Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption

Yannis Rouselakis[1] and Brent Waters[2]

[1] The University of Texas at Austin
yannis.rouselakis@gmail.com
[2] The University of Texas at Austin
bwaters@cs.utexas.edu

Abstract. We propose an efficient large-universe multi-authority ciphertext-policy attribute-based encryption system. In a large-universe ABE scheme, any string can be used as an attribute of the system, and these attributes are not necessarily enumerated during setup. In a multi-authority ABE scheme, there is no central authority that distributes the keys to users. Instead, there are several authorities, each of which is responsible for the authorized key distribution of a specific set of attributes. Prior to our work, several schemes have been presented that satisfy one of these two properties but not both. Our construction achieves maximum versatility by allowing multiple authorities to control the key distribution for an exponential number of attributes. In addition, the ciphertext policies of our system are sufficiently expressive and overcome the restriction that "each attribute is used only once" that constrained previous constructions. Besides versatility, another goal of our work is to increase efficiency and practicality. As a result, we use the significantly faster prime order bilinear groups rather than composite order groups. The construction is non-adaptively secure in the random oracle model under a non-interactive q-type assumption, similar to one used in prior works. Our work extends existing "program-and-cancel" techniques to prove security and introduces two new techniques of independent interest for other ABE constructions. We provide an implementation and some benchmarks of our construction in Charm, a programming framework developed for rapid prototyping of cryptographic primitives.

Keywords: Attribute-Based Encryption, Multi-Authority, Large Universe, Unbounded, q-Type Assumption, Charm, Implementations

1 Introduction

Public key cryptography allows a sender to encrypt data such that it can only be decrypted by the owner of the corresponding secret key. Encrypting in this manner is useful for when the sender knows the specific identity of the recipient at the time the data is encrypted. However, in many scenarios the data owner might not know the exact recipients that he wishes to target, but instead wish to express sharing of the data in terms of a policy formulated over the credentials or attributes of different users. Here the sender might not even know who the exact recipients are that match this policy or someone might acquire the exact credentials well after the data was encrypted and stored.

Sahai and Waters [41] put forth a different vision of encryption call Attribute-Based Encryption (ABE). In a (Ciphertext-Policy) ABE scheme the encryption algorithm takes as input the public parameters as issued by some authority as well as a boolean formula over a set of attributes. Users in the system will be issued private keys by the authority that are associated with a set of attributes. A user is able to decrypt a ciphertext if the attributes of her private key satisfy the boolean formula associated with the ciphertext.

The typical scenario presented for ABE is where a single authority issues all private keys. This works well in the setting where data is managed within one organization or trust domain. However, there are many scenarios when one will wish to describe a policy that spans multiple trust domains. For example, U.S. military and defense are several organizations that wish to manage the distribution of their own credentials. If we wished to write an access policy that referenced credentials from both of them using standard ABE, we would require one organization ceding control to another or a third party. To address this issue multi-authority or decentralized [14] ABE systems were introduced where multiple parties could play the role of an authority. Initial attempts at such systems [14, 15] sacrificed a significant amount of expressiveness compared to analogs in the one authority setting. Fairly recently, though Lewko and Waters [27] provided a system that roughly matched the expressiveness. In their system a policy could be expressed as any monotonic boolean formula[3] over attributes that can be issued by any authority which publishes a public key. Their main construction technique is to use a hash over a global identifier. Upon decryption this extra component serves as a "blinding factor" that only disappears if the ciphertext is satisfied.

While the expressiveness, of the Lewko-Waters distributed ABE system is relatively strong, there are three major aspects that impact its practical performance compared to single authority systems. First, the construction is set in a group of composite order N where N is the product of three primes. This alone can make certain operations such as exponentiation over an order of magnitude slower (see App. D). Second, each authority in the system can "natively" support only a single attribute. If in practice we would like one party to act as an authority for up to c attributes, the party would have to create a public key consisting of c native public keys (thus blowing up the size by a factor of c). Furthermore, this only works if the attributes managed by that party can be enumerated ahead of time. This means that the attribute universe is restricted to polynomial size. Finally, the system has the native property that each authority can be used only once in each formula. In practice, if we want to get around this and let it be used up to d times we can apply a simple encoding technique due to Lewko et. al. [26].[4] This encoding however comes at the cost of blowing up both the parameters of the authority and the private key components issued by the authority by a factor of d. To make things concrete suppose that we wanted a system with an authority that managed 20 attributes each of which appeared at most 10 times in the any formula. Then the published parameters for just that one authority would need to blowup by a factor of 200 (compared to a contemporary single use CP-ABE system [11, 49]) just to deal with the encoding overhead.

We will construct and implement a new decentralized ABE cryptosystem that aims to get performance close to existing single authority constructions. Our approach is to use the LW construction as a substrate from which we make two significant changes to improve performance. First, we take the existing construction and pare it down to the prime order setting. This will make it inherently faster, but incompatible with the Dual System Encryption [48] proof techniques used before. (Note we do not simulate subspaces in prime order groups [18, 36, 25] which itself has additional overhead.) Second, we add an additional piece to each ciphertext and private key component which allows us to use *any* string as an attribute — thus addressing the problem of an authority only supporting a single attribute and the small universe restriction. At the same time, the second change allows the system to utilize each attribute as many times as needed in each policy.

[3] Actually, their system is more general in that it allows for monotone span programs.

[4] The one use restriction is needed to make the security proof of Lewko and Waters go through, if the one use restriction were violated there is neither a known attack nor a security proof.

With these changes we must prove security of our scheme. As mentioned with the removal of subgroups, the Dual System Encryption methodology is no longer available. We will create a proof of security in a static model of security where both the challenge ciphertexts and key queries are issued before the parameters are published. We needed the keys' queries to be non-adaptive, a property which has not been used in prior work, because the private key for a single user is issued in a piecemeal fashion. Each piece corresponds to a different authority, while in the single authority setting private key requests are naturally atomic. We extend the existing "program and cancel" techniques from two large universe constructions presented in [40] in order to adapt to the multi-authority setting and introduce two new ones. The trade-offs for our performance improvements are the use of the static model and an assumption whose size depends on the complexity of the challenge ciphertext policy.

To demonstrate the abilities of our system we implemented our algorithms in Charm and we provide timing results and comparisons to existing single-authority schemes.

1.1 Related Work

Attribute-Based Encryption was introduced by Sahai and Waters [41]. In this work, the key-policy and ciphertext-policy notions were defined and many selectively secure constructions followed [6, 16, 21, 37, 39, 49]. Most of them work for non monotonic access structures with the exception of the schemes by Ostrovsky, Sahai, and Waters [37], who showed how to realize negation by incorporating specific revocation schemes into the GPSW construction. Fully secure constructions in the standard model were first provided by Okamoto and Takashima [36] and Lewko, Okamoto, Sahai, Takashima, and Waters [26]. The first large universe KP-ABE construction in the standard model was given in [28] (composite order groups). Two large universe constructions in prime order groups were presented in [40] and both techniques, *layering* and *individual randomness*, from that paper are extended and utilized in our current cosntruction. Okamoto and Takashima initiated the dual pairing vector space framework in various works [34, 35, 36], which lead to the first large universe KP-ABE construction in prime order group groups by Lewko [25]. Parameterized (non static) assumptions were introduced in [7] and used in several subsequent works [20, 49]. The problem of an environment with multiple central authorities in ABE was considered in [14, 15, 27], while several authors have presented schemes that do not address the problem of collusion resistance [45, 30, 12, 2, 3, 4].

We note that several techniques in ABE schemes have roots in Identity-Based Encryption [42, 8, 17, 7, 47, 20, 9]. Finally, we mention here the related concept of Predicate Encryption introduced by Katz, Sahai, and Waters [23] and further refined in [44, 43, 35, 26, 36, 10].

2 Preliminaries

2.1 Notation

For $n \in \mathbb{N}$, we define $[n] = \{1, 2, \ldots, n\}$. Also, for $n_1, n_2, \ldots, n_k \in \mathbb{N}$: $[n_1, n_2, \ldots, n_k] = [n_1] \times [n_2] \times \ldots \times [n_m]$. By $\{X_i\}_{i \in [n]}$ we denote a sequence of elements $X_1, X_2, \ldots, X_n$.

When S is a set, we denote by $s \xleftarrow{R} S$ the fact that the variable s is picked uniformly at random from S. We write $s_1, s_2, \ldots, s_n \xleftarrow{R} S$ as shorthand for $s_1 \xleftarrow{R} S, s_2 \xleftarrow{R} S, \ldots, s_n \xleftarrow{R} S$.

The set of matrices of size $m \times n$ with elements in $\mathbb{Z}_p$ is denoted by $\mathbb{Z}_p^{m \times n}$. Special subsets are the set of row vectors of length n: $\mathbb{Z}_p^{1 \times n}$, and column vectors of length n: $\mathbb{Z}_p^{n \times 1}$. We denote by $\langle \boldsymbol{v}, \boldsymbol{w} \rangle$

the inner product of vector $\boldsymbol{v}$ with $\boldsymbol{w}$, where each vector can either be a row or a column vector. Finally, the operation $(\cdot)^\top$ denotes the transpose vector/matrix.

2.2 Access Structures and Linear Secret - Sharing Schemes

In this subsection, we present the formal definitions of access structures and linear secret-sharing schemes introduced in [5], adapted to match our setting.

Definition 1 (Access Structures [5]) *Let $\mathcal{U}$ be the attribute universe. An* access structure on $\mathcal{U}$ *is a collection $\mathbb{A}$ of non-empty sets of attributes, i.e. $\mathbb{A} \subseteq 2^{\mathcal{U}} \setminus \{\}$. The sets in $\mathbb{A}$ are called the* authorized sets *and the sets not in $\mathbb{A}$ are called the* unauthorized sets.

Additionally, an access structure is called monotone *if $\forall B, C \in \mathbb{A}$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \in \mathbb{A}$.*

In our construction, we only consider monotone access structures, which means that as a user acquires more attributes, he will not lose his possible decryption privileges. General access structures in large universe ABE can be realized by splitting the attribute universe in half and treating the attributes of one half as the negated versions of the attributes in the other half [22].

Definition 2 (Linear Secret-Sharing Schemes) *Let p be a prime and $\mathcal{U}$ the attribute universe. A secret-sharing scheme Π with domain of secrets $\mathbb{Z}_p$ realizing access structures on $\mathcal{U}$ is* linear over $\mathbb{Z}_p$ *if*

1. *The shares of a secret $z \in \mathbb{Z}_p$ for each attribute form a vector over $\mathbb{Z}_p$.*
2. *For each access structure $\mathbb{A}$ on $\mathcal{U}$, there exists a matrix $A \in \mathbb{Z}_p^{\ell \times n}$, called the share-generating matrix, and a function δ, that labels the rows of A with attributes from $\mathcal{U}$, i.e. $\delta : [\ell] \to \mathcal{U}$, which satisfy the following:*
   During the generation of the shares, we consider the column vector $\boldsymbol{v} = (z, r_2, \ldots, r_n)^\perp$, where r_2, ..., $r_n \xleftarrow{R} \mathbb{Z}_p$. Then the vector of ℓ shares of the secret z according to Π is equal to $\boldsymbol{\lambda} = A\boldsymbol{v} \in \mathbb{Z}_p^{\ell \times 1}$. The share λ_j with $j \in [\ell]$ "belongs" to attribute $\delta(j)$.
   We will be referring to the pair (A, δ) as the policy of the access structure $\mathbb{A}$.

According to [5], each secret-sharing scheme (not only the linear ones) should satisfy the *reconstruction requirement* (each authorized set can reconstruct the secret) and the *security requirement* (any unauthorized set cannot reveal any partial information about the secret). More concretely, let S denote an authorized set of attributes and let I be the set of rows whose labels are in S. There exist constants $\{c_i\}_{i \in I}$ in $\mathbb{Z}_p$ such that for any valid shares $\{\lambda_i = (A\boldsymbol{v})_i\}_{i \in I}$ of a secret z according to Π, it is true that: $\sum_{i \in I} c_i \lambda_i = z$, or equivalently $\sum_{i \in I} c_i \boldsymbol{A}_i = (1, 0, \ldots, 0)$, where $\boldsymbol{A}_i$ is the i-th row of A.

On the other hand, for unauthorized sets S' no such constants exist. In this case, it is also true that if I' is the set of rows whose labels are in S', there exists a vector $\boldsymbol{d} \in \mathbb{Z}_p^{1 \times n}$, such that its first component $d_1 = 1$ and $\langle \boldsymbol{A}_i, \boldsymbol{d} \rangle = 0$ for all $i \in I'$.

Finally, we note that if the access structure is encoded as a monotonic Boolean formula over attributes there is a generic algorithm that generates the corresponding access policy in polynomial time [5, 27].

Multi-Authority Attributes In the multi-authority setting, each attribute is controlled by a specific authority $\theta \in \mathcal{U}_\Theta$, where $\mathcal{U}_\Theta$ is the set (universe) of all authorities. We assume there is a publicly computable function $\mathsf{T} : \mathcal{U} \to \mathcal{U}_\Theta$ that maps each attribute to a unique authority. Using this mapping a second labeling of rows is defined in a policy (A, δ), which maps rows to attributes via the function $\rho(\cdot) \stackrel{\mathsf{def}}{=} \mathsf{T}(\delta(\cdot))$.

In our implementation, both the attribute id's and the authority id's consist of case-sensitive alphanumeric strings. The full attributes' names are of the form "[attribute–id]@[authority–id]" and the mapping T just extracts the part after the @ of the attribute string.

2.3 Bilinear Groups and Complexity Assumption

For the following we assume familiarity of the reader with bilinear groups of prime order (see App. B for more information). Our construction, the complexity assumption, and the security proof are all expressed in the simpler setting of symmetric groups and can be generically transformed to the asymmetric setting by substituting roughly half of the scheme's components with the respective $\mathbb{G}_2$ terms (i.e. with the same exponents). Our implementations are all written formally in the asymmetric setting, since this is what the Charm framework dictates, although 2 out of the 5 test runs were executed with super-singular symmetric groups (see Sec. 5).

For our security proof we will use a q-type assumption on prime order bilinear groups. It is a slightly modified version of the q-Decisional Parallel Bilinear Diffie-Hellman Exponent Assumption [49]. We will be referring to our assumption as q-DPBDHE2 for short. The assumption is defined as follows:

Choose a bilinear group $\mathbb{G}$ of order p according to the security parameter κ, which admits a non-degenerate bilinear mapping $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$. Pick $s, a, b_1, b_2, \ldots, b_q \stackrel{R}{\leftarrow} \mathbb{Z}_p$ and $R \stackrel{R}{\leftarrow} \mathbb{G}_T$. Let

$$D = \left(\mathbb{G}, p, e, g, g^s, \{g^{a^i}\}_{\substack{i \in [2q] \\ i \neq q+1}}, \{g^{b_j a^i}\}_{\substack{(i,j) \in [2q,q] \\ i \neq q+1}}, \{g^{s/b_i}\}_{i \in [q]}, \{g^{s a^i b_j / b_{j'}}\}_{\substack{(i,j,j') \in [q+1,q,q] \\ j \neq j'}} \right)$$

The assumption states that no polynomial-time distinguisher can distinguish the distribution $\left(D, e(g,g)^{s a^{q+1}}\right)$ from the distribution (D, R) with more than negligible advantage.

The only difference between the q-DPBDHE assumption in [49] and the above assumption is that in the latter the $\{g^{s a^i b_j / b_{j'}}\}$ terms go up to $i = q+1$ instead of q. The q-DPBDHE assumption was shown generically secure in [49] and following exactly the same proof path, one can prove that the q-DPBDHE2 assumption is also generically secure. Due to lack of space and the similarity to [49], the full proof is omitted.

The assumption is closely related to the two assumptions presented in [40]. Although incomparable to both of them, it contains fewer terms, hence it is relatively weaker. This comes in contrast to the fact that our multi-authority construction supports more features than the two ABE schemes of [40]. The reason of the apparent paradox is the use of the static security and random oracle model in this work versus selective security and standard model in [40].

3 Multi-Authority Ciphertext-Policy ABE

In this section we provide the necessary background on multi-authority CP-ABE schemes and the security definition for static security.

3.1 Algorithms

A multi-authority ciphertext-policy attribute-based encryption system consists of the following five probabilistic polynomial-time algorithms:

$\mathsf{GlobalSetup}(1^\kappa) \to \mathrm{GP}$**:** The global setup algorithm takes in the security parameter κ encoded in unary and outputs the public global parameters for the system. We require that descriptions of the attribute universe $\mathcal{U}$, the authority universe $\mathcal{U}_\Theta$, the global identifier universe $\mathcal{GID}$, and the mapping T are included in the global parameters.

$\mathsf{AuthSetup}(\mathrm{GP}, \theta) \to \{\mathrm{PK}_\theta, \mathrm{SK}_\theta\}$**:** The authority $\theta \in \mathcal{U}_\Theta$ calls the authority setup algorithm during its initialization with the global parameters GP as input and receives its public / secret key pair $\{\mathrm{PK}_\theta, \mathrm{SK}_\theta\}$.

$\mathsf{KeyGen}(\mathrm{GID}, \mathrm{SK}_\theta, u, \mathrm{GP}) \to \mathrm{SK}_{\mathrm{GID},u}$**:** The key generation algorithm takes in the global identifier GID of a user ($\mathrm{GID} \in \mathcal{GID}$), the secret key of an authority θ, an attribute u controlled by the authority θ, and the global parameters. It outputs a key for the identity - attribute pair (GID, u).[5]

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

$\mathsf{Decrypt}(\mathrm{CT}, \{\mathrm{SK}_{\mathrm{GID},u}\}, \mathrm{GP}) \to M$**:** The decryption algorithm takes in a ciphertext CT, the set of keys of a single user GID corresponding to different attributes u, and the global parameters. It outputs either the message M when the collection of attributes satisfies the access structure of the ciphertext, or decryption fails.

We require that all schemes satisfy the following correctness property:

Definition 1. *A multi-authority CP-ABE scheme is* correct *if for any* GP *generated by the global setup algorithm, for any set of keys* $\{\mathrm{PK}_\theta, \mathrm{SK}_\theta\}$ *generated by the authority setup algorithm, for any* CT *generated by the encryption algorithm using the relevant authorities' public keys on any message* M *and access structure* $\mathbb{A}$*, and for any set of keys* $\{\mathrm{K}_{\mathrm{GID},u}\}$ *generated by the key generation algorithm using the relevant authorities' secret keys for one user* GID *on any* $\mathbb{A}$-authorized *set of attributes, it is true that* $\mathsf{Decrypt}(\mathrm{CT}, \{\mathrm{SK}_{\mathrm{GID},u}\}, \mathrm{GP}) = M$*.*

3.2 Static Security

In this section we will define the static (or non-adaptive) security game between a challenger and an attacker. The difference between this security game and the adaptive one is that all queries done by the attacker are sent to the challenger immediately after seeing the public parameters. As usual, we also allow the attacker to corrupt a certain set of authorities that he can control. These authorities are chosen by the attacker after seeing the global parameters and remain the same until the end of the game.

The game consists of the following phases:

Global Setup: The challenger calls $\mathsf{GlobalSetup}(1^\kappa) \to \mathrm{GP}$ and gives the global parameters GP to the attacker.

Attacker's Queries: Then the attacker responds with:

[5] If a user wants a key that corresponds to multiple attributes from the same authority, the key generation algorithm is trivially extended to take in many attributes by running the "single attribute" version once for each attribute.

– A set $\mathcal{C}_\Theta \subseteq \mathcal{U}_\Theta$ of corrupt authorities and their respective public keys $\{\mathrm{PK}_\theta\}_{\theta\in\mathcal{C}_\Theta}$, which he might have created in a malicious way[6].
– A set $\mathcal{N}_\Theta \subseteq \mathcal{U}_\Theta$ of the non-corrupt authorities for which the adversary requests the public keys. Obviously, it should be disjoint from the set of corrupt authorities.
– A sequence $\mathcal{Q} = \{(\mathrm{GID}_i, S_i)\}_{i=1}^m$ of the secret key queries, where the global identities GID_i are distinct and $S_i \subseteq \mathcal{U}$ with $\mathsf{T}(S_i) \cap \mathcal{C}_\Theta = \emptyset$.
A pair (GID_i, S_i) in this sequence denotes that the attacker requests the secret keys for the user GID_i with attributes from the set S_i. That is, for every $u \in S_i$ the attacker gets a key $\mathrm{SK}_{\mathrm{GID}_i,u} \leftarrow \mathsf{KeyGen}(\mathrm{GID}_i, \mathrm{SK}_{\mathsf{T}(u)}, u, \mathrm{GP})$. According to the restriction $\mathsf{T}(S_i) \cap \mathcal{C}_\Theta = \emptyset$, none of these keys come from a corrupt authority.
– Two messages M_0, M_1 of equal length, and a challenge access structure $\mathbb{A}$ encoded in a suitable form. We require that for every $i \in [m]$ the set $S_i \cup \bigcup_{\theta\in\mathcal{C}_\Theta} \mathsf{T}^{-1}(\theta)$ is an unauthorized set of the access structure $\mathbb{A}$, where $\bigcup_{\theta\in\mathcal{C}_\Theta} \mathsf{T}^{-1}(\theta)$ is the set of all the attributes belonging to corrupt authorities. This way, the attacker will not be able to trivially win the game by decrypting the challenge ciphertext with a secret key given to him augmented with the key components from the corrupt authorities.

Challenger's Replies: The challenger flips a random coin $b \xleftarrow{R} \{0,1\}$ and replies with:

– The public keys $\mathrm{PK}_\theta \leftarrow \mathsf{AuthSetup}(\mathrm{GP}, \theta)$ for all $\theta \in \mathcal{N}_\Theta$.
– The secret keys $\mathrm{SK}_{\mathrm{GID}_i,u} \leftarrow \mathsf{KeyGen}(\mathrm{GID}_i, \mathrm{SK}_{\mathsf{T}(u)}, u, \mathrm{GP})$ for all $i \in [m]$ and for all $u \in S_i$.
– The challenge ciphertext $\mathrm{CT}^* \leftarrow \mathsf{Encrypt}(M_b, \mathbb{A}, \{\mathrm{PK}_\theta\}, \mathrm{GP})$ where $\{\mathrm{PK}_\theta\}$ is the set of all authority public keys (corrupt and non corrupt).

Guess: The attacker outputs a guess $b' \in \{0,1\}$.

Definition 2. *We say that an attacker* statically breaks *the scheme if it has a non negligible advantage in correctly guessing the bit b in the above security game.*

4 Our scheme

Our scheme constitutes an augmented version of the Lewko-Waters [27] CP-ABE construction and shares several of the existing techniques. Namely, in order to allow for multiple authorities and prevent collusion between users' keys it utilizes a hash function H that maps global identities to group elements. This hash function is modeled as a random oracle in the security proof. As noted in [27], this allows for a totally decentralized construction, since it provides all authorities with a secure way to personalize the secret key given to a specific user. To the best of our knowledge, it is still an open problem whether it is possible to create a multi-authority ABE scheme in the standard model, where no coordination is allowed between the different authorities.

Since we will be working in the random oracle model and we aim for practically deployable schemes, we combined the above technique with the technique from [49] that used a hash function F that hashes *attributes* to group elements. This way we overcame the restriction that each authority is used only once and at the same time achieved a large universe construction. This is because the random oracle usage naturally overcomes the "one-time" restriction and the policies are not any

[6] The only requirement is that they have the correct type.

more controlled by the authorities, but by the underlying attributes. The *individual randomness technique* from [28, 40] is integrated to the treatment of each attribute by choosing a fresh random exponent t.

Finally in order to "bind" the different ciphertext terms together we use the *layering* technique of [40]. For the same reason we introduce two secret sharing vectors: one that shares the secret z of the blinding factor and one that shares 0. In order to decrypt, a user has to use both of them. However, during decryption the "0-shares" are crucially entangled to the global identifier of the secret key of the user. As a result in the event that two or more users collude and try to decrypt the same ciphertext, the "0-shares" will result in a failed decryption, thus preventing collusion attacks.

4.1 Construction

Our proposed scheme consists of the following five algorithms:

$\mathsf{GlobalSetup}(1^\kappa) \to \mathrm{GP}$**:** The global setup algorithm takes as input the security parameter κ and chooses a suitable bilinear group $\mathbb{G}$ of prime order p with generator g. It also chooses a function H mapping global identities $\mathrm{GID} \in \mathcal{GID}$ to elements of $\mathbb{G}$,[7] and another function F mapping strings, interpreted as attributes, to elements of $\mathbb{G}$. Both of these functions will be modeled as random oracles in the security proof. Finally, it defines $\mathcal{U}$, $\mathcal{U}_\Theta$, and T as in Sec. 2.2. The global parameters are $\mathrm{GP} = \{p, \mathbb{G}, g, H, F, \mathcal{U}, \mathcal{U}_\Theta, \mathsf{T}\}$.

$\mathsf{AuthSetup}(\mathrm{GP}, \theta) \to \{\mathrm{PK}_\theta, \mathrm{SK}_\theta\}$**:** The authority setup algorithm chooses two random exponents $\alpha_\theta, y_\theta \xleftarrow{R} \mathbb{Z}_p$ and publishes $\mathrm{PK} = \{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$ as its public key. It keeps $\mathrm{SK} = \{\alpha_\theta, y_\theta\}$ as its secret key.

$\mathsf{KeyGen}(\mathrm{GID}, \theta, u, \mathrm{SK}_\theta, \mathrm{GP}) \to \{\mathrm{K}_{\mathrm{GID},u}, \mathrm{K}'_{\mathrm{GID},u}\}$**:** The key generation algorithm takes as input the user's global identifier GID, the identifier θ of the authority, the attribute u to create a key for, as well as the authority's secret key and the global parameters. It should be the case that $u \in \mathsf{T}^{-1}(\theta)$, i.e. that the attribute is controlled by the specific authority.

The algorithm first chooses a random $t \xleftarrow{R} \mathbb{Z}_p$ and it outputs the secret key:

$$\mathrm{SK}_{\mathrm{GID},u} = \left\{\mathrm{K}_{\mathrm{GID},u} = g^{\alpha_\theta} H(\mathrm{GID})^{y_\theta} F(u)^t, \mathrm{K}'_{\mathrm{GID},u} = g^t\right\}$$

$\mathsf{Encrypt}(M, (A, \delta), \{\mathrm{PK}_\theta\}, \mathrm{GP}) \to \mathrm{CT}$**:** The encryption algorithm takes in a message M, an access policy (A, δ) with $A \in \mathbb{Z}_p^{\ell \times n}$, the public keys of the relevant authorities, and the global parameters. As always, we define the function $\rho : [\ell] \to \mathcal{U}_\Theta$ as $\rho(\cdot) = \mathsf{T}(\delta(\cdot))$, i.e. the mapping of rows to authorities.

The algorithm first creates vectors $\boldsymbol{v} = (z, v_2, \ldots, v_n)^\top$ and $\boldsymbol{w} = (0, w_2, \ldots, w_n)^\top$, where z, v_2, …, v_n, w_2, …, $w_n \xleftarrow{R} \mathbb{Z}_p$. We let λ_x denote the share of z corresponding to row x, i.e. $\lambda_x = \langle \boldsymbol{A}_x, \boldsymbol{v} \rangle$, and ω_x denote the share of 0, i.e. $\omega_x = \langle \boldsymbol{A}_x, \boldsymbol{w} \rangle$, where $\boldsymbol{A}_x$ is the x-th row of A.

For each row x of A, it chooses a random $t_x \xleftarrow{R} \mathbb{Z}_p$. The ciphertext is computed as:

$$C_0 = M e(g,g)^z,\ \left\{C_{1,x} = e(g,g)^{\lambda_x} e(g,g)^{\alpha_{\rho(x)} t_x}, C_{2,x} = g^{-t_x},\ C_{3,x} = g^{y_{\rho(x)} t_x} g^{\omega_x},\ C_{4,x} = F(\delta(x))^{t_x}\right\}_{x \in [\ell]}$$

$\mathsf{Decrypt}(\mathrm{CT}, \{\mathrm{K}_{\mathrm{GID},u}, K'_{\mathrm{GID},u}\}, \mathrm{GP}) \to M$**:** Let (A, δ) be the access policy of the ciphertext. If the decryptor has the secret keys $\{\mathrm{K}_{\mathrm{GID},\delta(x)}, K'_{\mathrm{GID},\delta(x)}\}$ for a subset of rows $\boldsymbol{A}_x$ of A such that

[7] The global identifier universe $\mathcal{GID}$ can be any set that provides a unique identifier for each user and is mapped by H.

$(1, 0, \ldots, 0)$ is in the span of these rows, then for each such row x he computes:

$$C_{1,x} \cdot e(\mathrm{K}_{\mathrm{GID},\delta(x)}, C_{2,x}) \cdot e(H(\mathrm{GID}), C_{3,x}) \cdot e(\mathrm{K}'_{\mathrm{GID},\delta(x)}, C_{4,x}) = e(g,g)^{\lambda_x} e(H(\mathrm{GID}), g)^{\omega_x}$$

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

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

This is true because $\lambda_x = \langle \boldsymbol{A}_x, \boldsymbol{v} \rangle$ and $\omega_x = \langle \boldsymbol{A}_x, \boldsymbol{w} \rangle$, where $\langle (1, 0, \ldots, 0), \boldsymbol{v} \rangle = z$ and $\langle (1, 0, \ldots, 0), \boldsymbol{w} \rangle = 0$. The message can then be obtained as: $M = C_0 / e(g,g)^z$.

Re-Randomizing Re-randomizing techniques are applicable for the users' secret keys and the ciphertexts using only the public parameters, due to the linearity of all exponents. These techniques can provide properly distributed keys and ciphertexts even if originally the random choices in these algorithms are not uniform. We will use these techniques in our security reduction.

Specifically, if someone has a key $\left\{ \mathrm{K}_{\mathrm{GID},u}, \mathrm{K}'_{\mathrm{GID},u} \right\}$, he can acquire a new key for (GID, u) by picking $t' \xleftarrow{R} \mathbb{Z}_p$ and constructing $\left\{ \mathrm{K}_{\mathrm{GID},u} F(u)^{t'}, \mathrm{K}'_{\mathrm{GID},u} g^{t'} \right\}$.

For the ciphertext the re-randomization can be done by picking a new $z' \xleftarrow{R} \mathbb{Z}_p$, new random vectors $\boldsymbol{v}'$ and $\boldsymbol{w}'$ with the first elements z' and 0, respectively, and for each row x a new $t'_x \xleftarrow{R} \mathbb{Z}_p$. Then the re-randomized ciphertext is

$$C_0 e(g,g)^{z'}, \left\{ C_{1,x} e(g,g)^{\langle \boldsymbol{A}_x, \boldsymbol{v}' \rangle} e(g,g)^{\alpha_{\rho(x)} t'_x}, C_{2,x} g^{-t'_x}, C_{3,x} g^{y_{\rho(x)} t'_x} g^{\langle \boldsymbol{A}_x, \boldsymbol{w}' \rangle}, C_{4,x} F(\delta(x))^{t'_x} \right\}_{x \in [\ell]}$$

5 Implementation and Evaluation

Framework We implemented our scheme in Charm [1]; a framework developed to facilitate the rapid prototyping of cryptographic schemes and protocols. It is based on the Python language which allows the programmer to write code similar to the theoretical implementations. However, the routines that implement the dominant group operations use the PBC library [29] (written natively in C) and the time overhead imposed by the use of Python is usually less than 1%. Charm also provides routines for applying and using LSSS schemes needed for Attribute-Based systems. For more information on Charm we refer the reader to [13, 1].

We tested several ABE constructions on all elliptic curve bilinear groups provided by Charm, i.e. three super-singular symmetric EC groups and two "MNT" [32] asymmetric EC groups. In Table 2 of App. C we present the approximate security level each group provides with respect to the discrete log problem. The source code of our implementations can be found in [46]. All our benchmarks were executed on a dual core Intel® Xeon® CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04 and Python3.2.3.

Implementation Details All Charm routines use formally asymmetric groups (although the underlining groups might be symmetric) and therefore we translated our schemes to the asymmetric setting. Namely, we have three groups $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ and the pairing e is a function from $\mathbb{G}_1 \times \mathbb{G}_2$ to $\mathbb{G}_T$. We note here that we tried to implement our algorithms so that more operations are executed in the $\mathbb{G}_1$ group than in the $\mathbb{G}_2$ and that encryption consists mainly of operations in $\mathbb{G}_1$, compared

Our CP-ABE [Sec. 4.1] (Multi-authority, random oracle model, statically secure)

Curve	GS	AS	KG(4)	KG(8)	KG(12)	EC(4)	EC(8)	EC(12)	DE(4)	DE(8)	DE(12)
SS512	8.4	4.1	91.5	182.9	274.6	75.0	150.4	226.4	34.5	59.2	82.3
SS1024	58.0	43.8	631.4	1263.5	1894.5	666.9	1331.2	1997.2	641.4	1275.4	1907.4
MNT159	14.4	3.7	295.9	502.7	799.7	155.9	299.4	450.1	99.3	159.8	237.5
MNT201	19.5	4.6	370.5	787.0	1205.8	191.6	401.2	592.1	133.8	237.9	321.5
MNT224	24.1	5.5	489.5	838.4	1335.2	244.1	473.0	695.9	157.0	273.2	390.3

BSW CP-ABE [6] (Single-authority, generic group model, adaptively secure)

Curve	GS	AS	KG(4)	KG(8)	KG(12)	EC(4)	EC(8)	EC(12)	DE(4)	DE(8)	DE(12)
SS512	20.1	N/A	52.9	100.1	146.9	51.0	98.5	147.6	22.5	40.3	55.3
SS1024	213.3	N/A	394.0	710.3	1026.5	360.1	681.6	997.1	482.2	909.0	1333.9
MNT159	31.2	N/A	152.8	265.2	399.4	107.5	268.2	376.9	56.4	104.7	149.1
MNT201	42.2	N/A	221.5	335.1	557.8	169.8	331.7	564.5	76.3	142.5	205.5
MNT224	52.3	N/A	192.8	447.5	566.1	209.1	329.3	595.2	94.7	175.0	253.6

Waters CP-ABE [49] (Single-authority, random oracle model, adaptively secure)

Curve	GS	AS	KG(4)	KG(8)	KG(12)	EC(4)	EC(8)	EC(12)	DE(4)	DE(8)	DE(12)
SS512	20.4	N/A	39.6	73.9	108.0	64.2	124.8	186.4	32.5	60.1	85.3
SS1024	216.3	N/A	237.7	397.5	558.6	516.4	992.9	1464.4	627.0	1200.5	1770.1
MNT159	32.7	N/A	18.3	21.8	25.7	43.4	84.5	125.2	56.3	104.5	148.8
MNT201	44.6	N/A	25.4	31.7	37.2	58.8	118.3	170.7	77.0	143.4	206.9
MNT224	55.1	N/A	31.4	38.4	45.2	71.3	137.3	205.7	95.2	177.9	258.4

Table 1. Average running times in milliseconds of our scheme and two single authority schemes. The algorithms are denoted as GS: Global setup, AS: Authority setup, KG: Key generation for a user, EC: Encrypt, DE: Decrypt. The numbers in parentheses refer to the number of attributes in key generation, the number of rows of the policy in encryption, and the number of rows utilized during decryption. We can see the linear dependence between these numbers and the corresponding times.

to key generation. The reason is that the time taken to execute them in the $\mathbb{G}_1$ group is considerably smaller than $\mathbb{G}_2$ in specific asymmetric groups such as the "MNT" groups.

Regarding the comparisons to other schemes, the only fully decentralized multi-authority ABE scheme that provides expressive policies is the CP-ABE scheme of Lewko-Waters [27]. However, we decided to defer implementation and benchmarking of it for several reasons: Firstly, this scheme utilizes composite order groups, which are several orders of magnitude slower than the prime order groups that provide the same security level. We expect our scheme to be significantly faster. More information on the comparison between prime and composite groups can be found in Sec. D. Secondly, as mentioned in the introduction, the attributes utilized in the system have to be enumerated ahead of time and an one-use restriction is imposed on each attribute per policy. So even this scheme provides less flexibility than our construction. Thirdly, Charm does not support composite order groups, and finally, it is questionable the validity of the comparison between a prime order group and a composite order group, when the underlying elliptic curve is different and/or different optimizations have been applied to them.

Instead of this, we validate the claim that our system provides similar efficiency to existing single-authority ABE constructions, by providing implementation results of two single-authority ABE schemes. These are the Bethencourt-Sahai-Waters CP-ABE scheme [6] and the recent Waters CP-ABE [49]. Both of them were implemented by the Charm authors as typical examples. The former scheme is secure in the generic group model, while the implementation of the latter uses the random oracle version of it.

Timing Results Timing results in milliseconds are shown in Table 1. We see that our scheme achieves similar operation times to the two established single-authority schemes. In general, we attempted to keep execution times for encryption and decryption relatively low, while the times for setup and key generation can be significantly higher, since they are called only once.

6 Static Security

Our main security theorem is shown below.

Theorem 1. *If the q-DPBDHE2 assumption holds, then all probabilistic polynomial-time adversaries with a challenge matrix of size at most $q \times q$ have a negligible advantage in statically breaking our scheme in the random oracle model.*

In our security proof we combined several techniques, which we think might be of independent interest in the study of CP-ABE systems. The first technique allows the simulator of our reduction to isolate an unauthorized set of rows and essentially ignore it for the remaining of the security reduction. It can ignore the contributions of these rows even in the construction of the challenge ciphertext. In our case the simulator does that for the corrupt authorities, which are controlled by the adversary. The lemma that makes this technique possible is shown below and its proof follows in appendix A. The lemma allows the simulator to "zero-out" a subset of columns for the unauthorized set.

Lemma 1. *Let $A \in \mathbb{Z}_p^{\ell \times n}$ be the secret sharing matrix of a linear secret sharing scheme for an access policy $\mathbb{A}$ and let $\mathcal{C} \subseteq [\ell]$ be a non-authorized set of rows. Let $c \in \mathbb{N}$ be the dimension of the subspace spanned by the rows of $\mathcal{C}$.*

Then the distribution of the shares $\{\lambda_x\}_{x \in [\ell]}$ sharing the secret $z \in \mathbb{Z}_p$ generated with the matrix A is the same as the distribution of the shares $\{\lambda'_x\}_{x \in [\ell]}$ sharing the same secret z generated with some matrix A', where $A'_{x,j} = 0$ for all $(x, j) \in \mathcal{C} \times [n - c]$ (see figure 1).

$$A = \begin{bmatrix} A_{1,1} & A_{1,2} & \dots & A_{1,n} \\ A_{2,1} & A_{2,2} & \dots & A_{2,n} \\ A_{3,1} & A_{3,2} & \dots & A_{3,n} \\ \vdots & \vdots & \ddots & \vdots \\ A_{\ell,1} & A_{\ell,2} & \dots & A_{\ell,n} \end{bmatrix} \rightsquigarrow A' = \left[\begin{array}{ccc|c} 0 & \dots & 0 & A'_{1,n-c+1} \dots A'_{1,n} \\ A'_{2,1} & \dots & A'_{2,n-c} & A'_{2,n-c+1} \dots A'_{2,n} \\ 0 & \dots & 0 & A'_{3,n-c+1} \dots A'_{3,n} \\ \vdots & \ddots & \vdots & \vdots \quad \ddots \quad \vdots \\ A'_{\ell,1} & \dots & A'_{1,n-c} & A'_{1,n-c+1} \dots A'_{\ell,n} \end{array}\right]$$

Fig. 1. Transformation of the policy matrix A to be used by the simulator. Rows that belong to corrupted authorities are highlighted.

Another technique utilized in the security proof is the "splitting" of the unknown parameters to two different vectors. During the generation of the authorities' public keys, the elements in the first column of the challenge policy are programmed into the $e(g, g)^{\alpha_\theta}$ component, while the remaining in the g^{y_θ}. The same technique is applied on the challenge ciphertext, where the secret sharing vector $\boldsymbol{v}$ will hold the secret $z = sa^{q+1}$ on only the first position and the zero sharing vector $\boldsymbol{w}$ will hold the unknown terms $sa^q, sa^{q-1}, \dots, sa^2$ on all positions but the first. During the generation of the users' secret keys and the generation of the challenge ciphertext, all these terms are "recombined" to give a full series of q terms.

Proof. In order to prove the theorem we assume that there exists a probabilistic poly-time adversary $\mathcal{A}$ that breaks the scheme with more than negligible advantage and we show how to construct a probabilistic poly-time algorithm $\mathcal{B}$ that simulates the static security game with $\mathcal{A}$ and breaks the q-DPBDHE2 assumption. Our simulator works as follows:

Global Parameters: Initially, it gets (D, T) from its q-DPBDHE2 challenger and sends the public parameters $\mathrm{GP} = (\mathbb{G},\, p,\, g,\, \mathcal{U},\, \mathcal{U}_\Theta,\, \mathsf{T})$ to $\mathcal{A}$. The two random oracles H, F will be programmed by the simulator.

Static security: According to the static security game, the attacker $\mathcal{A}$ outputs the lists / components $\mathcal{C}_\Theta$, $\mathcal{N}_\Theta$, $\mathcal{Q} = \{(\mathrm{GID}_i, S_i)\}_{i=1}^m$, $(M_0, M_1) \in \mathbb{G}_T^2$, and (A, δ). Since we are in the random oracle model, the attacker also outputs a sequence $\mathcal{L}_\mathcal{I}$ of global identities for the H oracle queries and a sequence $\mathcal{L} \subseteq \mathcal{U}$ of attributes for the F oracle queries. W.l.o.g. we assume that all global ID's and all attributes present in $\mathcal{Q}$ are queried on their respective oracle.

In order to proceed, the simulator substitutes the secret sharing matrix A with the matrix A' from Lemma 1. After $\mathcal{B}$ calculates the matrix A' (shown in figure 1) where $\mathcal{C} = \mathcal{C}_\Theta$ it proceeds to compute all the inputs to $\mathcal{A}$. According to the above lemma, if $\mathcal{B}$ uses A' instead of A in the simulation the view of $\mathcal{A}$ in this game is information-theoretically the same as if it used the given matrix A. Therefore, the shares will be properly distributed. In the remainder of the proof, we use $n' = n - c$.

Authority Public Keys: The simulator has to provide the public keys of all non-corrupted authorities in $\theta \in \mathcal{N}_\Theta$. To do that it considers two cases:

If the authority in question, θ, is not in the challenge policy, i.e. $\theta \notin \rho[\ell]$, the simulator $\mathcal{B}$ picks $\alpha_\theta, y_\theta \xleftarrow{R} \mathbb{Z}_p$ itself and outputs the public key $(e(g,g)^{\alpha_\theta}, g^{y_\theta})$.

For each authority $\theta \in \rho[\ell] \setminus \mathcal{C}_\Theta$, let $X = \{x | \rho(x) = \theta\} \subseteq [\ell]$. This is the set of rows in the challenge policy that belong to authority θ. Then the simulator $\mathcal{B}$ picks $\tilde{\alpha}_\theta, \tilde{y}_\theta \xleftarrow{R} \mathbb{Z}_p$ and sets implicitly $\alpha_\theta = \tilde{\alpha}_\theta + \sum_{x\in X} b_x a^{q+1} A'_{x,1}$ and $y_\theta = \tilde{y}_\theta + \sum_{x\in X} \sum_{j=2}^{n'} b_x a^{q+2-j} A'_{x,j}$. It outputs the public key

$$(e(g,g)^{\alpha_\theta}, g^{y_\theta}) = (e(g,g)^{\tilde{\alpha}_\theta} \prod_{x\in X} e(g^{b_x a}, g^{a^q})^{A'_{x,1}},\ g^{\tilde{y}_\theta} \prod_{x\in X} \prod_{j=2}^{n'} \left(g^{b_x a^{q+2-j}}\right)^{A'_{x,j}})$$

Since $n' = n - c \leq q$ and $\ell \leq q$, the simulator can compute all these terms using suitable terms of the assumption. Also due to $\tilde{\alpha}_\theta, \tilde{y}_\theta$ these terms are properly distributed.

***H*-Oracle Queries:** If the queried global identity GID is in $\mathcal{L}_\mathcal{I}$ but not in $\{\mathrm{GID}_i\}_{i\in[m]}$, then the simulator outputs a random element of $\mathbb{G}$ for $H(\mathrm{GID})$. These elements are not going to be used anywhere else.

If the queried global identity is equal to GID_i for some i and there is no row x such that $\delta(x) \in S_i$ (i.e. if this user is not entitled to any shares), then the simulator picks $\tilde{h}_i \xleftarrow{R} \mathbb{Z}_p$ and outputs

$$H(\mathrm{GID}_i) = g^{\tilde{h}_i} \cdot g^a \cdot g^{a^2} \cdot \cdots \cdot g^{a^{n'-1}} = g^{\tilde{h}_i} \prod_{k=2}^{n'} g^{a^{k-1}}$$

Otherwise, for some rows $X' \subseteq [\ell]$ it is true that $\delta(x) \in S_i$. According to our restriction we know that the set of these rows together with the set of the rows that belong to corrupted authorities is non-authorized. This means that there exists a vector $\boldsymbol{d}_i \in \mathbb{Z}_p^{1\times n}$ such that the first element is $d_{i,1} = 1$ and the inner product of it with any of the aforementioned rows is equal to zero.

Additionally, according to the construction of A' we know that the set of the corrupted rows spans the entire subspace of dimension c (see Lemma 1). Hence the vector $\boldsymbol{d}_i$ is orthogonal to any of the vectors $(\overbrace{0,\ldots,0}^{n'},\overbrace{0,\ldots,0,1,0,\ldots,0}^{c}) \in \mathbb{Z}_p^{1\times n}$. These are the vectors with exactly one "1" in one of the last c positions. This implies that $d_{i,j}=0$ for $n-c+1 \le j \le n$. Hence $\langle \boldsymbol{A}'_x, \boldsymbol{d}_i\rangle = 0$ even if we restrict the row $\boldsymbol{A}'_x$ and the vector $\boldsymbol{d}_i$ to the first $n'=n-c$ positions. We will denote this inner product by $\overline{\langle \boldsymbol{A}'_x, \boldsymbol{d}_i\rangle} = 0$.

In this case the simulator picks $\tilde{h}_i \xleftarrow{R} \mathbb{Z}_p$ and outputs

$$H(\mathrm{GID}_i) = g^{\tilde{h}_i}\cdot (g^a)^{d_{i,2}}\cdot (g^{a^2})^{d_{i,3}}\cdot \cdots \cdot (g^{a^{n'-1}})^{d_{i,n'}} = g^{\tilde{h}_i}\prod_{k=2}^{n'}\left(g^{a^{k-1}}\right)^{d_{i,k}}$$

All the answers programmed in the random oracle are uniformly random in $\mathbb{G}$, thus properly distributed.

F-Oracle Queries: Let $\theta = \mathsf{T}(u)$ be the authority of the queried attribute u. Then if $\theta \notin \rho[\ell]$ or $\theta \in \mathcal{C}_\Theta$, the simulator outputs a random element of $\mathbb{G}$ for $F(u)$ and stores the value so that he might reuse it in a secret key query.

If $\theta \in \rho[\ell]$, let $X'' = \{x|\rho(x)=\theta\} \setminus \{x|\delta(x)=u\} \subseteq [\ell]$. This is the set of rows that belong to authority θ but *do not have* u as the corresponding attribute. Then the simulator picks $\tilde{f}_u \xleftarrow{R} \mathbb{Z}_p$ and outputs

$$F(u) = g^{\tilde{f}_u} g^{\sum_{x\in X''}\sum_{j\in[n']} b_x a^{q+1-j} A'_{x,j}} = g^{\tilde{f}_u}\prod_{x\in X''}\prod_{j\in[n']}\left(g^{b_x a^{q+1-j}}\right)^{A'_{x,j}}$$

Secret Keys: Consider the query (GID_i, S_i) where $S_i \subseteq \mathcal{U}$. The simulator $\mathcal{B}$ has to create the secret key $\{\mathrm{K}_{\mathrm{GID}_i,u}, \mathrm{K}'_{\mathrm{GID}_i,u}\}$ for every $u \in S_i$. It has to consider the following cases and act accordingly:

- $\mathsf{T}(u) = \theta \notin \rho[\ell]$: That is, the authority of the attribute is not present in the challenge policy. Here the simulator knows α_θ and y_θ. Therefore it picks $t \xleftarrow{R} \mathbb{Z}_p$ and outputs $\mathrm{K}_{\mathrm{GID}_i,u} = g^{\alpha_\theta} H(\mathrm{GID}_i)^{y_\theta} F(u)^t$ and $\mathrm{K}'_{\mathrm{GID}_i,u} = g^t$.
- $\mathsf{T}(u) = \theta \in \rho[\ell]$ and $S_i \cap \delta[\ell] = \emptyset$: In this case, the authority of the attribute is present in the challenge policy, but none of the attributes of this user is in it. Then, according to the H- and F-oracle phases $H(\mathrm{GID}_i) = g^{\tilde{h}_i} g^{\sum_{k=2}^{n'} a^{k-1}}$ and $F(u) = g^{\tilde{f}_u} g^{\sum_{x\in X}\sum_{j\in[n']} b_x a^{q+1-j} A'_{x,j}}$, where $X = \{x|\rho(x)=\theta\}$.

In this case, the simulator sets implicitly $t = -\sum_{k\in[n']} a^k$ and computes the key

$$\begin{aligned}
\mathrm{K}_{\mathrm{GID}_i,u} &= g^{\alpha_\theta} H(\mathrm{GID}_i)^{y_\theta} F(u)^t \\
&= g^{\sum_{x\in X} b_x a^{q+1} A'_{x,1}} g^{\sum_{x\in X}\sum_{j=2}^{n'}\sum_{k=2}^{n'} b_x a^{q+1+k-j} A'_{x,j}} g^{-\sum_{x\in X}\sum_{j\in[n']}\sum_{k\in[n']} b_x a^{q+1+k-j} A'_{x,j}} \\
&\quad \cdot g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \\
&= g^{-\sum_{x\in X}\sum_{j=2}^{n'} b_x a^{q+2-j} A'_{x,j}} g^{-\sum_{x\in X}\sum_{k=2}^{n'} b_x a^{q+k} A_{x,1}} g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \\
&= g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \prod_{x\in X}\prod_{j=2}^{n'}\left(g^{b_x a^{q+2-j}}\right)^{-A'_{x,j}} \cdot \prod_{x\in X}\prod_{k=2}^{n'}\left(g^{b_x a^{q+k}}\right)^{-A'_{x,1}}
\end{aligned}$$

$$\mathrm{K}'_{\mathrm{GID}_i,u} = g^t = \prod_{k\in[n']} (g^{a_k})^{-1}$$

Since t is not properly distributed, the simulator re-randomizes this key using the algorithm of section 4.1 and outputs the re-randomized key.

- $\mathsf{T}(u) = \theta \in \rho[\ell]$ and $S_i \cap \delta[\ell] \neq \emptyset$: Now the user holds some shares of the challenge policy. Therefore, we have that $H(\mathrm{GID}_i) = g^{\tilde{h}_i} g^{\sum_{k=2}^{n'} a^{k-1} d_{i,k}}$ and $F(u) = g^{\tilde{f}_u} g^{\sum_{x\in X''} \sum_{j\in[n']} b_x a^{q+1-j} A'_{x,j}}$ (Notice the X''). The simulator sets implicitly $t = -\sum_{k\in[n']} a^k d_{i,k}$ and outputs

$$
\begin{aligned}
\mathrm{K}_{\mathrm{GID}_i,u} &= g^{\alpha_\theta} H(\mathrm{GID}_i)^{y_\theta} F(u)^t \\
&= g^{\sum_{x\in X} b_x a^{q+1} A'_{x,1}} g^{\sum_{x\in X} \sum_{j=2}^{n'} \sum_{k=2}^{n'} b_x a^{q+1+k-j} A'_{x,j} d_{i,k}} g^{-\sum_{x\in X''} \sum_{j\in[n']} \sum_{k\in[n']} b_x a^{q+1+k-j} A'_{x,j} d_{i,k}} \\
&\quad \cdot g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \\
&\overset{\star\star}{=} g^{\sum_{x\in X\setminus X''} b_x a^{q+1} \overline{\langle \boldsymbol{A}'_x, \boldsymbol{d}_i\rangle}} g^{\sum_{x\in X\setminus X''} \sum_{\substack{j=2,k=2 \\ j\neq k}}^{n',n'} b_x a^{q+1+k-j} A'_{x,j} d_{i,k}} g^{-\sum_{x\in X} \sum_{j=2}^{n'} b_x a^{q+2-j} A'_{x,j} d_{i,1}} \\
&\quad \cdot g^{-\sum_{x\in X} \sum_{k=2}^{n'} b_x a^{q+k} A'_{x,1} d_{i,k}} g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \\
&\overset{\star\star}{=} \prod_{x\in X\setminus X''} \prod_{\substack{j=2,k=2 \\ j\neq k}}^{n',n'} \left(g^{b_x a^{q+1+k-j}}\right)^{A'_{x,j} d_{i,k}} \prod_{x\in X} \prod_{j=2}^{n'} \left(g^{b_x a^{q+2-j}}\right)^{-A'_{x,j}} \\
&\quad \cdot \prod_{x\in X} \prod_{k=2}^{n'} \left(g^{b_x a^{q+k}}\right)^{-A'_{x,1} d_{i,k}} g^{\tilde{\alpha}_\theta} H(\mathrm{GID}_i)^{\tilde{y}_\theta} \left(g^{y_\theta}\right)^{\tilde{h}_i} \left(g^t\right)^{\tilde{f}_u} \\
\mathrm{K}'_{\mathrm{GID}_i,u} &= g^t = \prod_{k\in[n']} \left(g^{a_k}\right)^{-d_{i,k}}
\end{aligned}
$$

As before $\mathcal{B}$ re-randomizes this key using the public parameters and outputs the re-randomized key.

$\star\star$: Notice that $X \setminus X'' = \{x | \delta(x) = u\}$ contains rows that map to u in the challenge policy. Thus, if $u \notin \delta[\ell]$, this set is empty and the two products after the second equality are eliminated. On the other hand, if $u \in \delta[\ell]$ according to our discussion in the creation of the H-oracle's answers $\overline{\langle \boldsymbol{A}_x, \boldsymbol{d}_i\rangle} = 0$.

Challenge Ciphertext: The first part of the ciphertext is calculated as $C_0 = M_b \cdot T$, where $b \overset{R}{\leftarrow} \{0,1\}$ is a random bit and T is the challenge term. Thus the simulator $\mathcal{B}$ implicitly set $z = sa^{q+1}$.

The simulator also sets implicitly

$$
\boldsymbol{v} = \left(sa^{q+1}, 0, \ldots, 0\right) \in \mathbb{Z}_p^n \quad \text{and} \quad \boldsymbol{w} = \left(\overbrace{0, sa^q, \ldots, sa^{q-n'+2}}^{n'}, 0, \ldots, 0\right) \in \mathbb{Z}_p^n
$$

Therefore for a row $x^* \in [\ell]$ that belongs to a corrupted authority we have that $\lambda_{x^*} = 0$ and $\omega_{x^*} = 0$, due to the fact that these rows have all "0"s in the first n' columns. Thus for these rows the simulator picks $t_{x^*} \overset{R}{\leftarrow} \mathbb{Z}_p$ and using the public key $\{e(g,g)^{\alpha_\theta}, g^{y_\theta}\}$ of the corrupted authority it computes:

$$
\begin{aligned}
C_{1,x^*} &= e(g,g)^{\lambda_{x^*}} e(g,g)^{\alpha_{\rho(x^*)} t_{x^*}} = \left(e(g,g)^{\alpha_{\rho(x^*)}}\right)^{t_{x^*}} \qquad C_{2,x^*} = g^{-t_{x^*}} \\
C_{3,x^*} &= g^{y_{\rho(x^*)} t_{x^*}} g^{\omega_{x^*}} = \left(g^{y_{\rho(x^*)}}\right)^{t_{x^*}} \qquad C_{4,x^*} = F(\delta(x^*))^{t_{x^*}}
\end{aligned}
$$

On the other hand for a row x^* that does not belong to corrupted authorities, we have that $\lambda_{x^*} = sa^{q+1} \cdot A'_{x^*,1}$ and $\omega_{x^*} = \sum_{j=2}^{n'} sa^{q+2-j} A'_{x^*,j}$. For each one of these rows $\mathcal{B}$ sets implicitly $t_{x^*} = -s/b_{x^*}$ and computes:

$$
\begin{aligned}
C_{1,x^*} &= e(g,g)^{\lambda_{x^*}} e(g,g)^{\alpha_{\rho(x^*)} t_{x^*}} = e(g,g)^{sa^{q+1} A'_{x^*,1}} e(g,g)^{-\sum_{x \in X} sb_x a^{q+1} A'_{x^*,1}/b_{x^*}} \\
&= \prod_{x \in X \setminus \{x^*\}} e(g, g^{sb_x a^{q+1}/b_{x^*}})^{-A'_{x^*,1}} \\
C_{2,x^*} &= g^{-t_{x^*}} = g^{s/b_{x^*}} \\
C_{3,x^*} &= g^{y_{\rho(x^*)} t_{x^*}} g^{\omega_{x^*}} = g^{-\sum_{x \in X} \sum_{j=2}^{n'} sb_x a^{q+2-j} A'_{x^*,j}/b_{x^*}} g^{\sum_{j=2}^{n'} sa^{q+2-j} A'_{x^*,j}} \\
&= \prod_{x \in X \setminus \{x^*\}} \prod_{j=2}^{n'} \left(g^{sb_x a^{q+2-j}/b_{x^*}} \right)^{-A'_{x^*,j}} \\
C_{4,x^*} &= F(\delta(x^*))^{t_{x^*}} = g^{-\sum_{x \in X''} \sum_{j \in [n']} sb_x a^{q+1-j} A'_{x^*,j}/b_{x^*}} = \prod_{x \in X''} \prod_{j \in [n']} \left(g^{sb_x a^{q+1-j}/b_{x^*}} \right)^{-A'_{x^*,j}}
\end{aligned}
$$

Notice that $x^* \notin X''$. Therefore the simulator can compute C_{4,x^*}. Since $\boldsymbol{v}$, $\boldsymbol{w}$ and the t_{x^*}'s are not properly distributed, the simulator re-randomizes the ciphertext using the algorithm of section 4.1.

Guess: If the attacker $\mathcal{A}$ correctly guessed the bit b, then the simulator $\mathcal{B}$ outputs that the challenge term was $e(g,g)^{sa^{q+1}}$. That is because in this case it simulated the static security game perfectly. If the attacker did not guess the bit correctly, the simulator answers that T was a random group element of $\mathbb{G}_T$. In this case the simulator produced an encryption of a random message. Therefore, if $\mathcal{A}$ is successful with more than negligible advantage, so is $\mathcal{B}$. □

References

[1] Joseph A. Akinyele, Matthew Green, and Avi Rubin. Charm: A framework for rapidly prototyping cryptosystems. Cryptology ePrint Archive, Report 2011/617, 2011. `http://eprint.iacr.org/`.

[2] Sattam S. Al-Riyami, John Malone-Lee, and Nigel P. Smart. Escrow-free encryption supporting cryptographic workflow. *Int. J. Inf. Sec.*, 5(4):217–229, 2006.

[3] Walid Bagga, Refik Molva, and Stefano Crosta. Policy-based encryption schemes from bilinear pairings. In *ASIACCS*, page 368, 2006.

[4] Manuel Barbosa and Pooya Farshim. Secure cryptographic workflow in the standard model. In *INDOCRYPT*, pages 379–393, 2006.

[5] Amos Beimel. *Secure Schemes for Secret Sharing and Key Distribution.* PhD thesis, Dept. of Computer Science, Technion, 1996.

[6] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *IEEE Symposium on Security and Privacy*, pages 321–334, 2007.

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

[8] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the weil pairing. *SIAM J. Comput.*, 32(3):586–615, 2003. extended abstract in Crypto 2001.

[9] Dan Boneh, Craig Gentry, and Michael Hamburg. Space-efficient identity based encryption without pairings. In *FOCS*, pages 647–657, 2007.

[10] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In *TCC*, pages 253–273, 2011.

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

[12] Robert W. Bradshaw, Jason E. Holt, and Kent E. Seamons. Concealing complex policies with hidden credentials. In *ACM Conference on Computer and Communications Security*, pages 146–157, 2004.
[13] Charm. `http://www.charm-crypto.com`.
[14] Melissa Chase. Multi-authority attribute based encryption. In *TCC*, pages 515–534, 2007.
[15] Melissa Chase and Sherman S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 121–130, 2009.
[16] Ling Cheung and Calvin C. Newport. Provably secure ciphertext policy abe. In *ACM Conference on Computer and Communications Security*, pages 456–465, 2007.
[17] Clifford Cocks. An identity based encryption scheme based on quadratic residues. In *IMA Int. Conf.*, pages 360–363, 2001.
[18] David Mandell Freeman. Converting pairing-based cryptosystems from composite-order groups to prime-order groups. In *EUROCRYPT*, pages 44–61, 2010.
[19] Steven D. Galbraith, Kenneth G. Paterson, Nigel P. Smart, and Nigel P. Smart. Pairings for cryptographers. In *Discrete Applied Mathematics*, 2008.
[20] Craig Gentry. Practical identity-based encryption without random oracles. In *EUROCRYPT*, pages 445–464, 2006.
[21] Vipul Goyal, Abhishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute based encryption. In *ICALP (2)*, pages 579–591, 2008.
[22] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *ACM Conference on Computer and Communications Security*, pages 89–98, 2006.
[23] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT*, pages 146–162, 2008.
[24] Arjen K. Lenstra, Eric R. Verheul, and Eric R. Verheul. Selecting cryptographic key sizes. In *Public Key Cryptography*, pages 446–465, 2000.
[25] Allison B. Lewko. Tools for simulating features of composite order bilinear groups in the prime order setting. In *EUROCRYPT*, pages 318–335, 2012.
[26] Allison B. Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribute-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT*, pages 62–91, 2010.
[27] Allison B. Lewko and Brent Waters. Decentralizing attribute-based encryption. In *EUROCRYPT*, pages 568–588, 2011.
[28] Allison B. Lewko and Brent Waters. Unbounded hibe and attribute-based encryption. In *EUROCRYPT*, pages 547–567, 2011.
[29] Ben Lynn. The stanford pairing based crypto library. `http://crypto.stanford.edu/pbc`.
[30] Gerome Miklau and Dan Suciu. Controlling access to published data using cryptography. In *VLDB*, pages 898–909, 2003.
[31] Miracl crypto sdk. `https://certivox.com/solutions/miracl-crypto-sdk/`.
[32] Atsuko Miyaji, Masaki Nakabayashi, and Shunzo Takano. Characterization of elliptic curve traces under fr-reduction. In *ICISC*, pages 90–108, 2000.
[33] National Institute of Standards and Technology. Nist special publication 800-37, 2010.
[34] Tatsuaki Okamoto and Katsuyuki Takashima. Homomorphic encryption and signatures from vector decomposition. In *Pairing*, pages 57–74, 2008.
[35] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In *ASIACRYPT*, pages 214–231, 2009.
[36] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In *CRYPTO*, pages 191–208, 2010.
[37] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *ACM Conference on Computer and Communications Security*, pages 195–203, 2007.
[38] Dan Page, Nigel P. Smart, Frederik Vercauteren, and Frederik Vercauteren. A comparison of mnt curves and supersingular curves. In *IACR Cryptology ePrint Archive*, page 165, 2004.
[39] Matthew Pirretti, Patrick Traynor, Patrick McDaniel, and Brent Waters. Secure attribute-based systems. In *ACM Conference on Computer and Communications Security*, pages 99–112, 2006.
[40] Yannis Rouselakis and Brent Waters. Practical constructions and new proof methods for large universe attribute-based encryption. In *ACM Conference on Computer and Communications Security*, pages 463–474, 2013.
[41] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In *EUROCRYPT*, pages 457–473, 2005.
[42] Adi Shamir. Identity-based cryptosystems and signature schemes. In *CRYPTO*, pages 47–53, 1984.

[43] Emily Shen, Elaine Shi, and Brent Waters. Predicate privacy in encryption systems. In *TCC*, pages 457–473, 2009.
[44] Elaine Shi and Brent Waters. Delegating capabilities in predicate encryption systems. In *ICALP (2)*, pages 560–578, 2008.
[45] Nigel P. Smart. Access control using pairing based cryptography. In *CT-RSA*, pages 111–121, 2003.
[46] Source code of our constructions. `www.rouselakis.com\RWABE`.
[47] Brent Waters. Efficient identity-based encryption without random oracles. In *EUROCRYPT*, pages 114–127, 2005.
[48] Brent Waters. Dual system encryption: Realizing fully secure ibe and hibe under simple assumptions. In *CRYPTO*, pages 619–636, 2009.
[49] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. PKC, 2011.

A "Zero-Out" Lemma

In order to prove Lemma 1 we will use the following theorem:

Theorem 2. *Let $A \in \mathbb{Z}_p^{\ell \times n}$ be the secret sharing matrix of a linear secret sharing scheme for an access policy $\mathbb{A}$ and $L \in \mathbb{Z}_p^{n \times n}$ be a matrix such that:*

- *The first row of L is $(1, 0, \ldots, 0) \in \mathbb{Z}_p^n$.*
- *The lower right matrix $L' \in \mathbb{Z}_p^{(n-1)\times(n-1)}$ of L has rank $n-1$.*

Then the distribution of the shares $\{\lambda_x\}_{x\in[\ell]}$ sharing the secret $z \in \mathbb{Z}_p$ generated with the matrix A is the same as the distribution of the shares $\{\lambda'_x\}_{x\in[\ell]}$ sharing the secret $z \in \mathbb{Z}_p$ generated with the matrix $A \cdot L$.

Proof. Consider the distribution of the shares $\{\lambda'_x\}_{x\in[\ell]}$. According to the construction of LSS schemes, it is true that $\lambda'_x = \langle \boldsymbol{AL}_x, \boldsymbol{v}\rangle$ where $\boldsymbol{AL}_x$ is the x-th row of the matrix AL and $\boldsymbol{v}$ is a random vector with its first element equal to z.

This implies that $\lambda'_x = \langle \boldsymbol{A}_x, \boldsymbol{Lv}\rangle$, where $\boldsymbol{Lv} \in \mathbb{Z}_p^n$ is the vector acquired by multiplying L with $\boldsymbol{v}$. Since L has the first row $(1, 0, \ldots, 0)$ we get that the first element of $\boldsymbol{Lv}$ is z. Moreover the remaining $n-1$ elements are uniformly random from $\mathbb{Z}_p$ because each one, say the i-th one, is equal to $z \cdot L_{i,1} + \langle \boldsymbol{L}'_i, \boldsymbol{v'}\rangle$ where $\boldsymbol{L}'_i$ is the i-th row of L' and $\boldsymbol{v'} \in \mathbb{Z}_p^{n-1}$ are the last $n-1$ elements of $\boldsymbol{v}$. Since these are uniformly random and L' is full rank, we get that $\langle \boldsymbol{L}'_i, \boldsymbol{v'}\rangle$ is uniformly random.

Therefore, $\boldsymbol{Lv}$ is distributed exactly the same as a secret sharing vector of z. Thus the shares $\{\lambda'_x\}$ have the same distribution as the shares $\{\lambda_x\}$.

Proof of Lemma 1 To convert the matrix A to the target matrix A' we will apply theorem 2. Let $\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_c$ be the first c independent rows in $\mathcal{C}$. These rows form a basis of size c of the relevant subspace and they can be computed from $\mathcal{C}$ in polynomial time using linear algebra operations.

Next we are going to extend this basis to size n such that the final basis spans the entire space. The first step is to add the row $\boldsymbol{U} = (1, 0, \ldots, 0) \in \mathbb{Z}_p^n$ to the set. Since the set of rows in $\mathcal{C}$ is unauthorized, $\boldsymbol{U}$ is not in in the subspace spanned by them and therefore this is a valid choice.

We continue by picking $n - c - 1$ rows, $\boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{n-c-1}$, such that the set

$$\{\boldsymbol{U}, \boldsymbol{V}_1, \boldsymbol{V}_2, \ldots, \boldsymbol{V}_{n-c-1}, \boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_W\}$$

is a basis of $\mathbb{Z}_p^n$. Using linear algebra operations this can be done in polynomial time as well.

Finally construct the matrix

$$L = (L')^{-1} = \begin{bmatrix} \boldsymbol{U} \\ \boldsymbol{V}_1 \\ \dots \\ \boldsymbol{V}_{n-c-1} \\ \boldsymbol{W}_1 \\ \dots \\ \boldsymbol{W}_c \end{bmatrix}^{-1} \in \mathbb{Z}_p^{n \times n}$$

Using theorem 2 we will argue that the matrix $A' = A \cdot L$ will give us same distribution for the $\{\lambda_x\}$ shares. We should argue first that the matrix L satisfies the requirements of the theorem. This can be done by trying to compute the inverse matrix, but one straightforward way is to use the blockwise inversion formula shown in figure 2.

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} A^{-1} + A^{-1}B(D - CA^{-1}B)^{-1}CA^{-1} & -A^{-1}B(D - CA^{-1}B)^{-1} \\ -(D - CA^{-1}B)^{-1}CA^{-1} & (D - CA^{-1}B)^{-1} \end{bmatrix}$$

Fig. 2. Blockwise inversion formula

In our case we have $A = [1]$, $B = (0, 0, \dots, 0) \in \mathbb{Z}_p^{1 \times (n-1)}$, $C = (0, 0, \dots, 0)^\top \in \mathbb{Z}_p^{(n-1) \times 1}$, and D is the lower right submatrix of L' of size $(n-1) \times (n-1)$. Therefore we have that

$$L = \begin{bmatrix} 1 & 0 \dots 0 \\ 0 & \\ \vdots & D^{-1} \\ 0 & \end{bmatrix}$$

Since D is of full rank, we can see that L satisfies the requirements of theorem 2. The only thing left to prove is that A' has the required form. That is, that for all rows $\boldsymbol{A'}_x$ with $x \in \mathcal{C}$, we have that the first $n - c$ elements are equal to 0. We know that for fixed $x \in \mathcal{C}$ the row $\boldsymbol{A}_x$ is a linear combination of the basis rows $\{\boldsymbol{W}_1, \boldsymbol{W}_2, \dots, \boldsymbol{W}_c\}$. Therefore $\boldsymbol{A}_x = \sum_{i \in [c]} \gamma_i \boldsymbol{W}_i$ with γ_i constants in $\mathbb{Z}_p$.

Finally, notice that for all k such that $n - c + 1 \leq k \leq n$ we have that

$$(\overbrace{0, \dots, 0}^{n-c \text{ terms}}, \overbrace{0, \dots, 0, 1, 0, \dots, 0}^{\substack{c \text{ terms} \\ 1 \text{ on the } k\text{-th position}}}) \cdot L' = \boldsymbol{W}_k \implies \boldsymbol{W}_k \cdot (L')^{-1} = (\overbrace{0, \dots, 0}^{n-c \text{ terms}}, \overbrace{0, \dots, 0, 1, 0, \dots, 0}^{\substack{c \text{ terms} \\ 1 \text{ on the } k\text{-th position}}})$$

Therefore the row $\boldsymbol{A'}_x = \boldsymbol{A}_x \cdot L = \sum_{i \in [c]} \gamma_i \boldsymbol{Z}_i$, where $\boldsymbol{Z}_i = (\overbrace{0, \dots, 0}^{n-c \text{ terms}}, \overbrace{0, \dots, 0, 1, 0, \dots, 0}^{\substack{c \text{ terms} \\ 1 \text{ on the } i\text{-th position}}})$. As a result the first $n - c$ elements of $\boldsymbol{A'}_x$ are all equal to 0. □

B Bilinear Groups

Our construction works with instantiations of bilinear groups of prime order. Abstractly, let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p, where the group operation is efficiently

computable in the security parameter. Let g be a generator of $\mathbb{G}$ and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be an efficiently computable pairing function that satisfies the following properties:

1. Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$ it is true that $e(u^a, v^b) = e(u, v)^{ab}$.
2. Non-degeneracy: $e(g, g) \neq \mathbb{1}_{\mathbb{G}_T}$.

The above definition considers the so called *symmetric* groups, where the two arguments of the pairing belong to the same group. In general, there exist *asymmetric* bilinear groups, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ and $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ are three different groups of prime order p. Several asymmetric instantiations of bilinear groups possess beneficial properties such as faster operations under the same security level and/or easier hashing to group elements.

C Approximate Security Level of all Charm Elliptic Curves

In Table 2 we present the approximate security levels of all the elliptic curves supported by Charm. Although the results of the table do not necessarily translate to the security level of our assumption (or the various assumptions of the other ABE schemes), they provides an intuitive comparison between the security levels of the different instantiations. For more information on the security of discrete log and of q-type assumptions we refer the reader to [33, 24, 19, 38].

Curve	Security Level (Bits)
SS512	80
SS1024	112
MNT159	70
MNT201	90
MNT224	100

Table 2. Approximate security levels of the utilized ECC groups. "SS" are super singular curves (symmetric bilinear groups), while "MNT" are the Miyaji, Nakabayashi, Takano curves (asymmetric bilinear groups). The number after the type of the curve denotes the size of the base field in bits.

D Prime vs Composite Order Group Operations

In order to demonstrate the generic difference in the efficiency of prime order vs composite order implementations, we timed the group exponentiation (of a random group element with a random exponent) and pairing operations (on random group elements) in the MIRACL framework [31] for different security levels. The benchmarks were executed on a dual core Intel® Xeon® CPU W3503@2.40GHz with 2.0GB RAM running Ubuntu R10.04. The elliptic curve utilized for all benchmarks was the super-singular (symmetric) curve $y^2 = x^3 + 1 \bmod p$ with embedding degree 2 for suitable primes p.

In table 3 we can see the significant gap between the timings in prime and composite order groups for the same security levels. This is the main reason that we used prime order groups for our construction.

Group exponentiation

Security Level (Bits)	Prime	Composite(2 primes)	Composite (3 primes)
80	3.5	66.9	201.6
112	14.8	448.1	1404.3
128	34.4	1402.5	4512.5
192	273.8	20097.0	66526.0

Pairing

Security Level (Bits)	Prime	Composite(2 primes)	Composite (3 primes)
80	13.9	245.3	762.3
112	65.7	1706.8	5485.2
128	176.6	5428.2	17494.4
192	1752.3	79046.8	263538.1

Table 3. Average timing results in milliseconds over 100 repeats of group exponentiations and pairings in MIRACL.