Improving Privacy and Security in Multi-Authority Attribute-Based Encryption

Melissa Chase
Microsoft Research
1 Microsoft Way
Redmond, WA 98052, USA
melissac@microsoft.com

Sherman S.M. Chow*
Department of Computer Science
Courant Institute of Mathematical Sciences
New York University, NY 10012, USA
schow@cs.nyu.edu

ABSTRACT

Attribute based encryption (ABE) [13] determines decryption ability based on a user's attributes. In a multi-authority ABE scheme, multiple attribute-authorities monitor different sets of attributes and issue corresponding decryption keys to users, and encryptors can require that a user obtain keys for appropriate attributes from each authority before decrypting a message. Chase [5] gave a multi-authority ABE scheme using the concepts of a trusted central authority (CA) and global identifiers (GID). However, the CA in that construction has the power to decrypt every ciphertext, which seems somehow contradictory to the original goal of distributing control over many potentially untrusted authorities. Moreover, in that construction, the use of a consistent GID allowed the authorities to combine their information to build a full profile with all of a user's attributes, which unnecessarily compromises the privacy of the user. In this paper, we propose a solution which removes the trusted central authority, and protects the users' privacy by preventing the authorities from pooling their information on particular users, thus making ABE more usable in practice.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public key cryptosystems

General Terms

Security, Algorithms, Design

Keywords

attribute based encryption, anonymous credential, privacy, multi-authority, removing trusted party

*Work done while an intern with Microsoft Research

CCS'09, November 9–13, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

We often identify people by their attributes. In 2005, Sahai and Waters [13] proposed a system (described in more recent terminology as a key-policy attribute-based encryption (ABE) system for threshold policies) in which a sender can encrypt a message specifying an attribute set and a number d, such that only a recipient with at least d of the given attributes can decrypt the message. However, the deployment implications of their scheme may not be entirely realistic, in that it assumes the existence of a single trusted party who monitors all attributes and issues all decryption keys. Instead, we often have different entities responsible for monitoring different attributes of a person, e.g. the Department of Motor Vehicles tests whether you can drive, a university can certify that you are a student, etc. Thus, Chase [5] gave a multi-authority ABE scheme which supports many different authorities operating simultaneously, each handing out secret keys for a different set of attributes.

However, this solution was still not ideal. There are two main problems: one concern of security of the encryption, the other the privacy of the users.

1.1 Protecting the User's Privacy

Since each authority is responsible for different attributes, we want to allow them to issue decryption keys independently, without having to communicate with one another. As argued in [5], in order to prevent collusion in such a setting, we need some consistent notion of identity. (Otherwise, a user could easily obtain keys from one authority and then give them all to a friend.) The solution in that work is to require that each user have a unique global identifier (GID), which they must present to each authority (and to require that the user prove in some way that he is the owner of the GID he presents).[1] Unfortunately, the *mere existence of GID* makes it very hard for the users to guarantee *any* kind of privacy. Because a user must present the same GID to each authority, it is very easy for colluding authorities to pool their data and build a "complete profile" of all of the attributes corresponding to each GID. However, this might be undesirable, particularly if the user uses the ABE system in many different settings, and wishes to keep information about some of those settings private.

This situation seems to be unavoidable if all one's attributes are determined by some kind of public identity like a name or SSN – in that case users will need to identify themselves in any case in order to get the decryption keys for a certain set of attributes, so privacy is unavoidably com-

[1]see [5] for further discussion.

promised. However, there are many attributes which do not belong to this category. The ability to drive is a good example. One should be able to prove the ability to do something in an examination and then get the corresponding credential, without presenting any identifying information. Alternatively, one might interact with a service via a pseudonym (e.g. a login name) and wish to obtain attributes relating to this interaction without revealing one's full identity.

Regardless, as the attribute-authorities (AAs) are responsible for managing each user's attributes, it seems inevitable that they will learn which subsets of its attributes are held by different users. However, we could imagine applications where some of the authorities are different online service providers giving attributes related to online activities like blog/wiki contributions, access to online news sites, participation in social networking sites, or purchases at an online store. In this case, it would make sense for the user to be able to maintain different, unlinkable attribute sets with each authority. At the same time, it also makes sense for each AA to gather the statistics of their system usage (e.g. the number of users subscribed a particular service as indicated by the number of users who requested a decryption key for a certain attribute) without compromising individual's privacy.

1.2 Removing the Trusted Authority

The solution presented in [5] assumed the presence of a single trusted "central authority" (CA) in addition to the attribute authorities. This CA did not manage any attributes, but was responsible for issuing each user a unique key.

To see why the CA is crucial in [5], we first recall the intuition. The idea was that, for each user, each AA would use his own secret (not known by other AAs) to generate a share of a system-wide master secret key. The authorities needed to be able to generate these shares *independently* (i.e. without communicating with any other authority during user key issuing). At the same time, in order to prevent collusion it is necessary to use a *different* sharing for each user. This made it difficult to guarantee that all shares always add up to the same master secret. The solution was to have the CA issue each user a special value to cancel out all these shares from the AAs and enable the user to "recover" a function of the system-wide master secret key. Obviously, this computation requires the CA to know the master secret of the system, and the secret information of each AA. This implies that it must also have the power to decrypt any ciphertext.

However, this decryption power seems somehow contradictory to the original motivation of distributing control of the attributes over many potentially untrusted authorities. Thus, we asked whether it would be possible to instead distribute the functionality of the CA over all of the AAs, so that as long as some of them are honest, the scheme will still be secure.

1.3 Our Contributions

Here we present a multi-authority ABE with user privacy and without the trusted authority. These requirements are non-trivial to satisfy, due in both cases to the collusion resistance requirement of ABE.

Brent Waters suggested an approach for removing the CA requirement, in which each pair of attribute authorities would share a secret key. We formalize this idea, and prove that it is secure as long as at least two of the AAs are honest. The new solution uses techniques for distributed pseudorandom functions (PRF) introduced in [11].

Note that Lin *et al.* [10] recently proposed a different approach for building a multi-authority ABE scheme without a central authority. However, their construction requires designers to fix a constant m for the system, which directly determines efficiency. The resulting construction is such that any group of $m + 1$ colluding users will be able to break security of the encryption. Our scheme on the other hand is secure no matter how many users collude.

We also present an anonymous key issuing protocol which allows multi-authority ABE with enhanced user privacy – 1) we allow the users to communicate with AAs via pseudonyms instead of having to provide their GIDs in the clear, and 2) we prevent the AAs from pooling their data and linking multiple attribute sets belonging to the same user.

As a building block we construct a protocol for an oblivious computation of a key of the form $(SK \cdot PRF_\beta(u))^\gamma$, where u is a user's GID, [2] SK represents some secret information related to the private key of an authority, β is the secret seed for the PRF owned by an authority and γ corresponds to some secret related to an attribute controlled by an authority. The key is produced obliviously, i.e. without either the authority or the user revealing any of their secret information ((SK, β, γ) or u respectively). We chose to present the protocol in this "generic" way (without coupling with any particular ABE scheme) to illustrate its applicability. Our protocol can be applied to Chase system (with a little modification) in a rather straightforward manner (see full version for details). We also show how to efficiently apply this protocol to our scheme which removes the CA. (In this case the keys are a bit more complex, so we need somewhat more involved techniques - see Section 5.)

Finally, our results may be of additional interest because they show new applications of the distributed PRF of Naor, Pinkas, and Reingold [11], and a generalization of the oblivious PRF techniques of Jarecki and Liu [9].

2. RELATED WORK

2.1 ABE for Different Policies

ABE is actually a generalization of IBE (identity-based encryption [14]): in an IBE system, ciphertexts are associated with only one attribute (the identity).

The ABE scheme of Sahai-Waters [13] was proposed as a fuzzy IBE scheme, which allowed for some error tolerance around the chosen identity. In more recent terminology, it would be described as a key-policy (KP) ABE scheme that allows for threshold policies. Key-policy means that the encryptor only gets to label a ciphertext with a set of attributes. The authority chooses a policy for each user that determines which ciphertexts he can decrypt. A threshold policy system would be one in which the authority specifies an attribute set for the user, and the user is allowed to decrypt whenever the overlap between this set and the set associated with a particular ciphertext is above a threshold.

Goyal *et al.* [8] proposed a KP-ABE scheme which supports any monotonic access formula consisting of AND, OR, or threshold gates. A construction for KP-ABE with non-monotonic access structures (which also include NOT gates, i.e. negative constraints in a key's access formula) was pro-

[2] See footnote 6.

posed by Ostrovsky, Sahai and Waters [12]. All of these schemes are characterized as key-policy ABE since the access structure is specified in the private key, while the attributes are used to describe the ciphertexts.

The roles of the ciphertexts and keys are reversed in the ciphertext-policy ABE (CP-ABE) introduced by Bethencourt, Sahai and Waters [2], in that the ciphertext is encrypted with an access policy chosen by an encryptor but a key is simply created with respect to an attributes set. The security of their scheme is argued in the generic group model. Recently, [15] proposed CP-ABE constructions based on a few different pairing assumptions which work for any access policy that can be expressed in terms of an LSSS matrix.

In this paper, we will look only at the KP-ABE setting. We will look at both the simple threshold, and the more complicated monotonic access structure case, and will build a construction based on the same assumptions as Sahai and Waters [13] and Goyal *et al.*[8]. Both non-monotonic access structures and the ciphertext policy schemes require much stronger assumptions, and very different techniques, so we will not consider these cases in our work.

2.2 Multi-Authority ABE

All of the prior work described above considers the scenario where all of the attributes are monitored by a single authority. However, as we mentioned in Section 1, it seems natural that one might want to divide control of the various attributes over many different authorities. The main challenge here is to guarantee that two colluding users cannot each obtain keys from a different authority, and then pool their keys to decrypt a message that they are not entitled to. Furthermore, in the multi-authority case, we may wish to allow for some of the authorities to be untrusted. The techniques for single authority ABE cannot be easily generalized in this case – they rely on the fact that the single authority can generate all of a user's keys at once, to ensure that they can only be used together, and cannot be combined with any other user's keys.

The only multi-authority ABE schemes we are aware of are Chase's original proposal [5] (which has already been discussed in Section 1) and the very recent Lin *et al.* extension [10]. Both schemes are KP-ABE and operate in a setting where multiple authorities are responsible for disjoint sets of attributes. The disadvantages of Chase's scheme have already been discussed in Section 1.

The scheme of [10], like the scheme we will present here, has the advantage that it does not rely on a central authority. However, their scheme only achieves m-*resilience*, in that security is only guaranteed against a maximum of m colluding users. (In contrast, the results of [5] and our new results consider a much stronger model, which remains secure against any number of colluding users.) And this is not merely an issue of formal security: Lin *et al.* demonstrated a collusion attack of $m+1$ users [10]. In their scheme m is the number of secret keys that each authority obtains from a distributed key generation protocol. (This also means m must be determined when the system is initialized.) Clearly, for a large-scale system, m should set reasonably high in order to guarantee security (a very loose desirable lower bound should be N^2, where N is the number of authorities). This imposes burdens on the interactive distributed key generation protocol among all the authorities, and on their secure storage. Finally, $O(m)$ online modular operations are required by each authority to issue secret keys to a user. We further note that this weaker notion of security seems undesirable. It may be of commercial interest to have as many users as possible, yet it simultaneously increases the risk of being compromised. (Even if users themselves are not malicious, one might worry about malware on a user's machine, or information leaked unintentionally through side channels.) Thus, we argue that it is still a very important open problem to design an *efficient* and *secure* multi-authority ABE scheme without a trusted CA, and this is one of the problems we will attempt to solve here.

2.3 Anonymous Credentials

Up until now, there has been little relationship between anonymous credentials and ABE (except a recent work in [6] which borrows some techniques from anonymous credential to address the key-escrow problem of IBE). In our new schemes we will make use of some basic techniques in anonymous credential systems to protect the privacy of ABE users.

In an anonymous credential system (see [3, 4]), users wish to obtain and prove possession of credentials while remaining anonymous. In such work it is assumed that each user has a unique secret key (and there are different proposals for how to prove that a given key is valid and to prevent users from loaning out their keys). Then the user can interact with each authority under a different pseudonym in such a way that it is impossible to link multiple pseudonyms belonging to the same user. At the same time, all of a user's pseudonyms, and the resulting credentials, are tied to the same secret key so that the user can prove that he has *both* attribute set $\mathbb{A}$ from one authority and set $\mathbb{B}$ from another.

We will use techniques from anonymous credentials to allow the users to obtain decryption keys from the authorities without revealing their GID's.

The basic idea is to let the GID play the role of the anonymous credential secret key. We will now assume that each user has a unique and *secret* GID value. He interacts with authorities using pseudonyms based on this value, and thus obtains decryption keys.[3] Thus, we will replace the GID with the assumption that each user has unique secret key as in an anonymous credential system. Guaranteeing that this secret key is unique involves a number of subtle issues. Standard techniques can be found from the anonymous credential literature. Note, however, that anonymous credentials do not immediately solve the privacy issue in an ABE setting. Consider the following proposal: The user interacts with the authority via a pseudonym. When the user wants to obtain decryption keys corresponding to a set of attributes, he proves (via the anonymous credential system) that he is the owner of a credential for these attributes. Then he uses the ABE system to obtain decryption keys. This idea seems straightforward, but in fact it is unclear how to satisfy our security and privacy requirements. First, the existing constructions for multi-authority ABE schemes (by Chase [5] and Lin *et al.* [10]) require that the user presents the GID in the clear to each authority. The authority then uses this GID to generate the user's decryption keys, in order to ensure collusion-resistance. This obviously does not provide any user privacy. On the other hand, if the user was allowed

[3]Another option would be to allow the user to reveal the GID to select authorities, but to require that there be some additional secret information that was known only to the user, to prevent impersonation.

to present a different anonymized value to each authority, then we would no longer be able to guarantee the security of the multi-authority ABE against colluding users.

Instead, we will solve the privacy problem by designing a protocol by which a user can obtain a set of decryption keys for his secret GID without revealing any information about that GID to the authority. At the same time, the authority is guaranteed that the agreed upon decryption keys are the only thing that the user learns from the transaction.

Finally, we stress that, although we use several elements of anonymous credential systems, our solution does not encrypt with respect to a user's secret key. This is still strictly an attribute-based encryption system, in which decryption ability is determined only by a user's attributes. The secret key/GID is only used in communicating with the various authorities, and in determining the appropriate decryption keys.

3. PRELIMINARIES

3.1 Notations and Complexity Assumptions

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two cyclic multiplicative groups of prime order q generated by g_1 and g_2 respectively, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map such that $\forall x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and a, $b \in \mathbb{Z}_q$, $\hat{e}(x^a, y^b) = \hat{e}(x, y)^{ab}$ and $\hat{e}(g_1, g_2) \neq 1$. Let $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ be a computable isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$, with $\psi(g_2) = g_1$. $(\mathbb{G}_1, \mathbb{G}_2)$ are said to be admissible bilinear groups if the group action in $\mathbb{G}_1$, $\mathbb{G}_2$, the isomorphism ψ and the bilinear mapping $\hat{e}$ are all efficiently computable.

DEFINITION 1. *The Decisional Diffie-Hellman (DDH) problem in prime order group $\mathbb{G} = \langle g \rangle$ is defined as follows: on input $g, g^a, g^b, g^c \in \mathbb{G}$, decide if $c = ab$ or c is a random element of $\mathbb{Z}_q$.*

DEFINITION 2. *Let algorithm $\mathsf{BDH\_Gen}(1^\lambda)$ output the parameters $(\hat{e}(\cdot,\cdot), q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ where there is an efficiently computable isomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$. The Decisional Bilinear Diffie-Hellman (DBDH) problem is defined as follows: given $g_1 \in \mathbb{G}_1$, $g_2, g_2^a, g_2^b, g_2^c \in \mathbb{G}_2$ and $Z \in \mathbb{G}_T$ as input, decide if $Z = \hat{e}(g_1, g_2)^{abc}$ or $\hat{e}(g_1, g_2)^R$ for $R \in_R \mathbb{Z}_q$.*

The security of the ABE schemes by Sahai-Waters [13], Goyal *et al.* [8], and Chase [5], and of our construction rely on the intractability of the DBDH problem.

DEFINITION 3. *The k-Decisional Diffie-Hellman Inversion (k-DDHI) problem in prime order group $\mathbb{G} = \langle g \rangle$ is defined as follows: On input a $(k+2)$-tuple $g, g^s, g^{s^2}, \ldots, g^{s^k}, g^u \in \mathbb{G}^{k+2}$, decide if $u = 1/s$ or u is a random element of $\mathbb{Z}_q$.*

For our key issuing protocol, we will use a modified version of the of the Dodis-Yampolskiy PRF [7], suggested in [9], which relies on the intractability of the k-DDHI problem in group G_1 of a pairing. Note that k-DDHI is solvable when given a DDH oracle, thus we must also make the following assumption:

DEFINITION 4. *Let $\mathsf{BDH\_Gen}(1^\lambda)$ output the parameters for a bilinear mapping $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. The eXternal Diffie-Hellman (XDH) assumption states that, for all probabilistic polynomial time adversaries $\mathcal{A}$, the DDH problem is hard in $\mathbb{G}_1$. This implies that there does not exist an efficiently computable isomorphism $\psi' : \mathbb{G}_1 \to \mathbb{G}_2$.*

3.2 Definitions of Multi-Authority ABE

We begin by defining a multi-authority ABE scheme with a trusted setup (but without an online trusted CA), and without any privacy guarantees. For now, we consider a key-policy threshold scheme, where the user's decryption key corresponds to a set of attributes and a threshold value. (See Section 6 for an extension to more general policies.) In Section 4 we will discuss an extension which allows a user to obtain decryption keys without leaking his GID, and in Section 5 we will discuss an extension which replaces Setup with an interactive protocol between the authorities.

In a multi-authority ABE system, we have many attribute authorities, and many users. There are also a set of system-wide public parameters available to everyone (either created by a trusted party, or by a distributed protocol between the authorities). A user can choose to go to an attribute authority, prove that it is entitled to some of the attributes handled by that authority, and request the corresponding decryption keys. The authority will run the attribute key generation algorithm, and return the result to the user. Any party can also choose to encrypt a message, in which case he uses the public parameters together with an attribute set of his choice to form the ciphertext. Any user who has decryption keys corresponding to an appropriate attribute set can use them for decryption.

In what follows, we use GID to denote the global identity of a user and $\mathbb{A}$ to denote a set of attributes. We use $\mathbb{A}^u$ and $\mathbb{A}^C$ to denote the attribute set of a user and that specified by a ciphertext respectively. We assume all the attribute sets can be partitioned into N disjoint sets, handled by the N attribute authorities, and we use a subscript k to denote the attributes handled by the authority k.

DEFINITION 5. *An N-authority ABE scheme consists of four algorithms:*

1. *via $(params, \{(apk_k, ask_k)\}_{k\in\{1,\ldots N\}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, N)$ the randomized key generation algorithm takes a security parameter $\lambda \in \mathbb{N}$ and the number of authorities $N \in \mathbb{N}$, and outputs the system parameters params and N public/private key pairs (apk_k, ask_k), one for each attribute authority $k \in \{1, \ldots N\}$. The threshold values $\{d_k\}_{k\in\{1,\ldots N\}}$ for each authority are also included in params. For simplicity, we assume params and $\{apk_k\}_{k\in\{1,\ldots N\}}$ are the implicit inputs of the rest of the algorithms.*

2. *via $usk_k[\mathsf{GID}, \mathbb{A}_k] \xleftarrow{\$} \mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k)$ the attribute authority k uses its secret key ask_k to output a decryption key corresponding to the attribute set $\mathbb{A}_k$ for the user with identity GID.*

3. *via $C \xleftarrow{\$} \mathsf{Enc}(\{\mathbb{A}_k\}_{k\in\{1,\ldots N\}}, m)$ a sender encrypts a message m for the set of attributes $\{\mathbb{A}_k\}$, resulting in a ciphertext C, where $\mathbb{A}_k$ denotes a subset of the attribute domain of the authority k.*

4. *via $m \leftarrow \mathsf{Dec}(\{usk_k[\mathsf{GID}, \mathbb{A}_k]\}_{k\in\{1,\ldots N\}}, C)$ a user GID who possesses a sufficient set of decryption keys $\{usk_k[\mathsf{GID}, \mathbb{A}_k]\}$ from each authority k decrypts C to recover m.*

DEFINITION 6. *An N-authority ABE scheme satisfies the consistency property if for all $\lambda, N \in \mathbb{N}$, all identities GID*

and all messages m, for all $\{\mathbb{A}_k^u\}$ and $\{\mathbb{A}_k^C\}$ such that $|\mathbb{A}_k^C \cap \mathbb{A}_k^u| > d_k$ for all authorities $k \in \{1,\ldots,N\}$,

$$\Pr[params \leftarrow \mathsf{Setup}(1^\lambda, N); C \xleftarrow{\$} \mathsf{Enc}(\{\mathbb{A}_k^C\}_{k\in\{1,\ldots N\}}, m); \mathsf{Dec}(\{\mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k^u)\}_{k\in\{1,\ldots N\}}, C) = m] = 1$$

, where the probability is taken over the random coins of all the algorithms in the expressions above.

DEFINITION 7. *An N-authority ABE scheme is (t, n, ϵ)-secure against selective-attribute attack if all t-time adversaries $\mathcal{A}$ compromising at most n authorities have advantage at most ϵ in making the game below return 1.*

Experiment $\mathbf{Exp}^{\mathrm{saa}}_{N-\mathcal{ABE},\mathcal{A}}(\lambda)$

$(\mathbb{A}^C = \{\mathbb{A}_1^C, \ldots, \mathbb{A}_N^C\}, \mathbb{K}_{corr} \subset [1, N]\}) \leftarrow \mathcal{A}$;
if $|\mathbb{K}_{corr}| > n$ *then return 0;*
$\{\mathbb{U}_k\}_{k\notin\mathbb{K}_{corr}} \leftarrow \emptyset$;
$(params, \{(apk_k, ask_k)\}_{k\in\{1,\ldots N\}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, N)$;
$(m_0^*, m_1^*, st) \xleftarrow{\$} \mathcal{A}^{\mathsf{AKeyGenO}(\cdot,\cdot,\cdot)}(\text{`find'}, params, \{apk_k\}_{k\in\{1,\ldots N\}}, \{ask_k\}_{k\in\mathbb{K}_{corr}})$;
$b \xleftarrow{\$} \{0,1\}$; $C^* \xleftarrow{\$} \mathsf{Enc}(\mathbb{A}^C, m_b^*)$;
$b' \xleftarrow{\$} \mathcal{A}^{\mathsf{AKeyGenO}(\cdot,\cdot,\cdot)}(\text{`guess'}, C^*, st)$;
if $b \neq b'$ *then return 0 else return 1;*

where st is state information, and the attribute-key generation oracle $\mathsf{AKeyGenO}(\mathsf{GID}, \mathbb{A}_k^u, k)$ *is defined as:*

if $(k \in \mathbb{K}_{corr})$ *return* $\perp$;
if $(\exists \mathbb{A}_k^{u\prime}$ *s.t.* $(\mathsf{GID}, \mathbb{A}_k^{u\prime}) \in \mathbb{U}_k)$ *return* $\perp$;[4]
if $(|\mathbb{A}_k^u \cap \mathbb{A}_k^C| \geq d_k)$
$\wedge \{\forall j \neq k, [(j \in \mathbb{K}_{corr})$
$\vee(\exists \mathbb{A}_j^u$ *s.t.* $((\mathsf{GID}, \mathbb{A}_j^u) \in \mathbb{U}_j \wedge |\mathbb{A}_j^u \cap \mathbb{A}_j^C| \geq d_j))]\}$
return $\perp$;
$\mathbb{U}_k \leftarrow \mathbb{U}_k \cup (\mathsf{GID}, \mathbb{A}_k^u)$; *return* $\mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k^u)$.

4. AUTHORITY-UNLINKABLE ABE

As mentioned before, a multi-authority ABE system which requires a user to present his unique identifier to every authority would have severe privacy shortcomings. In particular, it will be trivial for the various authorities to combine their data and assemble a complete picture of all of a user's attributes in all domains. To avoid this we look to related work on anonymous credentials [3, 4]. We will treat the GID as the user's secret key. Then the user can form different pseudonyms based on this GID to use when interacting with different authorities. When the user wishes to obtain decryption keys for certain attributes associated with this authority, he performs an interactive protocol with the authority. As a result of this protocol, he gets decryption keys tied to the GID that corresponds to his pseudonym. These can then be combined with decryption keys obtained from other authorities using other pseudonyms for the same GID. However, from the authorities' point of view the GID is completely hidden. In fact it is even infeasible for two authorities to tell that they are talking to the same user.

[4]As in all ABE schemes to date, users are not allowed to simply add attributes to their decryption key set. Instead, a user who wants to update his attribute set must receive an entirely new set of keys. In the multi-authority case (see e.g. [5]), this means that a user cannot simply return to an authority with the same GID – he must obtain new keys from all authorities.

4.1 Framework and Security Requirements

Our definition of an authority-unlinkable ABE scheme extends the definition in Section 2.2 by adding an interactive protocol to allow the user to obtain a decryption key from the authority without revealing his GID.

DEFINITION 8. *An N-authority-unlinkable ABE scheme is an N-authority ABE scheme with three extra algorithms (params and $\{apk_k\}_{k\in\{1,\ldots N\}}$ are omitted from the input):*

1. $(nym, aux) \xleftarrow{\$} \mathsf{FormNym}(\mathsf{GID})$ *probabilistically outputs a pseudonym for identity* GID*, and some auxiliary information aux.*

2. $\mathsf{Obtain}(apk_k, \mathsf{GID}, \mathbb{A}_k, nym, aux) \leftrightarrow \mathsf{Issue}(ask_k, \mathbb{A}_k, nym)$ *are two interactive algorithms which execute a user secret key issuing protocol between a user and the attribute authority k. The user takes as input the public key apk_k of the attribute authority k, an attribute set $\mathbb{A}_k$, an identity* GID*, and the corresponding pseudonym nym with auxiliary information aux, and gets what $\mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k)$ outputs, i.e. a decryption key for identity* GID *corresponding to the attribute set $\mathbb{A}_k$. The attribute authority gets the secret key ask_k, the set of attributes $\mathbb{A}_k$ and the pseudonym nym as input, and gets nothing as output.*

with the following properties

1. $(nym, aux) \xleftarrow{\$} \mathsf{FormNym}(\mathsf{GID})$ *produces a commitment nym to the user's GID with randomness aux,*

2. $\mathsf{Obtain} \leftrightarrow \mathsf{Issue}$ *form a secure two party computation (2PC) protocol for the following functionality F, where $(\{(apk_k, ask_k)\}_{k\in\{1,\ldots N\}})$ is as output by $\mathsf{Setup}(1^\lambda, N)$:*

   F takes as public input the authority's public key apk_k, the user's pseudonym nym, and the attribute set $\mathbb{A}_k$. It also receives as secret input the user's identity GID *and the corresponding aux, and the authority's secret key ask_k. It outputs the result of $\mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k)$ to the user.*

4.2 Generic Anonymous Key Issuing Protocol

Here we present a "generic" protocol such that a user with a private value $u \in \mathbb{Z}_q$ and an authority with private keys $\alpha, \beta, \gamma \in \mathbb{Z}_q$ can jointly compute the value $(h^\alpha g^{1/(\beta+u)})^\gamma$ for commonly known $g, h \in \mathbb{G}$[5]. Only the user gets this output, and all other information is hidden.

The roles of each private value will be apparent when this protocol is used as the anonymous key issuing protocol for the ABE system to be presented in Section 5. The basic intuition is that the structure of the final value $(h^\alpha g^{1/(\beta+u)})^\gamma$ resembles a product of h^α, which corresponds to something related to the private key of an authority, and a randomizer computed as $PRF_\beta(u)$, where β is the secret seed for Dodis-Yampolskiy PRF [7], and u is the GID of the user. [6] γ corresponds to some secret related to an attribute.

[5]We also require that the discrete logarithm between g and h be unknown to any corrupt user.

[6] In order for this to be a valid PRF, we need u to be chosen from some predefined polynomial-sized domain. Alternatively, we can choose $u = H(\mathsf{GID})$ for hash function H, and the result will be secure in the random oracle model.

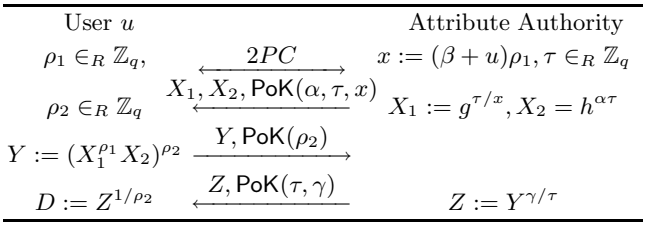

Figure 1: Our anonymous ABE key issuing protocol

Figure 1 shows our protocol for anonymous key issuing. In each step, PoK represents a proof of knowledge of the secret values used in the computation. For simplicity we have omitted the statement being proved. Here the first step denotes a 2PC protocol which takes (u, ρ_1) from the user and β from the authority and returns $x := (\beta + u)\rho_1 \mod q$ to the authority. This can be done via a general 2PC protocol for a simple arithmetic computation. Alternatively, we can do this more efficiently using the construction in [1]. The necessary proofs of knowledge (PoK) for the above statements can be efficiently realized, e.g. via a Schnorr protocol.

THEOREM 1. *The above protocol is a secure 2PC protocol for computing $(h^\alpha g^{1/(\beta+u)})^\gamma$, assuming that the underlying arithmetic 2PC and zero knowledge proofs are secure, and (for security against corrupt user) that DDH is hard.*

PROOF. To see note that $Z^{1/\rho_2} = Y^{\gamma/(\tau\rho_2)} = (X_1^{\rho_1\gamma/\tau} \cdot X_2^{\gamma/\tau}) = (h^\alpha g^{1/(\beta+u)})^\gamma$. To show security we consider the cases of corrupt issuer and corrupt user below.

Corrupt issuer.

For a corrupt issuer, our simulator proceeds as follows:

Sim_U First, it will run the arithmetic 2PC simulator for computation of $(\beta+u)\rho_1$. This 2PC will extract β from the issuer and expect to be provided with $x = \rho_1(\beta+u) \mod q$. We will choose a random value $x \in_R \mathbb{Z}_q$, and give it to the arithmetic 2PC simulator. Note that this is correctly distributed, since for any x, β, u, there is some ρ_1 such that $x = \rho_1(\beta + u) \mod q$. Next, our simulator will receive X_1, X_2 from the adversary, and two corresponding zero knowledge proofs. We will use the extractor for the proof system to extract α. We will choose a random $Y \in_R \mathbb{G}$ and return it. (Again, this will be distributed exactly as in a real execution.) Finally, we will receive Z from the adversary, and use the extractor to extract γ from the corresponding proof. We will give α, β, γ to the trusted party, and receive $(h^\alpha g^{1/(\beta+u)})^\gamma$, which will be the user's private output.

Consider a hybrid simulator Hyb_U that takes as input the user's identifier u. It first runs the arithmetic 2PC simulator for the computation of x (with the correct output value according to u), and then completes the protocol as the honest user would. This is clearly indistinguishable from the real user's protocol by the security of the arithmetic 2PC.

Now, assuming that the proof of knowledge scheme is secure, Hyb_U should be indistinguishable from the above simulator Sim_U. This is because the values x, Y used by Sim_U will be distributed identically to those in Hyb_U. (Since ρ_1, ρ_2 are chosen at random in the real protocol, x will be distributed uniformly over $\mathbb{Z}_q$, and Y will be distributed uniformly over $\mathbb{G}$ in the real protocol as in the simulated protocol.) Thus, interaction with our simulator is indistinguishable from interaction with an honest user.

Corrupt user.

For a corrupt user, our simulator proceeds as follows:

Sim_I First, it will run the arithmetic 2PC simulator for computation for $(\beta + u)\rho_1$ (in the process it will extract u). Next the simulator will choose random values $X_1, X_2 \in_R \mathbb{G}$, and send them to the user. It will receive Y from the user, and extract ρ_2 from the corresponding proof. Then it will send u to the trusted party and receive $D = (h^\alpha g^{1/(\beta+u)})^\gamma$. Finally, it will compute $Z = D^{\rho_2}$ and send it to the user.

Consider a hybrid simulator Hyb_I that takes as input the issuer secrets α, β, γ. It will compute $x = (\beta + u)\rho_1$ using the arithmetic 2PC simulator. When the 2PC simulator provides u, ρ_1 and asks for output, it will correctly compute $x = \rho_1(\beta + u)$. Then it will complete the execution as in the real protocol. This protocol is clearly indistinguishable from the real protocol by the security of the arithmetic 2PC.

Next, we consider a second hybrid Hyb'_I which proceeds as in Hyb_I, but which uses the zero-knowledge simulator for all proofs of knowledge. This must be indistinguishable by the zero-knowledge property of the proof system. Now we need only show that this second hybrid is indistinguishable from the interaction with the above simulator.

Consider the following reduction from DDH: Given $g, A = g^a, B = g^b, C = g^c$, and we must decide whether $c = ab$ or $c \in_R \mathbb{Z}_q$. We set $h = A^\theta$, for $\theta \in_R \mathbb{Z}_q$. As described in Sim_I, we run the arithmetic 2PC simulator to compute $x = \rho_1(\beta + u)$, and to extract u. Then we compute $X_1 = B^{(1/x)}, X_2 = C^{\theta\alpha}$, and send them to the adversary, along with a simulated proof of knowledge. We receive Y and extract ρ_2 from the corresponding proof. Finally, we compute $Z = (g^{1/(\beta+u)} A^{\alpha\theta})^{\gamma\rho_2}$, and return it to the user.

Note that, assuming that the proofs of knowledge are secure, if $c = ab$, X_1, X_2, Z will be distributed correctly, and this will be indistinguishable from Hyb'_I. On the other hand, if c is random, then X_1, X_2 are just values chosen at random from $\mathbb{G}$, as in Sim_I. Thus, any adversary that can distinguish Hyb'_I from Sim_I will allow us to solve DDH. We conclude that under the DDH assumption, interaction with Sim_I is indistinguishable from interaction with a real authority.

Thus our construction is a secure 2PC protocol. □

5. PROPOSED MULTI-AUTHORITY ABE

5.1 Removing the Trusted Authority

We review the motivation behind the use of the CA, and show how to avoid it. To have a concrete discussion, we assume the following details of an ABE system. The master public key is $\hat{e}(g_1, g_2)^{msk}$ and the message m is encrypted by $\hat{e}(g_1, g_2)^{s \cdot msk} \cdot m$ where s is the randomness of the ciphertext.

Simple Secret Sharing Allows Collusion.

To allow for multiple attribute authorities, the first step is to distribute the master secret key msk across the different attribute authorities. However, care must be taken to prevent collusion attacks so that users A and B who each have

the appropriate attributes from one of two different authorities cannot combine their knowledge to decrypt something neither of them is entitled to.

Now let's look at what happens when we want to divide this msk among the authorities. Consider the two-authority case. Suppose we use a trivial additive sharing of the master secret key $y_1 + y_2 = msk$ where one authority uses y_1 and the other uses y_2, and a scheme where an honest user gets a decryption key based on g^{y_1} and g^{y_2} from the respective authorities. Then a user A with enough attributes from the first authority can recover $\hat{e}(g_1, g_2)^{y_1 s}$, and similarly, user B with enough attributes from the second authority can recover $\hat{e}(g_1, g_2)^{y_2 s}$. Even if neither alone has sufficient attributes from both authorities, together they will be able recover $\hat{e}(g_1, g_2)^{s \cdot msk}$ and hence the message m. Thus we cannot use a straightforward sharing of the master secret key between the authorities.

The basic idea is to use a different sharing for each user. But, since we do not want these authorities to communicate among themselves for *every* secret key request, how can they ensure that the values used for each user always sum to msk?

Using PRFs to make the Key "User-Specific".

The answer in [5] was to require that authorities compute shares deterministically, each using their own PRF, and then to have a separate CA, whose job was to ensure that the sharing would add up: it would know each authority's PRF seed as well as the msk, it would use this information generate the shares used for each user, and it would generate the appropriate final share. Specifically, for user GID, each authority k uses share $g^{PRF_k(\mathsf{GID})}$, and the CA gives to user GID the value $g^{msk - \sum_{k=1}^{N}(PRF_k(\mathsf{GID}))}$, where $PRF_k(\cdot)$ denotes a pseudorandom function using authority k's secret seed. A user GID with enough attributes from authority k can recover $\hat{e}(g_1, g_2)^{s \cdot PRF_k(\mathsf{GID})}$ from the ciphertext. Then this can be combined with the "matching" value obtained from the CA and some component in the ciphertext to recover the session key $\hat{e}(g_1, g_2)^{s \cdot msk}$.

Ideas behind Our Proposal.

The beauty of a PRF family is that no polynomial-time adversary can distinguish (with significant advantage) between a randomly chosen function and a truly random function (in contrast with a degree m polynomial used in [10]). The idea here (suggested by Waters) idea is to eliminate the need for the CA by using a set of PRFs whose output values on any particular input always sum to zero. Each pair of authorities (j, k) shares a secret PRF seed $seed_{jk}$ (again, this sharing is done once and for all at the initial setup stage). This means there are $O(N^2)$ PRFs to be used in total. The final "random-looking" $F_k(\mathsf{GID})$ used by each authority is a linear combination of $N - 1$ basic PRFs. More specifically, it is the summation of all of these PRFs, each weighted by either 1 or -1. An appropriate choice of summation and subtraction makes all these PRF values cancel each other when $F_k(\mathsf{GID})$ for different k are added together. Informally, such a "sum-of-PRF" construction still looks pseudorandom to any adversary who knows less than $N - 2$ of a particular authority k's secret seeds $seed_{kj}$ (i.e. to any adversary controlling less than $N - 2$ other authorities). The final composite PRF is computed as $F_k(\mathsf{GID}) = \sum_{j<k} PRF_{jk}(\mathsf{GID}) - \sum_{j>k} PRF_{jk}(\mathsf{GID})$. This PRF construction is similar to the simplest construction in [11], where it is used to build a distributed key distribution center.

5.2 Adding the Anonymous Key Issuing

Before we can apply our oblivious key issuing protocol, we need to make suitable modifications to the scheme used in [5]. In particular, Chase assumes a PRF with range $\mathbb{Z}_q$ and generates decryption keys blinded by $g_1^{PRF_k(\mathsf{GID})}$. Here instead, we wish to use the modified Dodis-Yampolskiy PRF (DY-PRF), which has range $\mathbb{G}_1$. We observe that the PRF in the exponent used in [5] can be replaced by DY-PRF so that the values are instead blinded by $PRF_k(\mathsf{GID})$.With this modification and a little twist in the key structure, we can directly apply our key issuing protocol.

While our anonymous key issuing protocol is general enough to allow issuing keys of the form $(SK \cdot PRF(u))^{1/t_i}$ where SK is the secret key held by the key-issuing authority and u is a private value of the key-requesting user (e.g. GID), a straightforward adoption in the CA-less multi-authority ABE may result in a fairly inefficient system. To see this, recall that the keys are of the form $g_1^{p(i)/t_i} F_k(u)^{1/t_i}$ for polynomial p (this gives us "user-specific secret keys" which provides collusion-resistance) for each attribute $i \in \mathbb{A}^u$. This means that our key issuing protocol will be invoked $O(|\mathbb{A}^u|)$ times. On top of that, we require $O(N^2)$ copies of the underlying PRF in order to remove the trusted authority, which makes a total of $O(N^2|\mathbb{A}^u|)$ invocations of our key issuing protocol, an undesirably high price for preserving user privacy.

Instead, we will make the number of invocations independent of $|\mathbb{A}^u|$, by introducing extra randomness in the attribute key issuing process. We add $N - 1$ blinding factors R_{kj} to the secret value ask used to generate the attribute keys from authority k. The objective is to make the attribute part of the decryption key independent of user GID (but still different for each user) so that it can be generated without interaction with the user. We then use these R's to play the role of master secret key in the key issuing protocol, i.e. $g_1^{R_{kj}} \cdot PRF_{kj}(u)$ will be issued to user u for each value j. In the decryption process, the user will recover a function of $p(0)$, and can then use these values to remove the blinding.

To see how the new key structure ensures collusion resistance, when a user has enough attributes from a particular authority, he can recover a term with these R terms embedded (because of the way we define $p(0)$). The user secret key contains the term $g_1^{R_{kj}} \cdot PRF_{kj}(u)$, which is the only other information about these R terms. Intuitively, to get rid of these R terms will introduce the user-specific PRF value, which can only be cancelled out by the other PRF values for the *same* user, as hinted in the previous subsection.

5.3 Construction

Our final CA-less multi-authority anonymous ABE works as follows:

Setup.

The setup stage starts by the following initializations.

- (System Parameter) Given a security parameter λ and a public random string $\mathfrak{S} \in \{0,1\}^{\mathrm{poly}(\lambda)}$, the authorities generate an admissible bilinear group parameters $\langle \hat{e}(\cdot,\cdot), \psi(\cdot), q, g_1, g_2, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T \rangle$ from $\mathsf{BDH\_Gen}(1^\lambda; \mathfrak{S})$.

- (Collision-Resistant Hash Function (CRHF)) The authorities also generate from $\mathfrak{S}$ a CRHF $H : \{0,1\}^* \to \mathbb{Z}_q$, which takes the user's global identifier GID as an input. We denote the corresponding output by u.

Since the groups have prime order q and no hidden structure, this can safely be generated from public coins, so each authority can do this independently. The next stage is an interactive protocol. We assume the authorities have authenticated channels with one another.

- (Master Public/Secret Key) Each authority k picks $v_k \in_R \mathbb{Z}_q$ and sends $Y_k = \hat{e}(g_1, g_2)^{v_k}$ to the other authorities. They all individually compute $Y = \prod Y_k = \hat{e}(g_1, g_2)^{\sum_k v_k}$.
- (PRF Seed) Each pair of authorities engages in a 2-party key exchange such that each authority k shares with another authority j a seed $s_{kj} \in \mathbb{Z}_q$ which is only known to them and not to any other authority $i \notin \{j, k\}$. We define $s_{kj} = s_{jk}$.
- (PRF Base) Each authority k randomly picks $x_k \in \mathbb{Z}_q$ and computes $y_k = g_1^{x_k}$, which defines a pseudorandom function $PRF_{kj}(\cdot)$ that can only be computed by authority k and j. Define $PRF_{kj}(u) = g_1^{x_k x_j/(s_{kj}+u)}$, which can be computed by $y_k^{x_j/(s_{kj}+u)}$ or $y_j^{x_k/(s_{kj}+u)}$.

Each authority k also gives non-interactive proofs of knowledge of v_k and x_k.

The rest of the setup can be carried out by each authority autonomously:

- (Attribute Public/Private Key) Authority k proceeds as follows: for each attribute $i \in \{1, \ldots, n_k\}$ it picks $t_{k,i} \in \mathbb{Z}_q$ and computes $T_{k,i} = g_2^{t_{k,i}}$.

Each authority k stores

$$\langle x_k, \{s_{kj}\}_{j\in\{1,\ldots,N\}\setminus\{k\}}, \{t_{k,i}\}_{i\in[1,\ldots,n_k]}\}\rangle$$

securely as its private key. Finally the system parameters *params* are published as follows:

$$\langle Y = \hat{e}(g_1, g_2)^{\sum_k v_k}, \{y_k, \{T_{k,i} = g_2^{t_{k,i}}\}_{i\in[1,\ldots,n_k]}\}\}_{k\in[1,\ldots,N]}\}\rangle.$$

($\{y_k\}_{k\in[1,\ldots,N]}$ is only used by the authority.)

Key Issuing.

To get the key, user u executes the following with each authority k.

1. For $j \in \{1, \ldots, N\}\setminus\{k\}$, user u starts $N-1$ independent invocations of our anonymous key issuing protocol for $g = y_j^{x_k}$, $h = g_1$, $\alpha_k = \delta_{kj} R_{kj}$, $\beta_k = s_{kj}$ and $\gamma_k = \delta_{kj}$ where $R_{kj} \in \mathbb{Z}_q$ is randomly picked by authority k and $\delta_{kj} = 1$ if $k > j$ and -1 otherwise. As a result, user u obtains $D_{kj} = g_1^{R_{kj}} PRF_{kj}(u)$ for $k > j$ or $D_{kj} = g_1^{R_{kj}} / PRF_{kj}(u)$ for $k < j$.
2. Authority k randomly picks a degree d_k polynomial $p_k(\cdot)$ with $p_k(0) = v_k - \sum_{j\in\{1,\ldots,N\}\setminus\{k\}} R_{kj}$.
3. Authority k issues $S_{k,i} = g_1^{p(i)/t_{k,i}}$ for each eligible attribute i for the user.
4. User u computes $D_u = \prod_{(k,j)\in\{1,\ldots,N\}\times(\{1,\ldots,N\}\setminus\{k\})} D_{kj} = g_1^{R_u}$, where $R_u = \sum_{(k,j)\in\{1,\ldots,N\}\times(\{1,\ldots,N\}\setminus\{k\})} R_{kj}$. (Note that All PRF terms in the above project cancel each other out by the choice of δ_{kj}.)

Encryption.

To encrypt m for attribute set $\{\mathbb{A}_1^C, \ldots, \mathbb{A}_N^C\}$, pick $s \in_R \mathbb{Z}_q$, return $\langle E_0 = mY^s, E_1 = g_2^s, \{C_{k,i} = T_{k,i}^s\}_{i\in\mathbb{A}_k^C, \forall k\in[1\ldots N]}\rangle$. (Note that this is identical to the encryption algorithm in [5].

Decryption.

1. For each authority $k \in [1, \ldots N]$:
   (a) For any d_k attributes $i \in \mathbb{A}_k^C \cap \mathbb{A}_k^u$, pair up $S_{k,i}$ and $C_{k,i}$, i.e. compute $\hat{e}(S_{k,i}, C_{k,i}) = \hat{e}(g_1, g_2)^{sp_k(i)}$.
   (b) Interpolate all the values $\hat{e}(g_1, g_2)^{sp_k(i)}$ to get $P_k = \hat{e}(g_1, g_2)^{sp_k(0)} = \hat{e}(g_1, g_2)^{s(v_k - \sum_{j\neq k} R_{kj})}$.
2. Multiply P_k's together to get $Q = \hat{e}(g_1, g_2)^{s(\sum\{v_k\} - R_u)} = Y^s / \hat{e}(g_1^{R_u}, g_2^s)$.
3. Compute $\hat{e}(D_u, E_1) \cdot Q = \hat{e}(g_1^{R_u}, g_2^s) \cdot Q = Y^s$.
4. Recover m by E_0 / Y^s.

5.4 Confidentiality

THEOREM 2. *The construction of N-authority ABE described in Section 5.3 is a $(poly(t), N-2, \epsilon)$-secure multi-authority ABE under the assumption that no t-time algorithm can solve DBDH or q-DDHI with probability ϵ better than $1/2$, where q is polynomial in t.*

THEOREM 3. *The construction of N-authority ABE described in Section 5.3 is an authority unlinkable ABE under the XDH assumption, and the assumption that the underlying 2PC and zero knowledge proofs of knowledge are secure.*

The full proofs can be found in the full version. The intuition behind our reduction to DBDH problem is as follows: We are given a DBDH problem instance g_2^a, g_2^b, g_2^c, Z. We choose the secret key share of one honest attribute authority to be something that $\mathcal{S}$ cannot compute (ab). Then decrypting the challenge ciphertext would require computing a function of this value ($\hat{e}(g_1^{ab}, C_2)$). The original Sahai-Waters single authority scheme [13] had techniques for setting up a challenge ciphertext, and for issuing decryption keys for attribute sets that were insufficient to decrypt the challenge. What makes things challenging in the multi-authority case is that the adversary can request decryption keys for sufficient attribute sets from all but one of the authorities. And we do not know a priori which authority that one will be. Thus, we have to set up our parameters so that we can set any of our authorities as the one that corresponds to the uncomputable portion of the master key.

The way we do this in the proof is first to choose at random an authority k^* and form its parameters based on this uncomputable value. If it turns out that this is the authority from which the adversary requests insufficient attributes for user u, then we are all set, and we can simply reuse the Sahai-Waters techniques. If not, we use the fact that,

in our scheme, the authority does not give out keys only based on its share of the secret – it also incorporates some pseudorandom values based on the seeds that it shares with other authorities. Thus, we can pretend the challenge values are incorporated in this pseudorandomness. (If only honest parties know the seed of a PRF, then in our reduction we can replace it with any other "random-looking" function.) This of course means that at least one other authority must adjust its pseudorandomness to compensate, and we choose this authority to be one of the other honest authorities from which the adversary does not request sufficient attributes. (The security game guarantees that there must be at least one other such authority.)

Finally, recall that our anonymous key issuing protocol requires that the discrete logarithm between $g_1 = g^{x_j x_k}$ and $h = g_1$ be unknown. This condition is satisfied since a collusion of $N-2$ AAs cannot learn the discrete logarithm x_j if the authority j is outside the collusion group. (In this case the colluding parties will only see $y_j = g_1^{x_j}$ and a corresponding zero-knowledge proof of knowledge of x_j.)

5.5 Efficiency

The above construction requires that each authority store $N-1$ seeds and run $N-1$ invocations of our anonymous key issuing protocol for each user. The user in turn has to store $|\mathbb{A}_k^u| + 1$ values for each authority k. The main overhead is on the side of the authority, and even so, it seems a fairly small cost to pay in exchange for guaranteeing security when any $N-2$ out of N authorities are corrupted.

For initial setup, we do not require any explicit distributed key generation (DKG). Thanks to the non-interactive zero-knowledge proof, the number of communication rounds is quite minimal (2 rounds). When compared with the trusted CA approach, the load on the authorities is still somewhat higher in that they must participate in an initial setup phase and communicate with all other authorities. However, it seems unavoidable given that we have no party who is guaranteed to be trusted. Finding an approach that would avoid this limitation, while still providing the strong security guarantees that we consider, is a very interesting problem.

A detailed comparison among different multi-authority ABE proposals is given in Table 1. "Tolerance" refers to the maximum colluding set against which a system remains secure. "DKG instance" refers to the number of invocations of the distributed key generation protocol required among the AAs in the setup stage. "Ciphertext" refers to the ciphertext overhead, i.e. the ciphertext size minus the plaintext size. For a fair comparison with Lin *et al.*'s scheme [10], since our scheme is secure against polynomially-many users, we suggest m should be much larger than N, since it seems reasonable to assume there are more malicious users than malicious authorities. Our system performs better when $m > N-1$. The efficiency of our scheme compares favorably with that of previous multi-authority ABE schemes, even though we provide a stronger security guarantee.

6. EXTENSIONS

6.1 Supporting Large Universe

In our basic construction, the universe of attributes is constrained by the size of the public parameters (specifically $\{T_{k,i}\}$). This contrasts with the large universe model (first introduced in [13]), in which the universe of attributes is exponentially large, but public parameter size depends on a fixed maximum on the number of attributes allowed in a ciphertext. That approach also has the advantage that any arbitrary string can be used as an attribute via the use of a collision resistant hash function. A large universe construction was presented in [13] and a similar concept has been used in [8].

Properties	Chase [5]	Lin *et al.* [10]	Ours
Tolerance	0 CA	m users	$(N-2)$ AAs
DKG Instance	0	$m+2$	0
AA Key Size	$\|\mathbb{A}_k\|+1$	$\|\mathbb{A}_k\|+m+1$	$\|\mathbb{A}_k\|+N$
User Key Size	$\|\mathbb{A}^u\|+1$	$\|\mathbb{A}^u\|$	$\|\mathbb{A}^u\|+1$
Ciphertext	$\|\mathbb{A}^C\|+1$	$\|\mathbb{A}^C\|$	$\|\mathbb{A}^C\|+1$

Table 1: Comparisons of Different ABE Proposals

Our anonymous key issuing protocol and the removal of the central authority technique can also be applied to the multi-authority version of the large universe and complex access structure construction in [8]. We highlight five major distinctions of the large universe construction:

1. Functions $\{T_k(i)\} : \mathbb{Z}_q \to \mathbb{G}_1$ are used to replace the group elements $T_{k,i}$ for attribute i of each authority k. $T_k(i)$ is publicly computable.
2. $C_{k,i}$ in the ciphertext is changed from $T_{k,i}^s$ to $T_k(i)^s$.
3. $g_1^{p(i)/t_{k,i}}$ in the user secret key is replaced with $g_1^{p(i)} T_k(i)^r$.
4. Since the randomness r is introduced in the user secret key, g_2^r is also given to "cancel out" r in the decryption.
5. To decrypt a ciphertext, merely computing $\hat{e}(g_1^{p(0)} \cdot T_k(i)^r, g_2^s)$ results in $\hat{e}(g_1, g_2)^{sp(0)} \cdot \hat{e}(T_k(i)^r, g_2^s)$, so the later term should be cancelled out by $\hat{e}(T_k(i)^s, g_2^r)$.

While the proof of confidentiality in [8] relies critically on the construction of $T_k(i)$ in the simulation, the key idea of the multi-authority scheme in [5] is that $p(0)$ will be set as the PRF computed on the user's GID, and thus this technique is independent of how $T_k(i)$ is constructed. As hinted at in the intuition provided in the proof of our basic scheme, the crux in our proof for multi-authority ABE is about how to embed an unknown master secret key (which is related to the solution of the hard problem) by taking advantage of the pseudorandom values blinding the user secret key. We can show security by applying the same techniques as in section 5. (Details of the proof are deferred to the full version.)

6.2 Complex Access Structure

Another limitation of our basic construction as described in Section 5 is that it only supports simple d_k-out-of-n threshold policies, while Goyal *et al.*'s construction [8] supports a tree access structure. When we consider the tree as a circuit, the interior nodes consist of t-out-of-n gates for arbitrary values of t and n, and each leaf node is associated with an attribute and has value 1 if that attribute is present in a given ciphertext.

This tree is the key idea behind the complex access structure construction. A polynomial p_x is chosen for each node

x in the tree. These polynomials are chosen in a top-down manner, starting from the root node r, such that the degree of the polynomial p_x is one less than the threshold value t_x of that node. The value $p_r(0)$ at the root node depends on the $\mathsf{AKeyGen}$ algorithm. (In our case $p_r(0) = v_k - \sum_{j\in\{1,\ldots,N\}\setminus\{k\}} R_{kj}$.) For the other nodes x of the tree, $p_x(0)$ is defined to be $p_{\mathsf{parent}[x]}(\mathsf{index}[x])$ where $\mathsf{parent}[x]$ denotes the parent node of x and $\mathsf{index}[x]$ is merely a distinct number for each node at the same level. Using the same approach as in [5], it is not difficult to see that the same tree-based key-structure can be used in our schemes, simply by changing how the root key $p(0)$ is generated.

6.3 Variable Thresholds across Authorities

Our basic construction requires the user to have enough attributes from *every* authority, but we can easily let the encryptor leave out a certain subset of authorities by asking each authority to issue to every user a decryption key corresponding to d_k dummy attributes.[7]

Generalizing, each encryptor can reduce the threshold for a chosen set of authorities by adjusting the number of dummy variables included for those authorities accordingly. Suppose $dmax$ is the maximum threshold. If the encryptor wanted to require $d' < dmax$ of the attributes, he could encrypt with respect to $dmax - d'$ dummy attributes in additional to the usual attributes. This does not incur heavy penalty in the efficiency of the system, especially when we have a large universe construction to host the dummy attributes.

The use of dummy variables for flexible threshold policy in the *ciphertext* was suggested in [5]. We note that our scheme also allows flexibility in setting the threshold policy in the *key*, simply due to the fact that our scheme supports different threshold values d_k for different users.

7. CONCLUSION

It is unrealistic to assume there is a single authority which can monitor every single attribute of all users. Multi-authority attribute-based encryption enables a more realistic deployment of attribute-based access control, such that different authorities are responsible for issuing different sets of attributes. The original solution by Chase employs a trusted central authority and the use of a global identifier for each user, which means the confidentiality depends critically on the security of the central authority and the user-privacy depends on the honest behavior of the attribute-authorities. We propose an attribute-based encryption scheme without the trusted authority, and an anonymous key issuing protocol which works for both existing schemes and for our new construction. We hope that our work gives a more practice-oriented attribute based encryption system.

Acknowledgement

We thank Brent Waters for suggesting the sum of PRFs construction.

[7]In contrast to a normal threshold cryptosystem, here the threshold will only be reduced if the encryptor chooses to do so (by including dummy attributes in the ciphertext attribute set). Thus, each ciphertext may have a different threshold.

8. REFERENCES

[1] Mira Belenkiy, Jan Camenisch, Melissa Chase, Markulf Kohlweiss, Anna Lysyanskaya, and Hovav Shacham. Randomizable Proofs and Delegatable Anonymous Credentials. In *CRYPTO*, LNCS. Springer, 2009. To appear.

[2] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-Policy Attribute-Based Encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334. IEEE Computer Society, 2007.

[3] Stefan Brands. *Rethinking Public Key Infrastructure and Digital Certificates – Building in Privacy*. PhD thesis, Eindhoven Inst. of Tech. 1999.

[4] Jan Camenisch and Anna Lysyanskaya. Efficient Non-transferable Anonymous Multi-show Credential System with Optional Anonymity Revocation. In *EUROCRYPT 2001*, volume 2045 of *LNCS*, pages 93–118. Springer Verlag, 2001.

[5] Melissa Chase. Multi-authority Attribute Based Encryption. In *TCC*, volume 4392 of *LNCS*, pages 515–534. Springer, 2007.

[6] Sherman S.M. Chow. Removing Escrow from Identity-Based Encryption. In *Public Key Cryptography*, volume 5443 of *LNCS*, pages 256–276. Springer, 2009.

[7] Yevgeniy Dodis and Aleksandr Yampolskiy. A Verifiable Random Function with Short Proofs and Keys. In *Public Key Cryptography*, volume 3386 of *LNCS*, pages 416–431. Springer, 2005.

[8] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-Based Encryption for Fine-Grained Access Control of Encrypted Data. In *Computer and Communications Security*, pages 89–98. ACM, 2006.

[9] Stanislaw Jarecki and Xiaomin Liu. Efficient Oblivious Pseudorandom Function with Applications to Adaptive OT and Secure Computation of Set Intersection. In *TCC*, pages 577–594. Springer, 2009.

[10] Huang Lin, Zhenfu Cao, Xiaohui Liang, and Jun Shao. Secure Threshold Multi Authority Attribute Based Encryption without a Central Authority. In *INDOCRYPT*, volume 5365 of *LNCS*, pages 426–436. Springer, 2008.

[11] Moni Naor, Benny Pinkas, and Omer Reingold. Distributed Pseudo-random Functions and KDCs. In *EUROCRYPT*, volume 1592 of *LNCS*, pages 327–346. Springer, 1999.

[12] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-Based Encryption with Non-Monotonic Access Structures. In *Computer and Communications Security*, pages 195–203, 2007.

[13] Amit Sahai and Brent Waters. Fuzzy Identity-Based Encryption. In *EUROCRYPT*, volume 3494 of *LNCS*, pages 457–473. Springer, 2005.

[14] Adi Shamir. Identity-Based Cryptosystems and Signature Schemes. In *CRYPTO*, pages 47–53. Springer, 1984.

[15] Brent Waters. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. Cryptology ePrint 2008/290.