters PK and a master key MK.

Encryption. This is a randomized algorithm that takes as input a message M, a set of attributes γ, and the public parameters PK. It outputs the ciphertext E.

Key Generation. This is a randomized algorithm that takes as input an access structure $\mathbb{A}$, the master key MK, and the public parameters PK. It outputs a decryption key D.

Decryption. This algorithm takes as input the ciphertext E that was encrypted under a set γ of attributes, the decryption key D for access control structure $\mathbb{A}$, and the public parameters PK. It outputs the message M if $\gamma \in \mathbb{A}$.

We now discuss the security of an ABE scheme. Following [21, 16], we define the selective-set model for proving the security of the attribute based under chosen plaintext attack. This model can be seen as analogous to the selective-ID model [11, 12, 6] used in identity-based encryption

(IBE) schemes [23, 7, 15].

Selective-Set Model for ABE

Init The adversary declares the set of attributes, γ, that he wishes to be challenged upon.
Setup The challenger runs the Setup algorithm of ABE and gives the public parameters to the adversary.
Phase 1 The adversary is allowed to issue queries for private keys for many access structures $\mathbb{A}_j$, where $\gamma \notin \mathbb{A}_j$ for all j.
Challenge The adversary submits two equal-length messages M_0 and M_1. The challenger flips a random coin b, and encrypts M_b with γ. The ciphertext is passed to the adversary.
Phase 2 Phase 1 is repeated.
Guess The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

We note that the model can easily be extended to handle chosen-ciphertext attacks by allowing for decryption queries in Phase 1 and Phase 2.

Definition 2 *An attribute-based encryption scheme is secure in the selective-set model of security if all polynomial time adversaries have at most a negligible advantage in the selective-set game.*

2.2 Bilinear Maps

We present a few facts related to groups with efficiently computable bilinear maps.

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a generator of $\mathbb{G}$ and e be a bilinear map, $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. The bilinear map e has the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, we have $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non degeneracy: $e(g, g) \neq 1$.

We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ are both efficiently computable. Notice that the map e is symmetric since $e(g^a, g^b) = e(g, g)^{ab} = e(g^b, g^a)$.

2.3 The Decisional Bilinear Diffie-Hellman (BDH) Assumption

Let $a, b, c, z \in \mathbb{Z}_p$ be chosen at random and g be a generator of $\mathbb{G}$. The decisional BDH assumption [6, 21] is that no probabilistic polynomial-time algorithm $\mathcal{B}$ can distinguish the tuple $(g, A = g^a, B = g^b, C = g^c, e(g, g)^{abc})$ from the tuple $(g, A = g^a, B = g^b, C = g^c, e(g, g)^z)$ with more than a negligible advantage. The advantage of $\mathcal{B}$ is

$$\left| \Pr[\mathcal{B}(A, B, C, e(g, g)^{abc}) = 0] - \Pr[\mathcal{B}(A, B, C, e(g, g)^z)] = 0 \right|$$

where the probability is taken over the random choice of the generator g, the random choice of a, b, c, z in $\mathbb{Z}_p$, and the random bits consumed by $\mathcal{B}$.

2.4 Linear Secret-Sharing Schemes

We will make essential use of linear secret-sharing schemes. We adapt our definitions from those given in [2]:

Definition 3 (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* M *called the share-generating matrix for* Π*. The matrix* M *has* ℓ *rows and* $n+1$ *columns. For all* $i = 1, \ldots, \ell$*, the* i*'th row of* M *is labeled with a party named* $\breve{x}_i \in \mathcal{P}$*. When we consider the column vector* $v = (s, r_1, r_2, \ldots, r_n)$*, where* $s \in \mathbb{Z}_p$ *is the secret to be shared, and* $r_1, \ldots, r_n \in \mathbb{Z}_p$ *are randomly chosen, then* Mv *is the vector of* ℓ *shares of the secret* s *according to* Π*. The share* $(Mv)_i$ *belongs to party* $\breve{x}_i$*.*

It is shown in [2] that every linear secret sharing-scheme according to the above definition also enjoys the *linear reconstruction* property, defined as follows: Suppose that Π is an LSSS for the access structure $\mathbb{A}$. Let $S \in \mathbb{A}$ be any authorized set, and let $I \subset \{1, 2, \ldots, \ell\}$ be defined as $I = \{i : \breve{x}_i \in S\}$. Then, there exist constants $\{\omega \in \mathbb{Z}_p\}_{i \in I}$ such that, if $\{\lambda_i\}$ are valid shares of any secret s according to Π, then $\sum_{i \in I} \omega_i \lambda_i = s$.

Furthermore, it is shown in [2] that these constants $\{\omega_i\}$ can be found in time polynomial in the size of the share-generating matrix M.

3 Our Construction

In showing how to construct an Attribute-Based Encryption system with non-monotone access formulas, we begin by describing a "core" construction, in which the we assume that *every* ciphertext is annotated with *exactly* d attributes. We then show how to remove that restriction and still achieve systems parameters that compare favorably with the less-expressive ABE system of Goyal et al. [16].

We choose to first describe our construction in generality; we describe our access policies in terms of monotonic access structures with negative attributes. (This will actually allow for more general policies than non-monotone formulas.) Later, we show how to instantiate our constructions to yield ABE schemes for any (monotone or non-monotone) boolean formula.

Moving from monotonic access structures to non-monotonic access structures. As alluded to in the introduction we can think about ABE non-monotonic access structures in terms of ABE monotonic access structures with negative attributes. The challenge in designing our construction will be how to realize this concept without requiring a ciphertext to explicitly include negative attributes for each attribute not present. Before we describe our construction we develop some notation for describing how non-monotonic access structures can be described in terms of monotonic access structures with negative shares, *without blowing up the share sizes.*

Assume we are given a family of linear secret-sharing schemes $\{\Pi_{\mathbb{A}}\}_{\mathbb{A} \in \mathcal{A}}$ for a set of possible monotone access structures $\mathcal{A}$. Note that, of course, all access structures in $\mathcal{A}$ must necessarily be monotonic because these access structures correspond to secret-sharing schemes. However, we assume that for each access structure $\mathbb{A} \in \mathcal{A}$, the set of parties $\mathcal{P}$ underlying the access structure

has the following properties: The names of the parties in $\mathcal{P}$ may be of two types: either the name is normal (like x) or it is *primed* (like x'), and if $x \in \mathcal{P}$ then $x' \in \mathcal{P}$ and vice versa. We will conceptually associate primed parties as representing the negation of unprimed parties. We will sometimes write $\breve{x}$ to refer to a party in $\mathcal{P}$ that may be primed or unprimed.

Then, we can define the following family $\tilde{\mathcal{A}}$ of possibly non-monotonic access structures. For each access structure $\mathbb{A} \in \mathcal{A}$ over a set of parties $\mathcal{P}$, we define a possibly non-monotonic access structure $NM(\mathbb{A})$ over the set of parties $\tilde{\mathcal{P}}$, where $\tilde{\mathcal{P}}$ is the set of all unprimed parties in $\mathcal{P}$. First, for every set $\tilde{S} \subset \tilde{\mathcal{P}}$ we define $N(\tilde{S}) \subset \mathcal{P}$ as follows: First, all parties in $\tilde{S}$ are in $N(\tilde{S})$, so $\tilde{S} \subset N(\tilde{S})$. Second, for each party $x \in \tilde{\mathcal{P}}$ such that $x \notin \tilde{S}$, we have that $x' \in N(\tilde{S})$. Essentially, $N(\tilde{S})$ consists of all the parties in S plus the primes (or negation) of all the parties in the universe that are not included in S.

Finally, we define $NM(\mathbb{A})$ by specifying that $\tilde{S}$ is authorized in $NM(\mathbb{A})$ iff $N(\tilde{S})$ is authorized in $\mathbb{A}$. The set of these $NM(\mathbb{A})$ access structures is $\tilde{\mathcal{A}}$. Therefore, the non-monotonic access structure $NM(\mathbb{A})$ will have only unprimed parties in its access sets. For each access set X in $NM(\mathbb{A})$ there will be a set in $\mathbb{A}$ that has the elements in X plus primed elements for each party not in X.

We will show how to use a linear secret sharing scheme Π for the monotonic access structure $\mathbb{A}$ to yield an ABE key for the (possibly non-monotonic) access structure $NM(\mathbb{A})$. Again, *we stress that the share sizes of* Π *only depend on the size of the non-monotonic access structure* $NM(\mathbb{A})$.

Mathematical Background. Let $\mathbb{G}$ be a bilinear group of prime order p, and let g be a generator of $\mathbb{G}$. In addition, let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ denote the bilinear map. A security parameter, κ, will determine the size of the groups. We will also implicitly make use of Lagrange coefficients: for any $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$: define $\Delta_{i,S}(x) = \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$. We will associate each attribute with a unique element in $\mathbb{Z}_p^*$. (This could be accomplished by means of a collision-resistant hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$.)

Our main construction follows.

3.1 Main Construction

Setup(d)**.** In the basic construction, a parameter d specifies how many attributes *every* ciphertext has. (We will show later how this constraint can be removed with only a small loss in efficiency.) Two secrets α, β are chosen uniformly at random from $\mathbb{Z}_p$, and we denote $g_1 = g^\alpha$ and $g_2 = g^\beta$. In addition, two polynomials $h(x)$ and $q(x)$ of degree d are chosen at random subject to the constraint that $q(0) = \beta$. (There is no constraint on $h(x)$.) The public parameters PK are $(g, g_1;\ g_2 = g^{q(0)}, g^{q(1)}, g^{q(2)}, \ldots, g^{q(d)};\ \ g^{h(0)}, g^{h(1)}, \ldots, g^{h(d)})$. The master key MK is α.

These public parameters define two publicly computable functions $T, V : \mathbb{Z}_p \to \mathbb{G}$. The function $T(x)$ maps to $g_2^{x^d} \cdot g^{h(x)}$, and the function $V(x)$ maps to $g^{q(x)}$. Note that both $g^{h(x)}$ and $g^{q(x)}$ can be evaluated from the public parameters by interpolation in the exponent. (For further details on how to do this using Lagrange coefficients, see, e.g., [21, 16].)

Encryption (M, γ, PK). To encrypt a message $M \in \mathbb{G}_T$ under a set of d attributes $\gamma \subset \mathbb{Z}_p^*$, choose a random value $s \in \mathbb{Z}_p$ and output the ciphertext as

$$E = \left(\gamma, E^{(1)} = Me(g_1, g_2)^s, E^{(2)} = g^s, \{E_x^{(3)} = T(x)^s\}_{x \in \gamma}, \{E_x^{(4)} = V(x)^s\}_{x \in \gamma}\right)$$

Key Generation $(\tilde{\mathbb{A}}, \mathrm{MK}, \mathrm{PK})$. This algorithm outputs a key that enables the user to decrypt an encrypted message *only* if the attributes of that ciphertext satisfy the access structure $\tilde{\mathbb{A}}$. We require that the access structure $\tilde{\mathbb{A}}$ is $NM(\mathbb{A})$ for some monotonic access structure $\mathbb{A}$, over a set $\mathcal{P}$ of attributes, associated with a linear secret-sharing scheme Π. First, we apply the linear secret-sharing mechanism Π to obtain shares $\{\lambda_i\}$ of the secret α. We denote the party corresponding to the share λ_i as $\breve{x}_i \in \mathcal{P}$, where x_i is the attribute underlying $\breve{x}_i$. Note that $\breve{x}_i$ can be primed (negated) or unprimed (non negated). For each i, we also choose a random value $r_i \in \mathbb{Z}_p$.

The private key D will consist of the following group elements: For every i such that $\breve{x}_i$ is *not* primed (i.e., is a non-negated attribute), we have

$$D_i = (D_i^{(1)} = g_2^{\lambda_i} \cdot T(x_i)^{r_i}, D_i^{(2)} = g^{r_i})$$

For every i such that $\breve{x}_i$ is primed (i.e., is a negated attribute), we have

$$D_i = (D_i^{(3)} = g_2^{\lambda_i + r_i}, D_i^{(4)} = V(x_i)^{r_i}, D_i^{(5)} = g^{r_i})$$

The key D consists of D_i for all shares i.

Decryption (E, D). Given a ciphertext E and a decryption key D, the following procedure is executed: (All notation here is taken from the above descriptions of E and D, unless the notation is introduced below.) First, the key holder checks if $\gamma \in \tilde{\mathbb{A}}$ (we assume that this can be checked efficiently). If not, the output is $\perp$. If $\gamma \in \tilde{\mathbb{A}}$, then we recall that $\tilde{\mathbb{A}} = NM(\mathbb{A})$, where $\mathbb{A}$ is an access structure, over a set of parties $\mathcal{P}$, for a linear secret sharing-scheme Π. Denote $\gamma' = N(\gamma) \in \mathbb{A}$, and let $I = \{i : \breve{x}_i \in \gamma'\}$. Since γ' is authorized, an efficient procedure associated with the linear secret-sharing scheme yields a set of coefficients $\Omega = \{\omega_i\}_{i \in I}$ such that $\sum_{i \in I} \omega_i \lambda_i = \alpha$. (Note, however, that these λ_i are not known to the decryption procedure, so neither is α.)

For every positive (non negated) attribute $\breve{x}_i \in \gamma'$ (so $x_i \in \gamma$), the decryption procedure computes the following:

$$\begin{aligned} Z_i &= e\left(D_i^{(1)}, E^{(2)}\right) / e\left(D_i^{(2)}, E_i^{(3)}\right) \\ &= e\left(g_2^{\lambda_i} \cdot T(x_i)^{r_i}, g^s\right) / e\left(g^{r_i}, T(x)^s\right) \\ &= e\left(g_2, g\right)^{s\lambda_i} \end{aligned}$$

For every negated attribute $\breve{x}_i \in \gamma'$ (so $x_i \notin \gamma$), the decryption procedure computes the following: We consider the set $\gamma_i = \gamma \cup \{x_i\}$. Note that $|\gamma_i| = d+1$ and recall that the degree of the polynomial q underlying the function V is d. Using the points in γ_i as an interpolation set, compute Lagrangian

coefficients $\{\sigma_x\}_{x\in\gamma_i}$ such that $\sum_{x\in\gamma_i}\sigma_x q(x) = q(0) = \beta$. Now, perform the following computation:

$$\begin{aligned}
Z_i &= \frac{e\left(D_i^{(3)}, E^{(2)}\right)}{e\left(D_i^{(5)}, \prod_{x\in\gamma}\left(E_x^{(4)}\right)^{\sigma_x}\right)\cdot e\left(D_i^{(4)}, E^{(2)}\right)^{\sigma_{x_i}}} \\
&= \frac{e\left(g_2^{\lambda_i+r_i}, g^s\right)}{e\left(g^{r_i}, \prod_{x\in\gamma}\left(V(x)^s\right)^{\sigma_x}\right)\cdot e\left(V(x_i)^{r_i}, g^s\right)^{\sigma_{x_i}}} \\
&= \frac{e\left(g_2^{\lambda_i}, g^s\right)\cdot e\left(g_2^{r_i}, g^s\right)}{e\left(g^{r_i}, g^{s\sum_{x\in\gamma}\sigma_x q(x)}\right)\cdot e\left(g^{r_i\sigma_{x_i}q(x_i)}, g^s\right)} \\
&= \frac{e\left(g_2, g\right)^{s\lambda_i}\cdot e\left(g, g\right)^{r_i s\beta}}{e(g,g)^{r_i s\sum_{x\in\gamma'}\sigma_x q(x)}} \\
&= e\left(g_2, g\right)^{s\lambda_i}
\end{aligned}$$

Finally, the decryption is obtained by computing

$$\frac{E^{(1)}}{\prod_{i\in I} Z_i^{\omega_i}} = \frac{Me(g_2,g)^{s\alpha}}{e(g_2,g)^{s\alpha}} = M$$

Note on Efficiency. We note that encryption requires only a single pairing, which may be precomputed, regardless of the number of attributes associated with a ciphertext. We also note that decryption requires two or three pairings per share utilized in decryption, depending on whether the share corresponds to a non-negated attribute or a negated attribute, respectively.

3.2 Amortizing the Cost of Multiple Systems

In practice we might actually have several different Attribute-Based Encryption systems run by different authorities. In this setting we might want to minimize the size of the public key material that users need to maintain, since each authority will need to post its public key. We can mitigate this cost by using a shared trusted party and applying a similar technique to that of the Broadcast Encryption scheme of Boneh, Gentry, and Waters [8].

We first observe that once the public key material is published an authority only needs to know α in order to create private keys. In addition, only one element, $g_1 = g^\alpha$ depends upon α. Therefore, a trusted party then can create all public key material except g_1. This public key material will be shared across several different systems. An authority X that wishes to create his own system simply chooses his own α_X private key and creates a public key $g_{1,X} = g^{\alpha_X}$. A user encrypting a ciphertext under this authority's system will use $g_{1,X}$ in addition to the shared public key material. The added public key material for a whole new system is just one group element.

3.3 Removing Fixed Attribute Restriction

The drawback of using our main construction directly in a system is that it imposes a "one size fits all" restriction in that each ciphertext must have exactly d attributes. We describe how to get around these restrictions and maintain efficient performance.

First, we note that a ciphertext will often be associated with s attributes where s is less than d, the maximum number of attributes in our construction. A simple technique is for the encryption algorithm to create $d - s$ "filler" attributes for strings that have no semantic meaning in the system. For a ciphertext with s real attributes, the encryption algorithm can just add the attributes "FILLER:1", "FILLER:2", "FILLER:$d - s$".

A more problematic issue is that a system will need to accommodate ciphertexts that might have a large maximum, n, number of attributes. This will mean that ciphertexts with a relatively small number of attributes will have unnecessarily high ciphertext overhead. To mitigate this issue in a system we can use k different constructions that respectively accommodate $d_1, \ldots, d_k$ attributes. When encrypting a ciphertext with s attributes the decryption algorithm will simply use the encryption system with the smallest d_i such that $d_i \geq s$, and then only $d_i - s$ filler attributes will be necessary.

Consider the case when there are a maximum of n attributes for any ciphertext. For simplicity we assume $n = 2^k$ for some k. Then we can create a system that uses k parallel encryption systems, where encryption system i is set up for $d_i = 2^i$ attributes. The aggregate system has performance that compares favorably with existing systems: ciphertexts for s attributes will have $O(s)$ group elements, the public key material will consist of $O(n)$ group elements, and the private keys for an access structure of t shares will have $O(t \cdot \lg(n))$ group elements (a copy for each encryption system) is kept. We point out that all these efficiency parameters, other than the private key size, are identical to the less-expressive scheme of Goyal et al.[3]

3.4 Realizing Any Access Formula

Our main construction shows how to create private keys that can be represented by any linear secret-sharing scheme that uses both negative and non negative attributes. It is a relatively straightforward exercise to show that these techniques are powerful enough to express any access formula. To do so, we first use repeated applications of DeMorgan's law to transform any access formula into a monotonic one with negative attributes. Then, we can represent the access formula in terms of a secret-sharing scheme in a way similar to [16]. We leave the details of this transformation to Appendix A.

3.5 Ciphertext-Policy ABE

We also note that our techniques can be applied to the Ciphertext-Policy Attribute-Based Encryption (CP-ABE) scheme of Bethencourt, Sahai, and Waters [4]. The primary modification is that the polynomial for the revocation scheme will be embedded by the encryptor in the negated nodes of the encryption policy. The attributes will then be represented in the tree.

One disadvantage of the BSW scheme is that its proof is in the generic group model. This stems from the fact that their scheme allows for arbitrary access formulas in the ciphertext policy. Since the challenge ciphertext policy may be bigger than the public parameters, it is difficult to "program" the challenge ciphertext into the public parameters. However, for more restricted CP-ABE schemes that are less expressive there exists schemes proved on concrete assumptions. The original threshold scheme of Sahai and Waters [21] was written before the distinction of Key-Policy versus Ciphertext-Policy was made explicit; however, it can be interpreted in either way. Using the

[3]This claim applies to the Large Universe scheme of Goyal et al. that does not use the random oracle heuristic. The authors noted that in the random oracle model they can reduce the public parameter size.

Sahai-Waters large-universe construction we can realize a non-monotonic CP-ABE scheme with k-of-n threshold policies, where n is fixed and k can be determined by the encryptor by using "dummy attributes". Pirretti et al. [19] show tradeoffs that can be made between key and ciphertext sizes and Cheung and Newport provide another realization [14] with similar properties.

4 Proof of Security

We prove that the security of our main construction in the attribute-based selective-set model reduces to the hardness of the Decisional BDH assumption.

Theorem 1 *If an adversary can break our scheme with advantage ϵ in the attribute-based selective-set model of security, then a simulator can be constructed to play the Decisional BDH game with advantage $\epsilon/2$.*

PROOF:

Suppose there exists a polynomial-time adversary $\mathcal{A}$ that can attack our scheme in the selective-set model with advantage ϵ. We build a simulator $\mathcal{B}$ that can play the Decisional BDH game with advantage $\epsilon/2$. The simulation proceeds as follows:

We first let the challenger set the groups $\mathbb{G}$ and $\mathbb{G}_T$ with an efficient bilinear map, e. The challenger flips a fair binary coin μ, outside of $\mathcal{B}$'s view. If $\mu = 0$, the challenger sets $(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^{abc})$; otherwise, it sets $(g, A, B, C, Z) = (g, g^a, g^b, g^c, e(g,g)^z)$ for random a, b, c, z.

Init The simulator $\mathcal{B}$ runs $\mathcal{A}$. $\mathcal{A}$ chooses the challenge set, γ, a set of d members of $\mathbb{Z}_p^*$.

Setup The simulator assigns the public parameters $g_1 = A$ and $g_2 = B$, thereby implicitly setting $\alpha = a$ and $\beta = b$. It then chooses a random degree d polynomial $f(x)$ and fixes a degree d polynomial $u(x)$ as follows: set $u(x) = -x^d$ for all $x \in \gamma$ and $u(x) \neq -x^d$ for some (arbitrary) other $x \notin \gamma$. Because $-x^d$ and $u(x)$ are two degree d polynomials, they will have at most d points in common or they are the same polynomial. This construction ensures that $\forall x, u(x) = -x^d$ if and only if $x \in \gamma$.

The simulator will now implicitly set the polynomials h and q as follows: First, $h(x) = \beta u(x) + f(x)$. Now, let's write $\gamma = \{x_1, x_2, \ldots, x_d\}$. Then, the simulator chooses d points $\theta_{x_1}, \ldots, \theta_{x_d}$ uniformly at random from $\mathbb{Z}_p$, and implicitly sets $q(x)$ such that $q(0) = \beta$, while $q(x_i) = \theta_{x_i}$ for $i = 1, 2, \ldots, d$. Thus, the simulator outputs the following group elements for the public key: For $i = 1, \ldots, d$, it sets outputs $g^{q(i)}$ by interpolation in the exponent using $\{\theta_{x_i}\}$ and B. For $i = 0, 1, \ldots, d$, it sets $g^{h(i)} = g_2^{u(i)} g^{f(i)}$. Observe that these values are (jointly) distributed identically to their distribution in the actual scheme. Note that implicitly we have $T(x) = g_2^{x^d + u(x)} g^{f(x)}$.

Phase 1 $\mathcal{A}$ adaptively makes requests for several access structures such that γ passes through none of them. Suppose $\mathcal{A}$ makes a request for the secret key for an access structure $\tilde{\mathbb{A}}$ where $\tilde{\mathbb{A}}(\gamma) = 0$. Note that by assumption, $\tilde{\mathbb{A}}$ is given as $NM(\mathbb{A})$ for some monotonic access structure $\mathbb{A}$, over a set $\mathcal{P}$ of parties (whose names will be attributes), associated with a linear secret-sharing scheme Π.

Let M be the share-generating matrix for Π: Recall, M is a matrix over $\mathbb{Z}_p$ with ℓ rows and $n+1$ columns. For all $i = 1, \ldots, \ell$, the i'th row of M is labeled with a party named $\breve{x}_i \in \mathcal{P}$, where

x_i is the attribute underlying $\breve{x}_i$. Note that $\breve{x}_i$ can be primed (negated) or unprimed (non-negated). When we consider the column vector $v = (s, r_1, r_2, \ldots, r_n)$, where s is the secret to be shared, and $r_1, \ldots, r_n \in \mathbb{Z}_p$ are randomly chosen, then Mv is the vector of ℓ shares of the secret s according to Π.

We make use of the following well-known observation about linear secret-sharing schemes (see, e.g. [2][4]): If $S \subset \mathcal{P}$ is a set of parties, then these parties can reconstruct the secret iff the column vector $(1, 0, 0, \ldots, 0)$ is in the span of the rows of M_S, where M_S is the submatrix of M containing only those rows that are labeled by a party in S. Note that since $\tilde{\mathbb{A}}(\gamma) = 0$, we know that $\mathbb{A}(\gamma') = 0$, where $\gamma' = N(\gamma)$. Thus, we know that $(1, 0, \ldots, 0)$ is linearly independent of the rows of $M_{\gamma'}$.

During key generation, a secret sharing of the secret $\alpha = a$ is supposed to be selected. In this simulation, however, we will choose this sharing (implicitly) in a slightly different manner, as we describe now: First, we pick a uniformly random vector $v = (v_1, \ldots, v_{n+1}) \in \mathbb{Z}_p^{n+1}$. Now, we make use of the following simple proposition [1, 20] from linear algebra:

Proposition 1 *A vector π is linearly independent of a set of vectors represented by a matrix N if and only if there exists a vector w such that $Nw = \vec{0}$ while $\pi \cdot w = 1$.*

Since $(1, 0, \ldots, 0)$ is independent of $M_{\gamma'}$, there exists a vector $w = (w_1, \ldots, w_{n+1})$ such that $M_{\gamma'} w = \vec{0}$ and $(1, 0, \ldots, 0) \cdot w = w_1 = 1$. Such a vector can be efficiently computed [1, 20]. Now we define the vector $u = v + (a - v_1)w$. (Note that u is distributed uniformly subject to the constraint that $u_1 = a$.) We will implicitly use the shares $\vec{\lambda} = Mu$. This has the property that for any λ_i such that $\breve{x}_i \in \gamma'$, we have that $\lambda_i = M_i u = M_i v$ has no dependence on a.

Now that we have established how to distribute shares to "parties", which map to negated or non negated attributes, we need to show how to generate the key material.

We first describe how to generate decryption key material corresponding to negated parties $\breve{x}_i = x_i'$. Note that by definition, $\breve{x}_i \in \gamma'$ if and only if $x_i \notin \gamma$.

- If $x_i \in \gamma$, then since $\breve{x}_i \notin \gamma'$, we have that λ_i may depend linearly on a. However, by the simulator's choices at setup, recall that $q(x_i) = \theta_{x_i}$. The simulator now chooses $r_i' \in \mathbb{Z}_p$ at random, and implicitly sets $r_i = -\lambda_i + r_i'$. Thus, it outputs the following:

  $$D_i = (D_i^{(3)} = g_2^{r_i'}, D_i^{(4)} = g^{\theta_{x_i} \cdot (-\lambda_i + r_i')}, D_i^{(5)} = g^{-\lambda_i + r_i'})$$

  Note that the simulator can compute the latter two of these elements using A.

- If $x_i \notin \gamma$, then since $\breve{x}_i \in \gamma'$, we have that λ_i is independent of any secrets and is completely known to the simulator. In this case, the simulator chooses $r_i \in \mathbb{Z}_p$ at random, and outputs the following:

  $$D_i = (D_i^{(3)} = g_2^{\lambda_i + r_i}, D_i^{(4)} = V(x_i)^{r_i}, D_i^{(5)} = g^{r_i})$$

  Note that the simulator can compute the second element using B; indeed $V()$ is publicly computable given the public parameters already produced by the simulation.

We now describe how to give key material corresponding to non negated parties $\breve{x}_i = x_i$. The simulated key construction techniques for non negated parties is similar to previous work [16, 21].

[4]Here, we are essentially exploiting the equivalence between linear secret-sharing schemes and monotone span programs, as proven in [2]. The proof in [2] is for a slightly different formulation, but applies here as well.

- If $x_i \in \gamma$, then since λ_i has no dependence on any unknown secrets, we simply choose $r_i \in \mathbb{Z}_p$, and output $D_i = (D_i^{(1)} = g_2^{\lambda_i} \cdot T(x_i)^{r_i}, D_i^{(2)} = g^{r_i})$.
- If $x_i \notin \gamma$, then we work as follows: Let $g_3 = g^{\lambda_i}$. Note that the simulator can compute g_3 using A and g. Choose $r_i' \in \mathbb{Z}_p$ at random, and output the components of D_i as follows:

$$\begin{aligned} D_i^{(1)} &= g_3^{\frac{-f(x_i)}{x_i^d + u(x_i)}} (g_2^{x_i^d + u(x_i)} g^{f(x_i)})^{r_i'} \\ D_i^{(2)} &= g_3^{\frac{-1}{x_i^d + u(x_i)}} g^{r_i'} \end{aligned}$$

The proof of the following claim can be found in Appendix B.

Claim 1 *The simulation above produces valid decryption keys, that are furthermore distributed identically to the decryption keys that would have been produced by the ABE scheme for the same public parameters.*

Challenge The adversary $\mathcal{A}$, will submit two challenge messages M_0 and M_1 to the simulator. The simulator flips a fair binary coin ν, and returns an encryption of M_ν. The ciphertext is output as

$$E = \left(\gamma, E^{(1)} = M_\nu Z, E^{(2)} = C, \{E_x^{(3)} = C^{f(x)}\}_{x \in \gamma}, \{E_x^{(4)} = C^{\theta_x}\}_{x \in \gamma}\right)$$

If $\mu = 0$ then $Z = e(g,g)^{abc}$. Then by inspection, the ciphertext is a valid ciphertext for the message M_ν under the set γ.

Otherwise, if $\mu = 1$, then $Z = e(g,g)^z$. We then have $E^{(1)} = M_\nu e(g,g)^z$. Since z is random, $E^{(1)}$ will be a random element of $\mathbb{G}_T$ from the adversary's viewpoint and the message contains no information about M_ν.

Phase 2 The simulator acts exactly as it did in Phase 1.

Guess $\mathcal{A}$ will submit a guess ν' of ν. If $\nu' = \nu$ the simulator will output $\mu' = 0$ to indicate that it was given a valid BDH-tuple; otherwise, it will output $\mu' = 1$ to indicate it was given a random 4-tuple.

As shown above, the simulator's generation of public parameters and private keys is identical to that of the actual scheme.

In the case where $\mu = 1$ the adversary gains no information about ν. Therefore, we have $\Pr[\nu \neq \nu' | \mu = 1] = \frac{1}{2}$. Since the simulator guesses $\mu' = 1$ when $\nu \neq \nu'$, we have $\Pr[\mu' = \mu | \mu = 1] = \frac{1}{2}$.

If $\mu = 0$ then the adversary sees an encryption of M_ν. The adversary's advantage in this situation is ϵ by assumption. Therefore, we have $\Pr[\nu = \nu' | \mu = 0] = \frac{1}{2} + \epsilon$. Since the simulator guesses $\mu' = 0$ when $\nu = \nu'$, we have $\Pr[\mu' = \mu | \mu = 0] = \frac{1}{2} + \epsilon$.

The overall advantage of the simulator in the Decisional BDH game is $\frac{1}{2}\Pr[\mu' = \mu | \mu = 0] + \frac{1}{2}\Pr[\mu' = \mu | \mu = 1] - \frac{1}{2} = \frac{1}{2}(\frac{1}{2} + \epsilon) + \frac{1}{2}\frac{1}{2} - \frac{1}{2} = \frac{1}{2}\epsilon$. □

5 Conclusions and Future Directions

We presented the first Attribute-Based Encryption system that supports the expression of non-monotone formulas in key policies. We achieved this through a novel application of revocation

methods into existing ABE schemes. In addition, the performance of our scheme compares very favorably to that of existing, less-expressive ABE systems.

An important goal in ABE systems is to create even more expressive systems. Our work took a significant step forward by allowing key policies that can express any access formula. Eventually, we would like to have systems that can express any access circuit.

References

[1] H. Anton and C. Rorres. *Elementary Linear Algebra, 9th Edition*. 2005.

[2] A. Beimel. *Secure Schemes for Secret Sharing and Key Distribution*. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel, 1996.

[3] J. Benaloh and J. Leichter. Generalized Secret Sharing and Monotone Functions. In *Advances in Cryptology – CRYPTO*, volume 403 of *LNCS*, pages 27–36. Springer, 1988.

[4] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *Proceedings of the IEEE Symposium on Security and Privacy (To Appear)*, 2007.

[5] G. R. Blakley. Safeguarding cryptographic keys. In *National Computer Conference*, pages 313–317. American Federation of Information Processing Societies Proceedings, 1979.

[6] D. Boneh and X. Boyen. Efficient Selective-ID Secure Identity Based Encryption Without Random Oracles. In *Advances in Cryptology – Eurocrypt*, volume 3027 of *LNCS*, pages 223–238. Springer, 2004.

[7] D. Boneh and M. Franklin. Identity Based Encryption from the Weil Pairing. In *Advances in Cryptology – CRYPTO*, volume 2139 of *LNCS*, pages 213–229. Springer, 2001.

[8] D. Boneh, C. Gentry, and B. Waters. Collusion Resistant Broadcast Encryption with Short Ciphertexts and Private Keys. In *Advances in Cryptology – CRYPTO*, volume 3621 of *LNCS*, pages 258–275. Springer, 2005.

[9] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security*, pages 146–157, 2004.

[10] E. F. Brickell. Some ideal secret sharing schemes. *Journal of Combinatorial Mathematics and Combinatorial Computing*, 6:105–113, 1989.

[11] R. Canetti, S. Halevi, and J. Katz. A Forward-Secure Public-Key Encryption Scheme. In *Advances in Cryptology – Eurocrypt*, volume 2656 of *LNCS*. Springer, 2003.

[12] R. Canetti, S. Halevi, and J. Katz. Chosen Ciphertext Security from Identity Based Encryption. In *Advances in Cryptology – Eurocrypt*, volume 3027 of *LNCS*, pages 207–222. Springer, 2004.

[13] Melissa Chase. Multi-authority attribute-based encryption. In *The Fourth Theory of Cryptography Conference (TCC 2007)*, 2007.

[14] L. Cheung and C. Newport. Provably Secure Ciphertext Policy ABE. In *ACM conference on Computer and Communications Security (ACM CCS)*, 2007.

[15] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.

[16] 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 (ACM CCS)*, 2006.

[17] M. Ito, A. Saito, and T. Nishizeki. Secret Sharing Scheme Realizing General Access Structure. In *IEEE Globecom*. IEEE, 1987.

[18] Moni Naor and Benny Pinkas. Efficient trace and revoke schemes. In *Financial Cryptography*, pages 1–20, 2000.

[19] M. Pirretti, P. Traynor, P. McDaniel, and B. Waters. Secure Atrribute-Based Systems. In *ACM conference on Computer and Communications Security (ACM CCS)*, 2006.

[20] V.V. Prasolov. *Problems and Theorems in Linear Algebra*. American Mathematical Society, 1994.

[21] A. Sahai and B. Waters. Fuzzy Identity Based Encryption. In *Advances in Cryptology – Eurocrypt*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[22] A. Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[23] A. Shamir. Identity Based Cryptosystems and Signature Schemes. In *Advances in Cryptology – CRYPTO*, volume 196 of *LNCS*, pages 37–53. Springer, 1984.

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

A Realizing Any Access Formula

We show how our main construction above can be used to realize any access formula. Any formula can be represented as an access tree $\mathcal{T}$. Each interior node, y, in the tree will be either a threshold gate, with threshold k_y, and num_y children, or a **NOT** of a threshold gate. We also assume that the children of an interior node are ordered; we let $\text{parent}(y)$ denote the parent of node y and we let $\text{index}(y)$ denote which child node of $\text{parent}(y)$ node y is. In addition, each leaf will be either an attribute or the **NOT** of an attribute. We note that using threshold gates captures the case of **AND** and **OR** gates. We let $\text{attr}(y)$ denote a leaf node's attribute.

We first observe that by applying DeMorgan's law we can transform a tree $\mathcal{T}'$ into a tree $\mathcal{T}$ so that $\mathcal{T}$ represents the same access scheme as $\mathcal{T}'$, but has only **NOTs** at the leaves, where the attributes are. The observation follows from the fact that a negative k of n threshold gate is equivalent to a $k+1$ of n threshold gate, where all the children are negated. We can transform the tree $\mathcal{T}'$ by doing a pre-order traversal and applying this transformation to all interior nodes that are negated. The result is a tree $\mathcal{T}$ that only has negated leaves. For the rest of this discussion

we assume that such a transformation has been applied and that only the leaves of $\mathcal{T}$ are negated, interior nodes will consist only of positive threshold gates.

Now we need to show how to assign shares $\lambda_y \in \mathbb{Z}_p$ to each leaf node in a (transformed) access tree $\mathcal{T}$ for key generation and which $\omega_y \in \mathbb{Z}_p$ values to use for a particular decryption. We essentially assign each interior node a value on a random polynomial and recurse. We begin by assigning λ_y values to all nodes in the system. First, we assign the root of the tree the value $\lambda_{\text{root}} = \alpha$. Then, we repeat the following process until all nodes have been assigned. Pick an arbitrary interior node y that has λ_y defined, but where the children of y are undefined. Then pick a random polynomial $q_y(x)$ over $\mathbb{Z}_p$ of degree $k_x - 1$ with the restriction that $q_y(0) = \lambda_y$. For each child node of z assign $\lambda_z = q_y(\text{index}_z)$. After this is done the λ_y values for interior nodes can be discarded. For Each leaf node y, λ_y is a share in the scheme for attribute $\text{attr}(x)$ where the attribute is primed if the leaf node is negated.

Finally, suppose there is an encryption to a set S of attributes and further suppose that there is a satisfying assignment to the access tree $\mathcal{T}$ of a key. Then the ω_i values can be derived by recursively computing the Lagrangian coefficients of each satisfied node in the tree. We refer the reader to the work of Goyal et al. [16] and Bethencourt, Sahai, and Waters [4] for details on this and efficiency optimizations.

B Correctness of Simulation

Here we prove Claim 1.
PROOF:

We will establish this claim by a case analysis. For key material corresponding to negated parties $\breve{x}_i$:

- If $x_i \in \gamma$, then let $r_i = -\lambda_i + r'_i$. Note that r_i is distributed uniformly over $\mathbb{Z}_p$ and is independent of all other variables except r'_i. Then observe that $D_i^{(3)} = g_2^{r'_i} = g_2^{\lambda_i + r_i}$. Also, $D_i^{(4)} = g^{\theta_{x_i} \cdot (-\lambda_i + r'_i)} = V(x_i)^{r_i}$. And finally, $D_i^{(5)} = g^{-\lambda_i + r'_i} = g^{r_i}$. Thus, this key material is valid and distributed correctly.

- If $x_i \notin \gamma$, then the simulation produces key material using the same procedure as the ABE scheme.

For key material corresponding to non negated parties $\breve{x}_i$:

- If $x_i \in \gamma$, then the simulation produces key material using the same procedure as the ABE scheme.

- If $x_i \notin \gamma$, then to see why the simulated key material is good, note that by our construction of $u(x)$, the value $x_i^d + u(x_i)$ will be non-zero for all $x_i \notin \gamma$. Now let $r_i = r'_i - \frac{\lambda_i}{x_i^n + u(x_i)}$. Note that r_i is distributed uniformly over $\mathbb{Z}_p$ and is independent of all other variables except r'_i.

Then,

$$\begin{aligned} D_i^{(1)} &= g_3^{\frac{-f(x_i)}{x_i^d+u(x_i)}} (g_2^{x_i^d+u(x_i)} g^{f(x_i)})^{r'_i} \\ &= g^{\frac{-\lambda_i f(x_i)}{x_i^d+u(x_i)}} (g_2^{x_i^d+u(x_i)} g^{f(x_i)})^{r'_i} \\ &= g_2^{\lambda_i} (g_2^{x_i^d+u(x_i)} g^{f(x_i)})^{\frac{-\lambda_i}{x_i^d+u(x_i)}} (g_2^{x_i^d+u(x_i)} g^{f(x_i)})^{r'_i} \\ &= g_2^{\lambda_i} (g_2^{x_i^d+u(x_i)} g^{f(x_i)})^{r'_i - \frac{\lambda_i}{x_i^d+u(x_i)}} \\ &= g_2^{\lambda_i} T(x_i)^{r_i} \end{aligned}$$

and

$$D_i^{(2)} = g_3^{\frac{-1}{x_i^d+u(x_i)}} g^{r'_i} = g^{r'_i - \frac{\lambda_i}{x_i^d+u(x_i)}} = g^{r_i}$$

□