Multi-Authority Attribute Based Encryption

Melissa Chase

Computer Science Department
Brown University
Providence, RI 02912
mchase@cs.brown.edu

Abstract. In an identity based encryption scheme, each user is identified by a unique identity string. An attribute based encryption scheme (ABE), in contrast, is a scheme in which each user is identified by a set of attributes, and some function of those attributes is used to determine decryption ability for each ciphertext. Sahai and Waters introduced a single authority attribute encryption scheme and left open the question of whether a scheme could be constructed in which multiple authorities were allowed to distribute attributes [SW05]. We answer this question in the affirmative.

Our scheme allows any polynomial number of independent authorities to monitor attributes and distribute secret keys. An encryptor can choose, for each authority, a number d_k and a set of attributes; he can then encrypt a message such that a user can only decrypt if he has at least d_k of the given attributes from each authority k. Our scheme can tolerate an arbitrary number of corrupt authoritites.

We also show how to apply our techniques to achieve a multiauthority version of the large universe fine grained access control ABE presented by Gopal et al. [GPSW06].

1 Introduction

Identity based encryption(IBE), introduced by Shamir [Sha85], is a variant of encryption which allows users to use any string as their public key (for example, an email address). This means that the sender can send messages knowing only the recipient's identity (or email address), thus eliminating the need for a separate infrastructure to distribute public keys. The first IBE systems were given by Boneh and Franklin [BF01] and Cocks [Coc01], and IBE has received a lot of attention in the literature since then [CHK03,BB04,Wat05].

However, this scenario may not be entirely realistic, since we don't necessarily have a unique string identifier for each person. Instead, we often identify people by their attributes. We might want to send a message to the secretary in accounting in charge of travel reimbursements, or send a question to a nurse in a particular hospital who is knowledgeable about prescriptions, or announce a party to anyone living in town who is either a student or between the ages of 18 and 25. Thus, Sahai and Waters gave a fuzzy IBE scheme which could be used for attribute based encryption. In this model, a recipient is defined not by a single string, but by a set of attributes [SW05]. Sahai and Waters describe a scheme (from here on referred to as SW) in which a sender can encrypt a message specifying an attribute set and a number d so that only a recipient who has at least d of the given attributes can decrypt the message. For example, a sender could encrypt a message to be decryptable by anyone who has 2 out of 3 of: a Rhode Island driver's license, Rhode Island voter registration, or a student ID from Brown University. Thus, their scheme allows the sender to encrypt a message for more than one recipient, and to specify who should be able to decrypt, using attributes alone.

There is, however, one major limitation to the SW scheme. In their scheme, as in every IBE scheme, the user must go to a trusted party and prove his identity in order to obtain a secret key which will allow him to decrypt messages. In this case, each user must go to the trusted server, prove that he has a certain set of attributes, and then receive secret keys corresponding to each of those attributes. However, this means we must have one trusted server who monitors all attributes – who keeps records of driver's licenses, voter registration, and college enrollment. In reality, we have 3 different entities responsible for maintaining this information (the RI DMV, the RI Board of Elections, and the University office), so we would want to be able to entrust each of these to a different (and perhaps not entirely trusted) server. Thus, Sahai and Waters presented the following challenge: Is it possible to construct an attribute based encryption scheme in which many different authorities operate simultaneously, each handing out secret keys for a different set of attributes?

Our Results We resolve this problem in the affirmative. We give an efficient scheme for multiauthority attribute based encryption. We allow the sender to specify for each authority k a set of attributes monitored by that authority and a number d_k so that the message can be decrypted only by a user who has at least d_k of the given attributes from every authority. We allow any number of attribute authorities to be corrupted, and guarantee the security of encryption as long as the required attributes cannot be obtained exclusively from those authorities and the trusted authority remains honest.

We also provide several extensions to our basic multiauthority scheme. We describe techniques to allow the encryptor to determine for each ciphertext how many attributes to require from each authority. We also describe a variant of our scheme in which the encryptor can specify a number D such that a user can decrypt if he has sufficient numbers of the given attributes from at least D authorities. It is this variant that would be used to implement the RI example above. In this example, we have 3 authorities, and the ciphertext will include 1 attribute from each. However, we only want to require that a user must have satisfactory attributes from 2 out of the 3 authorities in order to decrypt.

Challenges and Techniques The most challenging aspect of a single authority ABE scheme is preventing collusion. Recall the above example. Now suppose Alice has a RI driver's license and Bob is a Brown University student. Together they have two out of three of the required attributes, but they should not be able to combine their keys and decrypt the ciphertext. In SW each user's keys are generated using different random sharings of a secret, so keys generated for different users cannot be combined.

It might seem that a multiauthority ABE scheme could be formed simply by letting each authority run its own copy of SW and then combining the results. However, here we once again run into the problem of collusion. The SW techniques will prevent collusion within authorities, so different keys obtained from any one authority cannot be combined. However, suppose we have a ciphertext which requires attributes from authority 1 and authority 2. If Alice has all the appropriate attributes from authority 1 and Bob has all the appropriate attributes from authority 2, they still should not be able to combine their keys and decrypt. Note that the SW techniques cannot be directly applied here: in SW, a single authority sees all the attributes requested by a user and gives a secret key, so it can easily rerandomize the secret sharing appropriately. In the multiauthority case, we would again like to split up a secret in a different way for each user, this time dividing it between multiple authorities. However, now we need to do this *without any communication* between the authorities.

We use two main techniques: The first is to require that every user have some kind of a global identifier (GID). We require only two properties from this: (1) no user can claim another user's identifier, and (2) all authorities can verify a user's identifier. Thus, the GID could be a name or SSN or any other identifying string for which a user has provable credentials, and it seems likely that such information would be present when users' attributes are verified. To see why this is necessary, consider the following two scenarios: In the first Bob requests keys for attribute set $\mathcal{A}_1$ from authority 1 and Alice requests keys for attribute set $\mathcal{A}_2$ from authority 2. In the second Bob requests attribute set $\mathcal{A}_1$ from authority 1 and attribute set $\mathcal{A}_2$ from authority 2. If the authorities do not communicate, and Alice and Bob are identified by nothing beyond their attributes, then in the authorities' view these scenarios must be identical. The global identifier allows the authorities to distinguish these two scenarios in order to prevent collusion.

At the same time we still want a user's ability to decrypt to depend only on his attributes (this is what distinguishes ABE from traditional IBE schemes). Thus, we use our second tool: the central authority. Each user will send his GID to the central authority and receive a corresponding key. Note that the authority will not get any information about the users' attributes; it's purpose is simply to give a setup key for the user's GID. We will also require that this authority be trusted: it will hold the master secret for the system, so it will be able to decrypt any message. Note that the presence of a trusted party is a fairly standard requirement: in an IBE scheme, the single authority must obviously be trusted, and even when this is extended to a hierarchical IBE (HIBE) scheme, in which many of the lower level authorities can be corrupted, one must require that the root authority be honest.

Each authority has a pseudorandom function (PRF) which it will use to randomize the secret keys it gives out. A PRF guarantees that, on the one hand, the secret keys for each user are derived deterministically, but, at the same time, that they will appear completely random. When a user requests a secret key, the authority will compute the PRF on the user's GID and then use the result as the secret in SW key generation. A user with sufficient attributes can then use his secret keys to reconstruct this secret for each authority. However, since the outputs of the PRFs will be different for each user, each user will reconstruct a different set of secrets. Thus, we need the central authority, who will know all of the other authorities' PRFs. For each user, it will compute the extra value which, when combined with the secrets the user has reconstructed, will result in a user-independent system decryption value which allows the user to decrypt.

Essentially this lets us break up a constant secret across multiple authorities based on the user's GID, in such a way that each authority can compute his part independently given only the GID. The use of PRFs mean that each user's secret keys are independent of any other user's keys, and collusion is impossible. Then the central authority gives the added keys necessary to ensure that if we compute each PRF on the same GID, we can always combine the results to obtain a fixed value, and thus it allows us to give ciphertexts that are independent of the GID. For a full description and explanation of this technique, see Section 4.

Other ABE Schemes As mentioned above, our scheme is an extension of the basic Fuzzy IBE scheme of Sahai and Waters. Their scheme requires that a user have t out of n of the desired attributes in order to decrypt. More recently, Gopal et al. presented a scheme for fine grained access control in the Key-Policy model [GPSW06]. In this model, when a user requests a private key, the authority determines what combinations of attributes must be present in order for this user to decrypt and gives the user the corresponding private key.

The main difference is that in this system, the private key no longer corresponds to a simple set of attributes that the user possesses. Instead, each private key represents a formula describing which sets of attributes must appear on the ciphertext in order for this user to decrypt. Ciphertexts are encrypted with a simple set of attributes.

Our techniques can also be applied to this more complex scheme to form a system in which, in order to decrypt a ciphertext encrypted with a set of attributes for each authority, a user must have received from each authority a policy which allows decryption for that set of attributes. Gopal et al. also present a large universe access structure scheme (an extension of the large universe scheme in SW). This also can be combined with our techniques to create a multiauthority large universe access structure scheme. For details, see Section 5.

2 Preliminaries

In our ABE scheme, we assume that the universe of attributes can be partitioned into K disjoint sets. Each will be monitored by a different authority. As mentioned above, we also have one trusted central authority who does not monitor any attributes.

Note: In the following we use $\mathcal{A}_u$ to denote the attribute set of user u and $\mathcal{A}_C$ to denote the attribute set of a ciphertext. $\mathcal{A}_u^k$ and $\mathcal{A}_C^k$ are the attributes handled by authority k in the attribute sets of the user and the ciphertext respectively.

A MultiAuthority ABE system is composed of K attribute authorities and one central authority. Each attribute authority is also assigned a value d_k. The system uses the following algorithms:

Setup : A randomized algorithm which must be run by some trusted party (e.g. central authority). Takes as input the security parameter. Outputs a public key, secret key pair for each of the attribute authorities, and also outputs a system public key and master secret key which will be used by the central authority.

Attribute Key Generation : A randomized algorithm run by an attribute authority. Takes as input the authority's secret key, the authority's value d_k, a user's GID, and a set of attributes in the authority's domain $\mathcal{A}_C^k$. (We will assume that the user's claim of these attributes has been verified before this algorithm is run). Output secret key for the user.

Central Key Generation : A randomized algorithm run by the central authority. Takes as input the master secret key and a user's GID and outputs secret key for the user.

Encryption : A randomized algorithm run by a sender. Takes as input a set of attributes for each authority, a message, and the system public key. Outputs the ciphertext.

Decryption : A deterministic algorithm run by a user. Takes as input a ciphertext, which was encrypted under attribute set $\mathcal{A}_C$ and decryption keys for an attribute set $\mathcal{A}_u$. Outputs a message m if $|\mathcal{A}_C^k \cap \mathcal{A}_u^k| > d_k$ for all authorities k.

Note that the number of authorities in the system need not be fixed permanently: it is possible to allow the central authority to add additional attribute authorities to the system at any point. For a discussion of this and other possible extensions to this scheme, see Section 6.

As in [SW05], our scheme is proved secure in the selective ID (sid) model, in which the adversary must provide the identity he wishes to attack(the challenge identity) before receiving the parameters of the system.

Let κ be the security parameter. We require that the number of authorities, K, and the number of attributes monitored by each authority, n_k, be upper bounded by a number n which is polynomial in κ.

Consider the following game:

Setup

- The adversary sends a list of attribute sets $\mathcal{A}_C = \mathcal{A}_C^1 \dots \mathcal{A}_C^K$, one for each authority. He must also provide a list of corrupted authorities which cannot include the central authority.
- The challenger generates parameters for the system and sends them to the adversary. This means the system public key, public keys for all honest authorities, and secret keys for all corrupt authorities.

Secret Key Queries

– The adversary can make as many secret key queries as he wants to the attribute authorities or to the central authority. The only requirements are (1) that for each GID, there must be at least one honest authority k from which the adversary requests fewer than d_k of the attributes given in $\mathcal{A}_C^k$, i.e. the adversary never requests enough attributes to decrypt the challenge ciphertext, and (2) that the adversary never queries the same authority twice with the same GID (see below for discussion).

Challenge

– The adversary sends two messages M_0 and M_1.
– The challenger chooses a bit b, computes the encryption of M_b for attribute set $\mathcal{A}_C$, and sends this encryption to the adversary.

More Secret Key Queries

– The adversary may make more secret key queries subject to the requirements described above.

Guess

– The adversary outputs a guess b' that message $M_{b'}$ has been encrypted.

The adversary is said to succeed if he can correctly identify the encrypted message, i.e. if $b = b'$.

Definition 1. *A multiauthority attribute scheme is sid-secure if there exists a negligible furnction ν such that, in the above game any adversary will succeed with probability at most $1/2 + \nu(\kappa)$.*

Note that our scheme is designed only for static attributes: each authority will only issue one set of secret keys for each GID. If a user later returns with the same GID but a different set of attributes, the authority will refuse the request. However this can easily be converted into a scheme which allows changes in attributes by allowing each user a range of GID instead of just one. Then when a user needs to change his attribute set, he simply moves on to a new GID and requests secret keys from each authority with the new attribute set and new GID (he must however obtain new secret keys from *all* authorities).

We have found no obvious attack when this requirement is removed; it seems to be an artifact of our proof techniques. Essentially, in our reduction, when we give out secret keys from a certain authority, we need to know whether the adversary will request sufficient attributes from that authority to decrypt the challenge ciphertext. Our reduction responses will depend crucially on that factor. (For more details, see Section 4.)

Definition 2 (Bilinear Diffie-Hellman(BDH) Assumption). *Let G be a group of prime order q and generator g where $|q|$ is proportional to the security parameter κ. There exists a negligible function ν such that for all adversaries A, given G, q, g, g^a, g^b, g^c and bilinear map e for randomly chosen $a, b, c \leftarrow Z_q$, A can distinguish $e(g,g)^{abc}$ from $e(g,g)^R$ for random $R \leftarrow Z_q$ with probability at most $\nu(\kappa)$.*

3 Single Authority ABE

We will begin by demonstrating how the simplest attribute based encryption (ABE) scheme, Sahai and Waters' "Fuzzy IBE" or "Threshold ABE" scheme, can be converted into a multiauthority scheme. Then in Section 5 we will describe a multi authority scheme for more complex ABE.

We will now explain some of the intuition behind our scheme. We will incrementally build up to the full robust multiauthority construction. We begin by examining the single authority case, which was considered by Sahai and Waters [SW05]. In their scheme, there is one authority giving out secret keys for all of the attributes. Each encryptor then specifies a list of attributes such that any user with at least d of those attributes will be able to decrypt. They show that the scheme they present is sid secure.

We review these results and attempt to explain a possible derivation, building up to the description of the SW scheme through a series of incomplete schemes. We will then show how this can be easily converted into a multiauthority scheme. We hope that once the intuition behind SW is completely clear, the changes necessary to convert this scheme into a multiauthority one will also be fully intuitive.

Step One – Feldman VSS

First, let's consider a very simplified scheme based on the Feldman Verifiable Secret Sharing scheme [Fel87].

Recall that, given d points $p(1), \ldots, p(d)$ on a $d-1$ degree polynomial, we can use Lagrange interpolation to compute $p(i)$ for any i. However, given only $d-1$ points, any other points are information theoretically hidden. According to the Lagrange formula, $p(i)$ can be computed as a linear combination of d known points. Let $\Delta_j(i)$ be the coefficient of $p(j)$ in the computation of $p(i)$. Then $p(i) = \sum_{j \in S} p(j)\Delta_j(i)$ where S is a set of any d known points and $\Delta_j(i) = \prod_{k \in S, j \neq k}(i-k)/(j-k)$. Note that any set of d random numbers defines a valid polynomial, and given these numbers we can find any other point on that polynomial.

Furthermore, if we are instead given $g^{p(1)}, \ldots, g^{p(d)}$, we can similarly compute $g^{p(i)}$ for any i, and the hiding property mentioned above still applies.

This suggests a technique for attribute based encryption: If a user has attribute i, his secret key will include $g^{p(i)}$, for some degree $d-1$ polynomial p. We can encrypt a message m by giving $g^{p(0)}m$. Then any user with at least d attributes can interpolate to obtain the secret $g^{p(0)}$ and thus discover m. However, to any user without d attributes $g^{p(0)}$ is information theoretically hidden and thus finding m will be impossible.

Note that we can easily extend this to prevent collusion: If we give all our users points from the same polynomial, any group with at least d attributes between them would be able to combine their keys to find $p(0)$. However, if we instead give each user u a different polynomial p_u (but still with the same zero point $p_u(0) = p(0)$), then one user's points will give no information on the polynomial held by the other (as long as neither has more than $d-1$ points). To see this, note that, given any $d-1$ points on polynomial p_1 and any $d-1$ points on polynomial p_2, with the requirement that these polynomials must intersect at 0, it is still the case that any value for $y = p_1(0) = p_2(0)$ will define a valid pair of polynomials. Thus, y is information theoretically hidden. Then our first scheme runs a follows:

Init First fix $y \leftarrow Z_q$.
SK for user u**:** Choose a random polynomial p such that $p(0) = y$. SK: $\{D_i = g^{p(i)}\}_{\forall i \in \mathcal{A}_u}$.
Encryption: $E = g^y m$.
Decryption: Use d SK elements D_i to interpolate to obtain $Y = g^{p(0)} = g^y$. Then $m = E/Y$.

Step Two – Specifying attributes

If we take this approach, any user with any d attributes will be able to decrypt. But we want each encryptor to be able to give a specific subset of attributes such that at least d are necessary for decryption.

In order to do this, we need an extra tool: bilinear maps. Recall that for bilinear map e, $g \in G_1$, and $a, b \in Z_q$, $e(g^a, g^b) = e(g, g)^{ab}$.

Now, suppose instead of giving each user $g^{p(i)}$ for each attribute i, we choose a random value t_i and give $g^{p(i)/t_i}$. If the user knew g^{t_i} for at least d of these attributes, he could compute $e(g, g)^{p(i)}$ for each i and then interpolate to find the secret $e(g, g)^{p(0)}$. Then if our encryption includes $e(g, g)^{p(0)}m$, the user would be able to find m. Thus, the encryptor can specify which attributes are relevant by providing g^{t_i} for each attribute i in the desired set.

Suppose we only give one secret key to one user u. Now, for $i \in \mathcal{A}_u, i \notin \mathcal{A}_C$ the t_i values appear only once: when we give $g^{p(i)/t_i}$. Thus, since t_i was chosen at random, $p(i)$ is still information theoretically hidden. The only attributes i for which user u has any information on $p(i)$ are those where $i \in \mathcal{A}_u \cap \mathcal{A}_C$. As long as there are less than d of these, $p(0)$ (and thus $e(g, g)^{p(0)}$) must be information theoretically hidden.

If we allow multiple secret key queries, this is no longer the case. However given the BDH Assumption, we can show that $e(g, g)^{p(0)}$ is still hidden as long as no user has more than $d-1$ attributes in common with the ciphertext. This will be a special case of the proof in the next step. The resulting scheme is as follows:

Init First fix $y, t_1, \ldots, t_n \leftarrow Z_q$. Let $Y = e(g, g)^y$.
SK for user u**:** Choose a random polynomial p such that $p(0) = y$. SK:$\{D_i = g^{p(i)/t_i}\}_{\forall i \in \mathcal{A}_u}$.
Encryption for attribute set $\mathcal{A}_C$**:** $E = Ym$ and $\{E_i = g^{t_i}\}_{\forall i \in \mathcal{A}_C}$.
Decryption: For d attributes $i \in \mathcal{A}_C \cap \mathcal{A}_u$, compute $e(E_i, D_i) = e(g, g)^{p(i)}$. Interpolate to find $Y = e(g, g)^{p(0)} = e(g, g)^y$. Then $m = E/Y$.

Step Three: Multiple Encryptions

There are several obvious problems with this scheme. First, we would like to be able to encrypt multiple times without the decryptor needing to get a new secret key each time. But, once a user has obtained $e(g,g)^{p(0)}$, he can decrypt any subsequent encryptions whether or not he has the appropriate attribute set.

What if instead of giving $e(g,g)^{p(0)}m$ in our encryption, we give $e(g,g)^{p(0)s}m$, where s is a different random number for each encryption? If we also give $\{E_i = g^{t_i s}\}_{\forall i \in \mathcal{A}_C}$ instead of $\{E_i = g^{t_i}\}$, the above process will allow a user with the appropriate attributes to find $e(g,g)^{p(0)s}$, and thus to decrypt m. Note that now our secret $e(g,g)^{p(0)s}$ is different for each ciphertext.

This also solves another of our problems: it gives us a way to compute the ciphertext. Before, computing the ciphertext required knowing the g^{t_i} values and $e(g,g)^{p(0)}$, which was in turn enough to decrypt any message. Now, decrypting a message requires knowing $g^{t_i s}$ for the appropriate attributes i and the appropriate s. Thus, we can now publish $T_i = g^{t_i}$ and $Y = e(g,g)^{p(0)}$ as the public key, and each encryptor can choose his own random s to compute $\{E_i = T_i^s\}_{\forall i \in \mathcal{A}_C}$, and $Y^s m$.

Furthermore, we can show that $e(g,g)^{p(0)s}$ is still hidden, even when the user knows T_i for all i, a set $\{T_i^s\}_{\forall i \in \mathcal{A}_C}$, and adaptively chose secret keys for user u with $|\mathcal{A}_u \cap \mathcal{A}_C|$ at most $d-1$. Thus, we can show this scheme is *sid* secure based on the BDH Assumption. We have now reconstructed the SW scheme:

Init First fix $y, t_1, \ldots, t_n \leftarrow Z_q$.
PK for system $T_1 = g^{t_1} \ldots T_n = g^{t_n}, Y = e(g,g)^y$. $PK = \{T_i\}_{1 \le i \le n}, Y$
SK for user u**:** Choose a random polynomial p such that $p(0) = y$. SK: $\{D_i = g^{p(i)/t_i}\}_{\forall i \in \mathcal{A}_u}$.
Encryption for attribute set $\mathcal{A}_C$**:** $E = Y^s = e(g,g)^{ys}m$ and $\{E_i = g^{t_i s}\}_{\forall i \in \mathcal{A}_C}$.
Decryption: For d attributes $i \in \mathcal{A}_C \cap \mathcal{A}_u$, compute $e(E_i, D_i) = e(g,g)^{p(i)s}$. Interpolate to find $Y^s = e(g,g)^{p(0)s} = e(g,g)^{ys}$. Then $m = E/Y^s$.

4 Multiple Authorities

Now we consider the multiauthority case. Once again, we will build up our construction by first considering a series of incomplete schemes.

As a first thought, we might simply have many copies of SW, one for each authority. We want to require that a user be able to decrypt a ciphertext only if he has at least d of the specified attributes from each of the K authorities. Recall that the SW scheme centers around finding enough polynomial shares $e(g,g)^{p(i)s}$ to reconstruct the secret $e(g,g)^{p(0)s} = e(g,g)^{ys}$ which has been used to blind the message. (Recall that the encryption includes $E = e(g,g)^{ys}m$). Now, if we want each authority to give out its own polynomials, one simple solution might be to do an additive secret sharing to form the SW secrets (i.e. the values y such that every random polynomial p is chosen with $p(0) = y$). Thus, we pick a random value for the master secret y_0 and for each authority $k = 1 \ldots K$, y_k is a share of y_0 so $\sum y_k = y_0$. We can output $e(g,g)^{y_0}$ as the entire system's public key. Then to encrypt message m, a user gives $E = e(g,g)^{y_0 s}m$ and $E_{k,i} = T_{k,i}^s$ for all i, k where they wish to allow a decryptor to use attribute

i from authority k. To decypt, the user has to perform SW decryption for each authority and find $Y_k^s = e(g,g)^{y_k s}$, then multiply the results together to get $\prod_{k=1}^{K} Y_k^s = \prod_{k=1}^{K} e(g,g)^{y_k s} = e(g,g)^{s \sum_{k=1}^{K} y_k} = e(g,g)^{y_0 s}$ and thus obtain m. However, if a user does not have enough of the required attributes from one authority $\hat{k}$, then the SW secret for that authority: $Y_{\hat{k}} = e(g,g)^{y_{\hat{k}} s}$ will remain indistinguishable from random and thus so will $e(g,g)^{y_0 s}$ and m. Thus our first attempt Multi Authority Scheme is as follows:

System

Init First fix $y_1 \dots y_k, \{t_{k,i}\}_{i=1\dots n, k=1\dots K} \leftarrow Z_q$. Let $y_0 = \sum_{k=1}^{K} y_k$.

System Public Key $Y_0 = e(g,g)^{y_0}$.

Attribute Authority k

Authority Secret Key The SW secret key: $y_k, t_{k,1} \dots t_{k,n}$.

Authority Public Key $T_{k,i}$ from the SW public key: $T_{k,1} \dots T_{k,n}$ where $T_{k,i} = g^{t_{k,i}}$.

Secret Key for User u **from authority** k Choose random $d-1$ degree polynomial p with $p(0) = y_k$. Secret Key: $\{D_{k,i} = g^{p(i)/t_{k,i}}\}_{i \in \mathcal{A}_u}$.

Encryption for attribute set $\mathcal{A}_C$ Choose random $s \leftarrow Z_q$. Encryption: $E = Y_0^s m, \{E_{k,i} = T_{k,i}^s\}_{i \in \mathcal{A}_C^k, \forall k}$.

Decryption: For each authority k, for d attributes $i \in \mathcal{A}_C^k \cap \mathcal{A}_u$, compute $e(E_{k,i}, D_{k,i}) = e(g,g)^{p(i)s}$. Interpolate to find $Y_k^s = e(g,g)^{p(0)s} = e(g,g)^{y_k s}$. Combine these values to obtain $\prod_{k=1}^{K} Y_k^s = Y_0^s$. Then $m = E/Y_0^s$.

There is a problem with the scheme as described above: Suppose an encryptor encrypts a message to the attribute set $\mathcal{A}_C$ which includes attributes $\mathcal{A}_C^k$ for each authority k. Now suppose we have a set of K users where each user k has attribute set $\mathcal{A}_u = \mathcal{A}_C^k$ from authority k, but no attributes from any other authority. Recall that we want to allow decryption only if the decryptor has enough of the required attributes from *every one* of the authorities. However, if the scheme is as described above, this set of users will be able to collude: Each user k will use his attribute set to find the SW secret for authority k: $Y_k^s = e(g,g)^{y_k s}$. Then the users combine these values to obtain $\prod_{k=1}^{K} Y_k^s = \prod_{k=1}^{K} e(g,g)^{y_k s} = e(g,g)^{y_0 s} = Y_0^s$ and thus m.

Clearly, if there is no way to identify users beyond their attribute sets, then the above collusion is impossible to prevent: to the authorities, k separate users each with attribute set $\mathcal{A}_C^k$ and one user with attribute set $\mathcal{A}_C$ look identical.

We solve this problem by requiring that each user have a unique global identifier (GID), as described in Section 1. A user must present the same GID to each authority in order to receive a coherent set of keys (and presumably prove to each authority that the GID is valid). However, encryption will, as before, only specify a set of attributes of which d will be required to decrypt. Thus, the ability to decrypt is independent of the GID (except in that all secret keys must have been obtained for the same GID).

Now that we can distinguish different users, we need some way to ensure that different users cannot combine their results from different authorities. Suppose we have each authority k choose a different random $y_{k,u}$ value for each user.

Let $y_{u,0}$ be the master secret for user u. If user u finds $e(g,g)^{y_{k,u}s}$ and shares it with user u' in an attempt to collude, it won't give user u' any information on his master secret $e(g,g)^{y_{0,u'}s} = \prod_{k=1}^{K} e(g,g)^{y_{k,u'}s}$ (since $y_{k,u}$ is independent of $y_{k,u'}$), so the above collusion will no longer be possible. Recall that the SW scheme uses different polynomials to split up the secret (y) in a different way for each user, thus preventing collusion. Now we are using a similar technique to prevent collusion across authorities: we choose a new set of $y_{k,u}$ values and divide the secret (y_0) among the authorities in a different way for each user.

Note that we need to include $\prod_{k=1}^{K} e(g,g)^{y_{k,u}} = e(g,g)^{y_{0,u}}$ in the public key so that it can be used to form an encryption ($E = e(g,g)^{y_{0,u}s}m$). Moreover, recall that these ciphertexts must be independent of the identity of the user – we would like the ability to decrypt to depend only on the attributes. This means, in order for the encryption/decryption to work with this new addition, we would need $\prod_{k=1}^{K} e(g,g)^{y_{k,u}} = e(g,g)^{y_{0,u}}$ to be the same for all users.

But if all authorities choose $y_{k,u}$ independently, how can we ensure that $\sum_{k=1}^{K} y_{k,u} = y_0$ for all u? It would seem that we must need some kind of communication between authorities, and our goal of k autonomous authorities is impossible with this approach.

An alternative might be to allow our authorities to share some state. If one of the authorities knew the other authorities' random choices, he could choose his $y_{k,u}$ values to ensure that $\sum_{k=1}^{K} y_{k,u} = y_0$. However, we don't necessarily want to require that any of our attribute authorities be completely trusted, so we may not want them to share this information.

Thus, we add the additional "central" authority (see Section 1), who handles no attributes, but who must be fully trusted. This authority will be allowed to know some of the state of each of the other authorities. In particular, it will know enough of their secret state to reconstruct $y_{k,u}$ for any user u and for all authorities k. It will use this information to provide a secret key which, when combined with a value g^s to be given in the encryption and with the "secrets" $Y_{k,u}^s$ obtained from each of the other authorities, will give user u the "master secret": $Y_0^s = e(g,g)^{y_0 s}$ which can then be used to obtain m. Now we only need to trust one authority and it need not be one of the attribute authorities. [1]

Finally, instead of using truly random values, we have each of our K authorities choose the $y_{k,u}$ values using a pseudorandom function (PRF). Thus, now the central authority has only to store the seeds of all of the PRFs.

Final MultiAuthority Scheme: (changes from previous schemes are underlined.)

System

Init Fix prime order groups G, G_1, bilinear map $e : G \to G_1$, and generator $g \in G$. Choose <u>seeds $s_1, \ldots, s_K$ for for all authorities</u>. Also choose

[1] Note, we could require that one of the attribute authorities be trusted and have it maintain the state information of all the other authorities. However, we chose to separate these functions in order to consider a more general case. The central authority could easily be combined with a trusted attribute authority.

$y_0, \{t_{k,i}\}_{k=1\ldots K, i=1\ldots n} \leftarrow Z_q$.

System Public Key $Y_0 = e(g,g)^{y_0}$.

Attribute Authority k

Authority Secret Key $\underline{s_k}$, $t_{k,1} \ldots t_{k,n}$.

Authority Public Key $T_{k,1} \ldots T_{k,n}$ where $T_{k,i} = g^{t_{k,i}}$.

Secret Key for User u $\underline{\text{Let } y_{k,u} = F_{s_k}(u)}$. Choose random $d-1$ degree polynomial p with $p(0) = y_{k,u}$. Secret Key: $\{D_{k,i} = g^{p(i)/t_{k,i}}\}_{i \in \mathcal{A}_u}$.

Central Authority

Central Authority Secret Key s_k for all authorities k, y_0.

Secret Key for User u Let $y_{k,u} = F_{s_k}(u)$ for all k. Secret Key: $D_{CA} = g^{(y_0 - \sum_{k=0}^{K} y_{k,u})}$.

Encryption for attribute set $\mathcal{A}_C$ Choose random $s \leftarrow Z_q$. $E = Y_0^s m$, $\underline{E_{CA} = g^s}$, $\{E_{k,i} = T_{k,i}^s\}_{i \in \mathcal{A}_C^k, \forall k}$.

Decryption: For each authority k, for d attributes $i \in \mathcal{A}_C^k \cap \mathcal{A}_u$, compute $e(E_{k,i}, D_{k,i}) = e(g,g)^{p(i)s}$. Interpolate to find $Y_{k,u}^s = e(g,g)^{p(0)s} = e(g,g)^{y_{k,u}s}$ for each authority k. $\underline{\text{Compute } Y_{CA}^s = e(E_{CA}, D_{CA})}$. Combine these values to obtain $\underline{Y_{CA}^s *} \prod_{k=1}^{K} Y_k^s = Y_0^s$. Then $m = E/Y_0^s$.

Theorem 1. *This scheme is sid-secure according to the definition in Section 2.*

First we give some main points of intuition behind the reduction. Then we follow with a more formal proof.

Basis of SW Reduction – We will show that we can reduce the BDH problem to the problem of breaking our encryption scheme. That means we are given $A = g^a, B = g^b, C = g^c$ and asked to distinguish $e(g,g)^{abc}$ from $e(g,g)^R$ for a random $R \leftarrow Z_q$. We assume there exists an adversary that can break the security properties of our multiauthority system (as defined in Section 2) and we show that we could use such an adversary to solve this problem.

We want to show that, even given a challenge encryption and adaptively chosen secret key queries, in our challenge encryption, $M_b = E/e(g,g)^{y_0 s}$ is indistinguishable from a random message (which means the adversary can have no more than negligible probability of correctly identifying b). We will show that $e(g,g)^{y_0 s}$ is indistinguishable from a random element of G_2. Since we are basing our reduction on the BDH assumption, this means we want to implicitly set $y_0 s = abc$. We need to be able to output $e(g,g)^{y_0}$ and g^s as part of the central public key, so we will implicitly set $s = c$ and $y_0 = ab$. (These values cannot be computed, but we will use them to determine the other values in our reduction.)

Extension to Multiple Authorities – Note that the adversary is allowed to request secret keys for a given user u and attribute set $\mathcal{A}_u$ as long as there remains one honest authority k such that $\mathcal{A}_C^k \cap \mathcal{A}_u^k < d$, i.e. the user has insufficient attributes from this authority to decrypt. Thus, in the worst case, for all but one authority k, the adversary will be able to compute $Y_{k,u}^s = e(g,g)^{y_{k,u}s}$ for additive share $y_{k,u}$. Every user will also be able to compute $Y_{CA,u}^s = e(g^s, D_{CA}) = e(g^s, g^{y_0 - \sum y_{k,u}})$. We need $\prod e(g,g)^{y_{k,u}s} * e(g^s, D_{CA}) = e(g,g)^{y_0 s}$ to be something which the adversary cannot compute (in particular,

$e(g,g)^{abc}$ which is indistinguishable from random). Thus, we must "hide" this incomputable value in the share $y_{k,u}$ for the one authority from which the adversary does not have sufficient attributes. Let $\hat{k}$ be this authority. Then we will implicitly set $y_{\hat{k},u} = ab + z_{k,u}b$ for random $z_{k,u}$. For all other honest authorities, we need $e(g,g)^{y_{k,u}s}$ to be computable, so we will implicitly set $y_{k,u} = z_{k,u}b$ for some random $z_{k,u}$ (this particular choice will be explained below). Note that $\hat{k}(u)$ may be a different authority for each user u. This is where the reduction will make use of the PRFs. Since to the adversary the PRFs for honest authorities are indistinguishable from true random functions, we can implicitly replace these PRFs with the necessary values for each user u and authority $k = \hat{k}(u)$ or $k \neq \hat{k}(u)$, and the result will be indistinguishable. (The values will still be randomly distributed).

Answering Secret Key Queries– Note that in order for our reduction to succeed, we need to generate public keys for honest authorities k so that:

- When $k = \hat{k}(u)$, we can output secret keys such that this authority's secret $(q(0) = F_{s_k}(u))$ is uncomputable (eg. $ab + z_{k,u}$), given that the user u does not have sufficient attributes from this authority. This situation is identical to that in a single authority security reduction.
- When $k \neq \hat{k}(u)$, using the *same* public keys, we can output secret keys for users who know (potentially) all of the attributes from this authority as long as this authority's secret is generated appropriately. In this case we set the secret $q(0) = F_{s_k}(u) = z_{k,u}b$.

For $i \in \mathcal{A}_C^k$, we must be able to output $T_{k,i} = g^{t_{k,i}}$ in the public key and $E_{k,i} = T_{k,i}^s = g^{t_{k,i}s} = g^{t_{k,i}c}$ in the challenge ciphertext. Thus, we choose $t_{k,i} = \beta_{k,i}$ for known random $\beta_{k,i}$. For $i \notin \mathcal{A}_C$, we set $t_{k,i} = \beta_{k,i}b$ for known random, $\beta_{k,i}$. (Note that for these attributes $E_{k,i} = T_{k,i}^s$ is not computable).

Consider the second case: We need to output $D_{k,i} = g^{p(i)/b\beta_{k,i}}$ and $D_{k,i} = g^{p(i)/\beta_{k,i}}$, where we require $p(0) = bz_{k,u}$. If we simply choose p in terms of b, (eg. $p = b\rho$ for known random $d-1$ degree polynomial ρ), this is trivial.

The first case, as mentioned above, follows the original single authority reduction almost exactly. The only difference is that now we have an extra randomizing term added ($F_{s_k}(u) = ab + z_{k,u}b$ instead of $F_{s_k}(u) = ab$). For up to $d-1$ points ($i \in \mathcal{A}_C^k \cap \mathcal{A}_u^k$), we need to give secret keys $D_{k,i} = g^{p(i)/t_{k,i}} = g^{p(i)/\beta_{k,i}}$, where $p(0) = ab + z_{k,u}b$. This might seem difficult since we can't compute g^{ab}. However, recall that, using interpolation, we can pick any d points and use them to define a polynomial. So, for the attributes i in the challenge, we will set $p(i)$ to be a random multiple of b. For these attributes $D_{k,i} = g^{p(i)/\beta_{k,i}}$ will be a computable multiple of B. Since we have also fixed $p(0)$, we have now chosen d points on the polynomial, so now for the remaining attributes, we can interpolate to find $g^{p(i)/t_{k,i}}$ as a weighted product of $g^{p(0)/t_{k,i}}$ and $g^{p(j)/t_{k,i}}$ for each of the fixed attributes j (those in the challenge ciphertext). Recall that for these attributes, we have $t_{k,i} = \beta_{k,i}b$. Thus, for each of the fixed attributes j, $g^{p(j)/t_{k,i}}$ will be a computable multiple of B, and for $p(0)$, $g^{p(0)/t_{k,i}} = g^{(ab+z_{k,u}b)/b\beta_{k,i}} = g^{(a+z_{k,u})/\beta_{k,i}}$

is a computable combination of A and g. That means $D_{k,i} = g^{p(i)/t_{k,i}}$ will be also computable for all $i \in \mathcal{A}_u^k - \mathcal{A}_C^k$.

Proof (of Theorem 1). Suppose there exists and adversary that plays the security game as described in Section 2 and succeeds with nonnegligible probability ϵ. Then we will show that we can use such an adversary to break the BDH assumption.

First, we assume that the adversary will still succeed with the same advantage even when the PRF F_{s_k} is replaced by a truly random function for each honest authority k. Note that if this is not the case, then we can distinguish the PRF from random, contradicting the definition of a PRF.

We need to specify how our reduction responds in each stage of the game (as described in Section 2). Our reduction will behave as follows:

- Given $A = g^a, B = g^b, C = g^c$ and $Z = e(g,g)^{abc}$ or $e(g,g)^R$ for random $R \leftarrow Z_q$
- Receive A_C and list of corrupted authorities *Corr* from adversary.
- Init:
  - System PK : $Y_0 = e(A,B)$ (implicitly set $y_0 = ab$.)
  - Honest Authority PK's: Choose random $\beta_{k,i}$; PK: $\{T_{k,i} = g^{\beta_{k,i}}\}_{i \in \mathcal{A}_u \cap \mathcal{A}_C^k}$, $\{T_{k,i} = B^{\beta_{k,i}}\}_{i \in \mathcal{A}_u - \mathcal{A}_C^k}$
  - Corrupt Authority k SK's: Choose random $t_{k,i} \leftarrow Z_q$, random PRF key s_k. SK:$s_k, \{t_{k,i}\}$
- SK queries: Let $\hat{k}(u)$ be the first authority k queried such that $|\mathcal{A}_u^k \cap \mathcal{A}_C^k| < d$.
  - SK queries for user u to Honest Attribute Authorities $k \neq \hat{k}(u)$: Recall that for these authorities we will implicitly set $p(0) = F_{s_k}(u) = z_{k,u}b$. Choose a random $z_{k,u}$ and choose a random polynomial ρ such that $\rho(0) = z_{k,u}$. We will implicitly set $p(i) = b\rho(i)$. Now for $i \in \mathcal{A}_C^k$, $t_{k,i} = \beta_{k,i}$, so $D_{k,i} = g^{p(i)/t_{k,i}} = g^{b\rho(i)/\beta_{k,i}} = B^{p(i)/\beta_{k,i}}$. For $i \notin \mathcal{A}_C^k$, $t_{k,i} = b\beta_{k,i}$, so $D_{k,i} = g^{p(i)/t_{k,i}} = g^{b\rho(i)/b\beta_{k,i}} = g^{\rho(i)/\beta_{k,i}}$. SK: $\{D_{k,i} = B^{\rho(i)/\beta_{k,i}}\}_{i \in \mathcal{A}_u^k \cap \mathcal{A}_C^k}, \{D_{k,i} = g^{\rho(i)/\beta_{k,i}}\}_{i \in (\mathcal{A}_u^k - \mathcal{A}_C^k)}$
  - SK queries for user u to Honest Attribute Authorities $k = \hat{k}(u)$: Recall that for authority $\hat{k}$ for user u, we will choose random $r_{k,u}$ and implicitly set $p(0) = F_{s_k}(u) = ab + z_{k,u}b$. Choose $d-1$ random points v_i. For $i \in \mathcal{A}_C^k$, we will implicitly set $p(i) = v_i b$. For these attributes, $t_{k,i} = \beta_{k,i}$, so that means $D_{k,i} = g^{p(i)/t_{k,i}} = g^{bv_i/\beta_{k,i}} = B^{v_i/\beta_{k,i}}$. Recall that we need $p(0) = F_{s_k}(u) = ab + z_{k,u}b$, and we have now set $p(i) = v_i b$ for $d-1$ other points. Thus, p is fully determined, and by interpolation, for any other attribute i, we have implicitly defined $\Delta_0(i)(ab + z_{k,u}b) + \sum \Delta_j(i)v_j b$. For these attributes $t_{k,i} = b\beta_{k,i}$, so $D_{k,i} = g^p(i)/t_{k,i} = g^{\frac{\Delta_0(i)(ab+z_{k,u}b)+\sum \Delta_j(i)v_j b}{b\beta_{k,i}}} = g^{\Delta_0(i)a} * g^{\frac{\Delta_0(i)z_{k,u}+\sum \Delta_j(i)v_j}{\beta_{k,i}}} = A^{\Delta_0(i)} * g^{\frac{\Delta_0(i)z_{k,u}+\sum \Delta_j(i)v_j}{\beta_{k,i}}}$
    SK: $\{D_{k,i} = B^{v_i/\beta_{k,i}}\}_{i \in \mathcal{A}_u^k \cap \mathcal{A}_C^k}$,
    $\{D_{k,i} = A^{\Delta_0(i)} * g^{\frac{\Delta_0(i)z_{k,u}+\sum \Delta_j(i)v_j}{\beta_{k,i}}}\}_{i \in (\mathcal{A}_u^k - \mathcal{A}_C^k)}$.

- SK queries for user u to Central Authority:
  $D_{CA} = g^{\,(\sum_{k \notin Corr} z_{k,u} - \sum_{k \in Corr} F_{s_k}(u)))}$
– Receive M_0, M_1 and pick a random b.
– Challenge Ciphertext: Zm, $E = g^s = C$, $\{C^{\beta_{k,i}}\}_{i \in \mathcal{A}_C}$
– Guess for Z: Receive a guess b' and If $b = b'$ guess "$e(g,g)^{abc}$" otherwise guess "$e(g,g)^R$".

After making all his secret key queries, the adversary will send a guess b' that the message encrypted was $M_{b'}$. Note that, if $Z = e(g,g)^{abc}$, then the encryption given was a valid encryption of M_b, and the adversary should have his usual non-negligible advantage ϵ of correctly identifying M_b. However, if $Z = e(g,g)^R$, then the encryption was a completely random value, so the adversary can have no better than 1/2 probability of guessing correctly. Thus, if the adversary guesses correctly, we guess that $Z = e(g,g)^{abc}$ and if he is wrong, we guess that $Z = e(g,g)^R$. If we analyze the probability that the reduction successfully distinguishes Z, we find that the reduction has an advantage of $\epsilon/2$. Thus an adversary which breaks this encryption scheme with advantage ϵ implies an algorithm for breaking the BDH Assumption with nonegligible advantage $\epsilon/2$. We can conclude that this encryption scheme is sid-secure.

Remark 1. Note that our reduction relies critically on the fact that, for each user u that the adversary queries about, we choose exactly one authority for which that user has less than d of the attributes in the challenge, and this is the authority for which we set $F_{s_k}(u) = a + z_{k,u}b$. Note that, if the adversary requests at least d attributes for u from this authority, even if none of them appears in the challenge, the value of $F_{s_k}(u)$ is completely determined by the secret keys that the authority returns (although it is not known, since the discrete logs t_i of the public key T_i are not known to the adversary.)

If we allowed the adversary to at some later point request a second set of attributes from this authority for this user, such that it overlapped by at least d with the challenge, we would not be able to give these secret keys in such a way that they would be consistent with the previously determined value of F_{s_k}. (Doing so would involve computing g^{ab}.)

To prevent this, we require that in order to change his attribute set, a user must also change his GID. However, our collusion resistance requires that keys from different authorities for different keys be incompatible. Thus, it is not clear how we could allow the adversary to change the attribute set for a user without obtaining a new key from all authorities using a completely new GID. See Section 2 for a discussion of possible ways to get around this problem.

5 MultiAuthority + Large Universe and Complex Access Structures

In [GPSW06], Goyal et al. showed a Large Universe Access Control Structure ABE scheme. We can also apply our techniques to this scheme to achieve a corresponding multiauthority system.

A large universe construction has the advantage that the size of the public keys is dependent only on the maximum number of attributes which can be required in an encryption. (In contrast, the basic SW scheme has public keys size proportional to the total number of attributes in the system.) Thus, this method allows a much larger universe of attributes. A large universe construction was first presented in [SW05].

Goyal et al. showed how to combine the large universe construction with a construction (GPSW) which allows the authority to give secret keys corresponding to a more general policy than simple t-out-of-n threshold . In particular, they allow any policy that can be described by an access tree. In such a tree, each leaf node corresponds to an attribute, and intermediate nodes are t-out-of-n gates for arbitrary values of t and n. (Thus, OR and AND gates are possible as special cases.) If we consider the input at each leaf node to be 1 if and only if the corresponding attribute is present in a given ciphertext, then evaluating this circuit will determine whether or not the user should be able to decrypt this ciphertext.

Below we give a multiauthority large universe access control structure ABE scheme. In this system, each authority will choose an access structure τ_k for the user and give him a corresponding secret key. A user will be able to decrypt a ciphertext with attribute set $\mathcal{A}_C$ if and only if all of his access structures τ_k output 1 when evaluated on the corresponding subset of the attributes, $\mathcal{A}_C^k$.

We use essentially the same techniques as in the simple multi-authority scheme. Each authority runs a copy of the single authority protocol, with a separate copy of the public key (in this case $t_{k,1}, \dots t_{k,n+1}$). The master secret for each authority (y_k) is replaced by a PRF on the user's ID, so that the master secret for the entire system (y_0) can be divided among all the authorities in a different way for each user. Finally, a central authority is necessary to ensure that, for each user, the PRFs from all authorities can be combined to obtain the system secret.

MultiAuthority Scheme for Large Universe and Complex Access Structures

System

Init Fix prime order groups G, G_1, bilinear map $e : G \to G_1$, and generator $g \in G$. Choose PRF keys $s_1 \dots s_K$ and $y_0 \leftarrow Z_q$, and $g_2 \leftarrow G_1$.

System Public Key $g_1 = g^{y_0}$.

Attribute Authority k

Authority Secret Key s_k

Authority Public Key $t_{k,1} \dots t_{k,n+1} \leftarrow G_1$.

Let $h(x)$ be the n degree polynomial defined by $t_{k,1}, \dots t_{k,n+1}$. Also define $T_k(x) = g_2^{x^n} g^{h(x)} = g_2^{x^n} \prod_{i=1}^{n+1} t_{k,i}^{\Delta_i(x)}$

Secret Key for User u **for access structure** τ_k Let $y_{k,u} = F_{s_k}(u)$. Now run the Key Generation from GPSW but with the master secret $y = y_{k,u}$ and with the points $t_{k,1}, \dots t_{k,n+1}$, i.e. KeyGeneration(τ, $MK = y_{k,u}$ and $PK = g^{y_{k,u}}, g_2, t_{k,1}, \dots t_{k,n+1}$): We choose a polynomial q_r for the root node of the tree τ_k such that $q_r(0) = y_{k,u}$. Then we choose random polynomials for all other nodes x such that $q_x(0) = q_{parent(x)}(x)$. Finally, for the leaf

node x, we choose random $r_{k,x}$ and compute secret key elements: $D_{k,x} = g_2^{q_x(0)} T(i)^{r_{k,x}}$ where $i = att(x)$ and $R_{k,x} = g^{r_{k,x}}$.
SK: $D_{k,x}, R_{k,x}$ for all leaf nodes x in τ_k

Central Authority

Central Authority Secret Key s_k for all authorities k, y_0.

Secret Key for User u Let $y_{k,u} = F_{s_k}(u)$ for all k. Secret Key: $D_{CA} = g_2^{(y_0 - \sum_{i=0}^{K} y_{k,u})}$.

Encryption for attribute set $\mathcal{A}_C$ Choose random $s \leftarrow Z_q$. $E = e(g_1, g_2)^s m$, $E' = g^s$, $\{E_{k,i} = T_k(i)^s\}_{i \in \mathcal{A}_C^k, \forall k}$.

Decryption: For each authority k, run the Decryption algorithm as in GPSW, i.e. run DecryptNode on the root of the tree τ_k:
For leaf node x if we have $D_{k,x}$, this algorithm computes $\frac{e(D_{k,x}, E')}{e(R_{k,x}, E_{k,i})} = \frac{e(g_2^{q_x(0)} T_k(i)^{r_{k,x}}, g^s)}{e(g^{r_{k,x}}, T_k(i)^s)} = \frac{e(g_2^{q_x(0)}, g^s) e(T_k(i)^{r_{k,x}}, g^s)}{e(g^{r_{k,x}}, T_k(i)^s)} = e(g, g_2)^{q_x(0)s}$. Then at each level of the tree, if we have successfully computed DecryptNode for sufficient children, it combines the results to obtain $e(g, g_2)^{q_x(0)s}$ for parent node x. Finally, if we have sufficient attributes for the tree τ_k, DecryptNode computes $e(g, g_2)^{q_r(0)s} = e(g, g_2)^{y_{k,u}s}$.
Now decryption proceeds as in the previous multiauthority scheme: If we have sufficient attributes to decrypt, then for every authority we will have computed $Y_{k,u}^s = e(g, g_2)^{y_{k,u}s}$. Next, use the key obtained from the central authority to compute $Y_{CA}^s = e(E', D_{CA}) = e(g^s, g_2^{y_0 - \sum_{i=1}^{K} y_{k,u}})$. Combine these values to obtain $Y_{CA}^s * \prod_{k=1}^{K} Y_k^s = e(g, g_2)^{y_0 s} = Y_0^s$. Then $m = E/Y_0^s$.

We will present only the key intuition for the proof of security. For a full proof, see the full version.

Basis of Single Authority Reduction – We again want to show a reduction from BDH, so we want the quantity which we will use to blind the message, in this case $e(g_1, g_2)$, to be equal to $e(g, g)^{abc}$. Thus, we will set $g_1 = a, g_2 = b$, and implicitly, $s = c$.

Extension to Multiple Authorities – Again here, as in the simple multiauthority scheme, for each user u, the adversary is allowed to request keys that are sufficient to decrypt the ciphertext from all but one authority. In this scheme, in the worst case, for all but one authority k, the adversary will be able to compute $Y_{k,u}^s = e(g_1, g_2)^{y_{k,u}s}$. Every user will also be able to compute $Y_{CA,u}^s = e(g^s, D_{CA}) = e(g, g_2)^{(y_0 - \sum y_{k,u})s}$. And, as mentioned above, we need $\prod e(g, g_2)^{y_{k,u}s} * e(g^s, D_{CA}) = e(g, g_2)^{y_0 s}$ to be an uncomputable value (in particular $e(g, g)^{abc}$). Thus, again we must "hide" this incomputable value in the share $y_{k,u}$ for the one authority $\hat{k}$ from which this user does not have sufficient attributes. Thus, we set $y_{\hat{k},u} = a + z_{\hat{k},u}$ and for all other honest authorities, $y_{k,u} = z_{k,u}$ for known random $z_{k,u}$. Once again, we make use of the pseudorandomness of the PRFs to claim that, since all these values are distributed randomly, the result will be indistinguishable from computing the values using a true PRF.

ANSWERING SECRET KEY QUERIES– Once again, we must show that, for all users u, we can form public keys for all honest authorities k such that:

- When $k = \hat{k}(u)$, we can output secret keys such that this authority's secret $(q(0) = F_{s_k}(u))$ is not known (eg. $a + z_{k,u}$), given that the user u does not have sufficient attributes from this authority. This situation is identical to that in a single authority security reduction.
- When $k \neq \hat{k}(u)$, using the *same* public keys, we can output secret keys for users who know (potentially) all of the attributes form this authority as long as this authority's secret is generated appropriately. In this case we set the secret $q(0) = F_{s_k}(u) = z_{k,u}$.

We set up the $t_{k,i}$ values for each authority k as in the GPSW reduction. This is done in such a way that if $i \in \mathcal{A}_C^k$, $T_k(i)^s$ is computable (recall that this must be part of the encryption), but $D_{k,x} = g_2^{q_x(0)} T(i)^{r_{k,x}}, R_{k,x} = g^{r_{k,x}}$ is computable if and only if we know $q_x(0)$. If $i \notin \mathcal{A}_C^k$, $T_k(i)^s$ will not be computable, but we will be able to compute $D_{k,x}, R_{k,x}$ when given only $g^{q_x(0)}$.

Now, in the first situation, we can proceed as in the single authority reduction, and run the PolyUnsat algorithm [GPSW06] to set the polynomials for each level of the tree so that at the root r, $q_r(0) = a + z_{k,u}$. At the leaves, this will make $q_x(0)$ completely known for nodes corresponding to $i \in \mathcal{A}_C$ and will make $g^{q_x(0)}$ known for $i \notin \mathcal{A}_C$. According to the setup above, this lets us form $D_{k,x}, R_{k,x}$ for all required leaf nodes x. In the second case, we simply proceed as in the real protocol.

Finally, the challenge encryption can be formed as $E = ZM_b, E' = C, \{E_{k,i} = T(i)^s\}$ for random bit b. The reduction's output and the analysis proceed as in the previous reduction.

Remark 2. Note that we cannot allow changes in a user's access structure (without corresponding changes in the user's GID) for the same reason that in the threshold multiauthority scheme we cannot allow changes in a user's attribute set (see Section 4).

6 Extensions

We briefly describe several possible extensions to our scheme. For security proofs, see full version.

CHANGING d_k SW noted that one could easily extend their scheme to allow d, the number of attributes in the ciphertext required to decrypt, to vary with each encryption. Essentially, the scheme would be instantiated with $d = dmax$, the maximum overlap one might want to require. We would also extend the attribute set by adding $dmax$ dummy attributes and each user would get a secret key element D_i for each of these new attributes. We refer to the set of dummy attributes as $\mathcal{A}_D$. If the encryptor wanted to require $d' < dmax$ of the attributes in the ciphertext, he could include $E_i = T_i^s$ for $dmax - d'$ dummy attributes $i \in \mathcal{A}_D$.

Note that in our mulitauthority scheme a similar approach would allow one to choose exactly how many of the attributes given in the ciphertext to require from each authority, i.e to vary the values d_k. We need only add $dmax_k$ dummy attributes for each authority k and proceed as described above.

Leaving out authorities In the mulitauthority scheme as stated, each user must go to every authority before he can decrypt any message. Using the technique above, an encryptor can allow decryption by someone who has none of the attributes handled by a specific authority by including all of the dummy attributes for that authority in the encryption set. However, a user would still have to go to that authority to obtain secret keys for these dummy values in order to decrypt. There could be an arbitrarily large number of authorities in the system, so this might involve a lot of work.

We can remove this requirement by adding one "authority attribute k" for each authority. We would add a corresponding $T_{Nk} = g^{t_{Nk}}$ to the public key and the central authority would give every user u a secret key for each authority: $D_{Nk} = g^{y_{k,u}/t_{Nk}}$. To encrypt without requiring any attributes from authority k, the encryptor would include T_{Nk}^s in the encryption. A user could then combine this with his D_{Nk} value and obtain $Y_{k,u}^s = e(g,g)^{y_{k,u}s}$, thus bypassing the need for any attributes from authority k.

More complicated functions of the authorities Our basic scheme as described in Section 4 requires that a user have sufficient attributes from all of the authorities in order to decrypt. However, we might want to allow a user to decrypt if he had sufficient attributes from at least D of the authorities.

We have to make the following changes to our basic scheme: Our central authority will now choose a random $D-1$ degree polynomial P with $P(0) = y_0$. For each authority k he will compute $P(k)$, and then compute a value which combined with $Y_{k,u}^s = e(g,g)^{y_{k,u}s} = e(g,g)^{F_{s_k}(u)s}$ will give $e(g,g)^{P(k)s}$. If the user obtains D of these values he can interpolate to find $e(g,g)^{P(0)s} = e(g,g)^{y_0 s} = Y_0^s$ and then obtain m. Thus, the secret key from the central authority for user u will be $D_{CA} = \{g^{P(k)-F_{s_k}(u)}\}_{k=1\ldots K}$.

Adding Attribute Authorities We can also allow the central authority to add additional attribute authorities to the system at any point in the execution of the scheme.

In the basic scheme, where attributes from all authorities are required for decryption, this will occur as follows: The central authority will choose a new system public key $Y_0' = e(g,g)^{y_0'}$, and all future encryption will be relative to this public key. The central authority will store the PRF seed for the new authority, and all secret keys it gives out will be computed using y_0' and the new enlarged set of PRFs. Note that this means that all users will need to obtain a new key from the central authority in order to decrypt any messages encrypted under the new key. However, they will not need to obtain new keys from any of the old attribute authorities.

In the scheme described in the above section, which uses a threshold over authorities, the central authority has simply to give each user an additional value for the new authority k: $e(g,g)^{P(k)-F_{s_k}(u)}$. Note that in this case, if a user does not intend to use any attributes from this new authority, he need not obtain a new key from the central authority.

Single Authority CNF Attribute Encryption In [SW05] it was left open whether decryption ability could be determined by a more complicated function of a user's attributes. Looked at differently, our scheme can be viewed as a single authority scheme in which the attributes necessary for decryption are given by a CNF formula chosen by the encryptor. In this case, each authority corresponds to a clause, and $d_k = 1$ for all authorities.

A user would obtain all of his secret keys ($\{D_{k,i}\}$ and D_{CA}) from the authority and then would be able to decrypt any message for which he had at least one of the specified attributes $i \in \mathcal{A}_C^k$ for each clause k.

The only additional complication is that our multiauthority scheme required that each authority's attribute set be disjoint. Thus, the set of attributes allowed in each clause must be disjoint. To get around this, we create a separate copy of each attribute for each clause in which it could possibly appear. Thus, if a user has attribute i, he will have in his secret key $D_{k,i}$ for every clause k in which attribute i could appear. Then the encryptor includes $T_{1,i}^s$ if attribute i appears in the first clause, and $T_{2,i}^s$ if it occurs in the second clause, etc.

Acknowledgements Thanks to Anna Lysyanskaya for advice and encouragement, and to Brent Waters for helpful comments and suggestions. The author is supported by NSF grant CNS-0374661 and NSF Graduate Research Fellowship.

References

[BB04] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity based encryption without random oracles. In *Proc. of EUROCRYPT 2004*, volume 3027, *LNCS*, 54–73. Springer.

[BF01] Dan Boneh and Matthew Franklin. Identity-based encryption from the Weil pairing. In *Proc. of CRYPTO 2001*, volume 2139, *LNCS*, 213–229. Springer.

[CHK03] Ran Canetti, Shai Halevi, and Jonathan Katz. A forward-secure public-key encryption scheme. In *Proc. of EUROCRYPT 2003*, volume 2656, *LNCS*, 255–271. Springer.

[Coc01] C. Cocks. An identity based encryption scheme based on quadratic residues. In *Proc. of Cryptography and Coding, 8th IMA International Conference*, volume 2260,*LNCS*, 360–363. Springer, 2001.

[Fel87] P. Feldman. A practical scheme for non-interactive verifiable secret sharing. In *Proc. of FOCS 1987*, 427–437.

[GPSW06] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of CCS 2006*, 89–98, New York. ACM Press.

[Sha85] Adi Shamir. Identity-based cryptosystems and signature schemes. In *Proc. of CRYPTO 1984*, volume 196, *LNCS*, 47–53. Springer.

[SW05] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *Proc. of EUROCRYPT 2005*, volume 3494, *LNCS*, 457-473. Springer.

[Wat05] Brent Waters. Efficent identity based encryption without random oracles. In *Proc. of EUROCRYPT 2005*, volume 3494, *LNCS*, 114–127. Springer.