A Framework of Multi-Authority Attribute-Based Encryption with Outsourcing and Revocation

Sherman S. M. Chow
Department of Information Engineering
The Chinese University of Hong Kong
Sha Tin, N.T., Hong Kong
sherman@ie.cuhk.edu.hk

ABSTRACT

Attribute-based encryption (ABE) is a cryptographic tool for fine-grained data access control. For practical needs, an ABE scheme should support multiple authority and revocation. Furthermore, decryption should also be outsourced for higher efficiency. Researchers have been extending existing ABE schemes for these goals. Yet, the rationales are often hidden behind tailor-made number-theoretic constructions.

This paper proposes a framework for constructing multi-authority ABE schemes with attribute revocation and outsourced decryption, from any pairing-based single-authority ABE scheme which satisfies a set of properties we identified.

Keywords

multi-authority attribute-based encryption; pairing-based; outsourced decryption; revocation; design framework

1. INTRODUCTION

Access control appears in many aspects of our life, ranging from physical access to security gates at a building, to virtual access by a computer program to memory. There are various access control models, and correspondingly many cryptographic authentication protocols for enforcing access control. However, these traditional methods require the server which verifies the authentication to be honest, *i.e.*, granting access only when the authentication passed the access control policy. This implies that its administrator should be honest. Moreover, this server should be trusted to be free from any vulnerability, yet its online requirement to perform authentication makes it more susceptible to hacking.

1.1 Cryptographic Access Control

A better way is to enforce cryptographic access control, instead of the above software-based approach. Consider a confidential document which should only be revealed to some selected parties, we may employ encryption. Identity-based encryption (IBE) has been identified as a useful primitive. Since the identities can be any arbitrary bit-strings, they can encode the access control policy, and only the one who should be granted access can get the decryption key corresponding to a particular policy. There are many extensions of the basic IBE notion for security and efficiency (*e.g.*, [2, 8]). Numerous applications of IBE in the context of access control have been proposed, such as supporting timed-release encryption (*e.g.*, [9]).

For IBE, decryption is only possible when the policy-string associated with the decryption key matches exactly with what was specified during encryption. In other words, it cannot support any fuzziness in the policy representation. Fuzzy IBE [25] is proposed to address this problem. As long as the strings are sufficiently close, decryption is still possible. A related concept is threshold policy, such that both the keys and the encryption are associated with different set of attributes. When the number of their overlapping elements exceeds a certain predefined threshold, decryption will be possible. A fuzzy IBE scheme with closeness defined upon a threshold is later generalized to attribute-based encryption (ABE) supporting threshold policy [11].

A distinctive feature of ABE is collusion-resistance. Consider using a trivial solution of double IBE encryption with respect to two policies involved in a conjunctive policy. Two colluding users can pull together their keys to decrypt something that none of them alone is entitled to[1].

Goyal *et al.* [11] formulated the idea of attribute-based encryption and classified it into ciphertext-policy ABE (CP-ABE) and key-policy ABE (KP-ABE). In CP-ABE [1, 28], ciphertexts are associated with access structures, and keys are labeled with attributes. Decryption is possible if and only if the key attribute set satisfies the access structure. KP-ABE [11] forms the reversed notion in which ciphertexts are associated with attribute set and keys are associated with access structures. Consider an untrusted-but-honest server model, such as in cloud storage systems, CP-ABE fit wells with application requiring access control of documents.

1.2 Applications of ABE

The policy supported by ABE constructions in the literature can be a tree of gates including AND, OR, t-out-of-n threshold, or even NOT gates. Recent ABE schemes support more generalized access structure in the form of monotone access structure, which can be represented by linear secret sharing [28]. Just like IBE, there is a trusted authority which issues attribute-based decryption keys to user. When com-

[1]Previous works which address the problem of access control, but without collusion resistance, are outside our scope.

SACMAT'16, June 05-08, 2016, Shanghai, China

DOI: http://dx.doi.org/10.1145/2914642.2914659

pared with traditional public-key encryption, ABE is more efficient and flexible since one just needs to specify an access control formula instead of enumerating legitimate decryptors. Due to this level of *fine-grained access control*, ABE has found various applications, such as decentralized online social network (OSN) for ensuring privacy of the content to be shared over the network [15], or electronic healthcare system supporting patient privacy [20, 26]. In these scenarios, each OSN user or each patient takes the role of an authority respectively. Attributes are issued to the friends over the OSN, or healthcare personnel, for granting access of different kind of contents depending on the policy.

For a larger scale deployment of an ABE scheme, it may be unrealistic to assume the existence of a single authority to monitor all different domains of attributes. A *multiple authority* ABE scheme [4, 5, 17] allows distributing the job of issuing attributes over many attribute-authorities (AA's). For practical needs, just like other cryptosystems, an ABE scheme should support *revocation*. Furthermore, decryption can also be outsourced for higher efficiency [12]. However, revocation is a notoriously tricky issue in cryptosystem, and ABE is no exception. Moreover, splitting a single authority into multiple ones but preserving collusion-resistance is not an easy task. Finally, we need a better solution than simply leaking the decryption key for *outsourced decryption*.

1.3 Our Contributions

Researchers have been extending existing ABE schemes for the goals of multi-authority, attribute-level revocation, and outsourced decryption. Yet, the designs are mostly ad-hoc. While there are many ABE schemes being proposed in this decade, many ABE schemes with these extended features are not described with respect to the existing schemes. The rationales of the extensions are often hidden behind specific and tailor-made number-theoretic constructions [30, 29, 18, 20]. It becomes a laborious task to reverse engineer the constructions and assert their security. Unfortunately, some are later shown to be insecure [10, 14, 27, 22][2].

In this paper, we give a framework for constructing multi-authority ABE (MA-ABE) scheme with efficient decryption and revocation, from any single-authority ABE scheme which satisfies a set of properties we identified.

We do not claim that it is our contribution to achieve the aforementioned goals simultaneously, or to formulate a completely generic construction which automatically upgrades any ABE construction. However, we aim to give a blueprint which upgrades a class of ABE schemes, and possibly shed lights on how some of the constructions are extended from some others in the literature. To argue for the correctness and security of the resulting scheme, we still need to argue for some identified properties depending on the specific scheme we start with. Yet, we gain the benefits that the analysis can be simpler and modular since we do not need to consider the construction as a whole from scratch, but we can start with some specific structures of an existing scheme.

1.4 Key-Policy vs. Ciphertext-Policy MA-ABE

We chose to extend ciphertext-policy ABE instead of a key-policy one. If the policy is associated to a key, it is unclear which authority can determine the policy across different authorities. One of the first multi-authority ABE schemes in the threshold setting [5] simply assumes a conjunction of *threshold clauses* where each clause is maintained by a specific authority.[3] For an ABE scheme to support more complicated policy, such as assigning different weights on the clauses managed by different authorities, probably more coordinations between the authorities are required. Either the weighting scheme is a result of communication between the authorities, or there is a "super-authority" with special overriding power and decides for all other authorities, which is against the original intention of powers separation.

1.5 Overview of Our Framework

We exploit the algebraic structures of pairing-based ABE constructions including different kinds of homomorphism. Our framework requires the underlying ABE schemes with keys and the ciphertexts which can be partitioned into two parts, namely, attribute-independent part, and attribute-dependent part. To support multi-authority extension in a generic manner, we require that the randomness contributed by a particular authority to be "carried over" to another authority. More precisely, the attribute-dependent part of the user key can still be generated with respect to the randomness introduced by other authorities, without revealing the random exponent, by requiring a two-round key generation process. The aim is to generate a key certifying attributes from multiple authorities, but with the same key structure as that of the underlying single-authority ABE scheme as if all the attributes are certified by a single authority.

We then use our multi-authority extension to realize outsourced decryption. Both extensions are described with respect to small universe ABE schemes (to be explained shortly), but they also apply on large-universe schemes. Yet, for attribute-level revocation, it only works for small-universe.

2. RELATED WORK

Since the first proposal of Sahai and Waters [25] and subsequent refinement by Goyal *et al.* [11], ABE attracted lots of attentions from researchers of different sub-fields on various aspects, *e.g.*, security, efficiency, functionality, modularity. A survey of the earlier ABE schemes can be found in [6].

Considering the workload issue and a single point of trust, single-authority ABE suffers from high risks. While in multi-authority ABE (MA-ABE) systems, attributes are independently managed by different authorities, and the corruption of one or more authorities would not influence the security of other authorities. Chase [4] proposes an MA-ABE solution, by still requiring the help of a central authority. Chase and Chow [5] removed the central authority, which resulted in a truly multi-authority ABE scheme. These schemes support threshold policy, which can be described as KP-ABE.

Multi-authority ABE schemes have been constructed using composite-order (bilinear) groups in the *random oracle model* (ROM) [17], and using prime-order groups but with proof relies also on generic group model [17]. These schemes

[2]Some of these schemes are omitted from the references.

[3]A few papers in the literature (*e.g.*, [17, 29]) mentioned that the scheme of Chase and Chow [5] is restricted to only "AND policy of a determined set of authorities", which should not be confused with "AND policy" in general since the AND gate happens at the authority level. The clause for the attributes managed by the same authority can be a threshold clause. As mentioned in two seminal papers [25, 4], a simple trick can be applied to adjust the threshold. One can also leave out a subset of authorities during encryption.

support an expressive kind of policies known as monotone span programs, but only over a *small universe* of attributes, which means that the size of the public parameter is linear in the number of supported attributes. For an ABE scheme which supports a *large universe*, any string can be used as an attribute, and these attributes are not necessarily pre-defined and enumerated during setup. Rouselakis and Waters [23] proposed a large universe multi-authority ABE scheme, also in the ROM. This scheme uses prime-order groups and hence is more efficient in general. Yet, a consequence is that the dual-system encryption framework basing on composite-order groups [17] cannot be applied. Instead, this scheme [23] is only proven secure in a static security model where both the challenge ciphertexts and key queries are issued before the parameters are published.

While these schemes form the foundation of later work, with their design rationale well-explained; they did not consider extra feature such as user revocation.

Revocation in ABE can be traced back to IBE revocation [2] where private keys are tied with expiration time and ciphertext are associated with creation time. Decryption is impossible if the expiration time of the key is earlier than the creation time of the ciphertext. Sahai *et al.* [24] designed a (single-authority) revocable-storage ABE system, with updatable dynamic credentials. The ciphertext can be self-updated, *i.e.*, without the intervention of any authority. While the self-update feature is attractive, the ciphertext size is large due to the tree-based revocation mechanism.

Apart from self-updatable encryption, there are other studies on revocable ABE, which may require data owner assistance [15] or proxy assistance [30, 15]. Some of these schemes only support user-level revocation. Since the possession of attributes defines users' access abilities, attribute-level revocation is more intuitive. Attribute-level revocation in MA-ABE can be realized by dynamically renewing ciphertext as well as the non-revoked users' keys when revocation happens [29, 20]. Instead of time-dependent revocation as discussed above, the revocable-ABE system can depend on a version key: a parameter embedded into ciphertexts and private keys. When revocation happens, the authority creates update messages for cloud server and non-revoked users. Also new keys (or new ciphertext) can be easily calculated from update messages and the old keys (or old ciphertext).

3. SYSTEM MODEL

We describe how different entities use the algorithms of an ABE system in realizing a multi-authority cryptographic access control system over cloud storage. Figure 1 depicts the major entities. It consists of a set of authorities and a cloud server with storage and computation ability. There are two kinds of users, namely, data uploader and data consumers. Any party can upload data (yet we can always incorporate external authentication mechanism to govern who can upload the data, *e.g.* [7]). Data consumers are characterized by different attributes, decided via external means (*e.g.*, an attributes list indexed by the data consumers' identities).

External to the ABE scheme, we assume the existence of a trusted certificate authority (CA) which certifies the identities of users. Note that for multi-authority ABE, the notion of global identifier is crucial to ensure collusion resistance [4]. This trusted CA is just a means to realize this necessary condition[4]. In particular, this CA does not help in any algorithms of the ABE system except simply outputting certificates (different from [21]), and it lacks of the power of issuing decryption key nor decrypting any ciphertext.

3.1 Multi-Authority ABE

There are a number of attribute-authorities (AA's), which cooperatively setup and maintain the (MA-)ABE system. They first execute the Setup algorithm. The resulting public parameters param and the master public key mpk will be published. Each of them stores msk_i in secret. Correspondingly, the master public key mpk could be in the form of multiple mpk_i, one for each authority i.

Once the ABE system is setup, anyone can perform the encryption and upload the resulting ciphertext to the cloud. To do that, the data uploader first executes the Encap mechanism according to param and mpk with respect to the policy $\mathcal{P}$. Here, Encap is a key-encapsulation mechanism which outputs a ciphertext $\mathfrak{C}$ together with an encapsulated key $\mathfrak{K}$. The actual data will be encrypted by $\mathfrak{K}$ via symmetric-key encryption. Since it is the uploader who defined the access policy, and it is the authorities which grant the keys; the cloud server simply stores these ciphertexts in the data storage, and does not need to perform any access control.

Before the users can decrypt their first ciphertext, they need to talk to the AA's for getting the decryption key. Each of the AA's will issue keys to the user according the user identity certified by a CA. The identity can be used to determine the set of eligible attributes. For efficiency, each AA does not need to communicate with each other during the key issuing. After a user has interacted with all of the AA's, the keys from different AA's can be combined into a user attribute-based decryption key. This concludes the basic working mechanism of a multi-authority ABE scheme.

3.2 Outsourced Decryption

In the cloud computing settings, it is natural for the user to not only outsource the data storage but also the computation. In ABE, while still practically efficient, the decryption is usually the most computationally expensive task. In view of this, a user can outsource the decryption key to a *decryption mediator* performed by the cloud server. As usual, users want to minimize the trust on the cloud server, and hence the user should perform a (one-time) process such that only a blinded version of the decryption key is delegated to the mediator. The outsourcing thus includes three steps: (1) the data consumer (user) generates a blinded private key, and outsources it to the decryption mediator; (2) for every ciphertext, the user can request the decryption mediator to partially decrypt it by the outsourced key, which the mediator will reply with a partial decryption of the ciphertext to the user; (3) the user completes the full decryption of the ciphertext. In this way, as long as the first step is a one-time process, and the last step is efficient enough, the user can outsource most of the decryption computation to the cloud server which significantly reduces the decryption cost.

3.3 Revocation by Ciphertext and Key Updates

Considering dynamic credential in cloud storage, users' possession of a particular attribute can be dynamic. As in any other cryptographic schemes, revocation is a useful

[4]There can be privacy problem associated with having a CA to certify the global identifier of users, which is addressed [5].

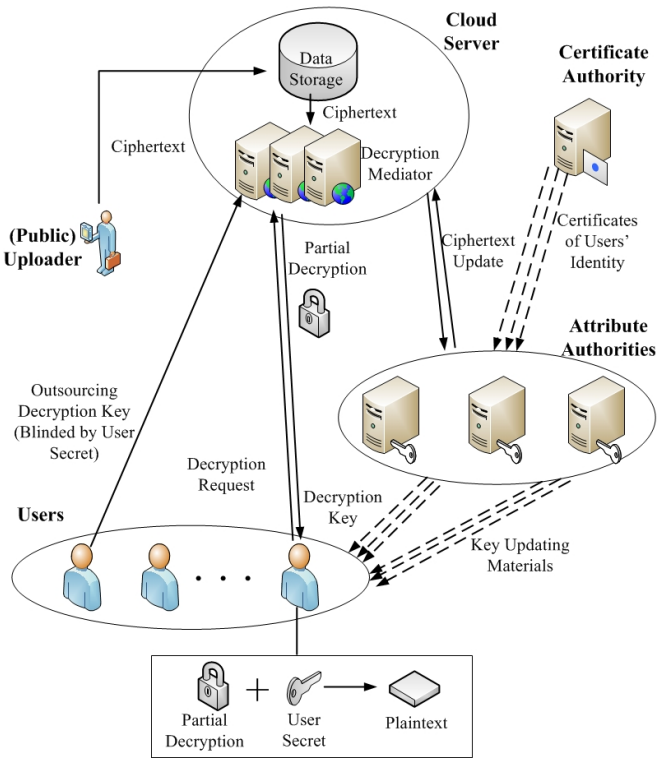

Figure 1: System Model

feature. For example, when a user reports a compromise of the decryption key, the whole key of this user should be revoked; or a user may be released from one of the duties and the corresponding attributes originally granted should be revoked. In this case, just like the access control policy, the revocation should also be fine-grained up to the attributes.

Fine-grained revocation is difficult to enforce especially for a multi-authority system due to the requirement of collusion-resistance. Some existing solutions only support revocation at the key-level instead of attribute-level, while some others require an online party to re-encrypt [30, 15] or partially decrypt for every ciphertext. Note that the partial decryption here is different from the outsourced decryption we discussed above. Revocation is for security concern and outsourced decryption is motivated by an efficiency concern.

Evolution of the "attribute-dependent key" is a way to support attribute-level revocation. It means that each AA can update a part of the master key which governs the attributes to its "next version" [30], via a *system-update* SU algorithm. After that, any existing ciphertexts should be updated, via a *ciphertext-update* CU algorithm. This update should not involve the decryption of the ciphertext for efficiency and security reasons. Yet, the update should probably be executed in a secure environment, and any old ciphertext should be securely deleted. To enforce revocation, only an updated decryption key can decrypt an updated ciphertext. Since the attribute-dependent keys for different domains of attributes are managed by the AA's independently, AA's are the parties to enforce the revocation. Via a *key-update* KU algorithm, AA's use the key-updating material to update the user decryption key if they are not revoked.

4. TECHNICAL PRELIMINARIES

This section gives a formal syntax of the suite of algorithms constituting an ABE scheme or an MA-ABE scheme. Then we present the necessary technical preliminaries for instantiating our framework, which include the relevant number theoretic primitives and cryptographic building blocks.

4.1 Syntax of ABE

In an MA-ABE scheme, each attribute authority handles a set of attributes, and all of these sets are disjoint.

Definition 1. *An N-authority ABE scheme consists of four algorithms:*

1. *Via* $(\mathsf{param}, \{(\mathsf{mpk}_k, \mathsf{msk}_k)\}_{k\in\{1,\dots N\}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, N)$ *the randomized key generation algorithm takes as input a security parameter $\lambda \in \mathbb{N}$ and the number of authorities $N \in \mathbb{N}$, and outputs the system parameters* param *and N public/private key pairs* $(\mathsf{mpk}_k, \mathsf{msk}_k)$*, one for each attribute authority $k \in \{1, \dots N\}$.*

   For an ABE scheme with small universe, the list of attributes can also be an input of the Setup *algorithm.*

   For simplicity, we assume param *and* $\{\mathsf{mpk}_k\}_{k\in\{1,\dots N\}}$ *are the implicit inputs of the rest of the algorithms.*

2. *Via* $\mathsf{usk}_k[\mathsf{id}, \mathbb{A}_k] \xleftarrow{\$} \mathsf{KGen}(\mathsf{msk}_k, \mathbb{A}_k, \mathsf{id})$ *the attribute authority k uses its secret key* msk_k *to output a user (secret) decryption key corresponding to a set of attribute $\mathbb{A}_k$ for the user with identity* id*.*

3. *Via* $\langle \mathfrak{C}, \mathfrak{K} \rangle \xleftarrow{\$} \mathsf{Encap}(\{\mathsf{mpk}_k\}_{k\in\{1,\dots N\}}, \mathcal{P})$ *a sender creates a ciphertext and its encapsulated key $\mathfrak{K}$ under the policy $\mathcal{P}$.*

4. *Via* $\mathfrak{K} \leftarrow \mathsf{Decap}(\{\mathsf{usk}_k[\mathsf{id}, \mathbb{A}_k]\}_{k\in\{1,\dots N\}}, \mathfrak{C})$ *a user* id *who obtained from each authority k a sufficient set of decryption keys* $\{\mathsf{usk}_k[\mathsf{id}, \mathbb{A}_k]\}$ *decrypts $\mathfrak{C}$ to recover the encapsulated key $\mathfrak{K}$.*

4.1.1 Key Generation

Our framework considers KGen algorithm to be probabilistic, such that different users cannot mix-and-match their key components. This is similar to many MA-ABE schemes in the literature [4, 5, 23]. Nevertheless, we remark here that being probabilistic is only one of the ways to ensure collusion-resistance. As long as the user identity id is also one of the inputs, different users having different identities can get different keys for the same attribute even if KGen is deterministic. When colluding users share their attributes, the key component of user id_1 for a certain attribute will not be in the same format expected by the user id_2, and hence it does not work with the other key components of id_2. In this way, collusion-resistance is still possible, *e.g.*, [17].

4.1.2 Encapsulation

The key encapsulation algorithm Encap implicitly samples a session key $\mathfrak{K}$ uniformly at random. The corresponding decapsulation algorithm Decap will recover $\mathfrak{K}$ if the attributes in the key satisfy the policy specified as an input of Encap.

Both KGen and Encap are probabilistic. When describing how to extend an underlying ABE scheme in our framework, it is useful to make the randomness used by these algorithms explicit, which will be denoted by $\mathsf{KGen}(\mathsf{msk}_k, \mathbb{A}_k, \mathsf{id}; u)$ and $\mathsf{Encap}(\mathcal{P}; s)$ where u or s is the randomness, respectively.

4.1.3 Correctness and Security

Definition 2. *An N-authority ABE scheme satisfies correctness if for all $\lambda, N \in \mathbb{N}$ and all identities id, for all $\{\mathbb{A}_k\}$ and $\mathcal{P}$ such that $\mathcal{P}(\{\mathbb{A}_k\}_{k\in\{1,\dots N\}})$ returns true, we have*

$$\Pr[\mathsf{Decap}(\{\mathsf{KGen}(\mathsf{msk}_k, \mathbb{A}_k, \mathsf{id})\}_{k\in\{1,\dots N\}}, \mathfrak{C}) = \mathfrak{K} :$$
$$(\mathsf{param},\ \{(\mathsf{mpk}_k, \mathsf{msk}_k)\}_{k\in\{1,\dots N\}}) \xleftarrow{\$} \mathsf{Setup}(1^\lambda, N);$$
$$\langle \mathfrak{C}, \mathfrak{K}\rangle \xleftarrow{\$} \mathsf{Encap}(\mathcal{P});] = 1$$

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

We follow a common existing security definition for multi-authority ABE, which only allows static corruption model of authority [4, 5, 17, 23]. There are variations in the strength of the security model. For example, adaptive query of decryption keys [17] or static model which all decryption keys must be asked before the system parameter is published [23].

We remark that, while Chase and Chow [5] gave a security analysis with respect to $(N-2)$ authorities (similar to the underlying distributed pseudorandom function), it can be extended to allow static corruption of $(N-1)$ authorities by a specific argument.

4.2 Bilinear Groups and Other Notations

Definition 3 (Bilinear Groups). *Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be three multiplicative cyclic groups all of order p. Suppose g_1 and g_2 are the generators of the base groups $\mathbb{G}_1$ and $\mathbb{G}_2$ respectively. A map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a* pairing *or a* bilinear map *if it satisfies the following properties:*

- (Bilinearity.) *$\hat{e}(u^x, v^y) = \hat{e}(u, v)^{xy}$ for all $u \in \mathbb{G}_1$, $v \in \mathbb{G}_2$, and $x, y \in \mathbb{Z}_p$.*
- (Non-degeneracy.) *$\hat{e}(g_1, g_2) \neq 1$, where 1 is the identity element in $\mathbb{G}_T$.*
- (Computability.) *$\hat{e}(u, v)$ can be computed in polynomial time for all $u \in \mathbb{G}_1$ and $v \in \mathbb{G}_2$.*

The order here is usually a prime number which is large (relative to the security parameter), but it can also be a product of a few large primes (*e.g.*, [21, 17]). Our framework can work in either case.

The bilinear groups context $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, \hat{e})$ will be used by any pairing-based ABE schemes. We overload the notation of param to refer to the groups description.

The pairing is symmetric when $\mathbb{G}_1 = \mathbb{G}_2$; otherwise, it is asymmetric. Group elements for asymmetric pairings are smaller, and the pairing is more efficient, when compared with the symmetric group at a fixed security level.

We assume that the isomorphism $\psi()$ from $\mathbb{G}_2$ to $\mathbb{G}_1$ is computable in polynomial time, but its inverse $\psi^{-1}()$ is not, *i.e.*, we are making the external Diffie-Hellman (XDH) assumption, which is for the pseudorandom function we use.

Assuming $\psi(g_2) = g_1$, where both g_1 and g_2 are the generators of the respective base groups given in param. To come up with a pair of random $\mathbb{G}_1$ and $\mathbb{G}_2$ element (h, h') such that $\psi(h') = h$, one may pick a random exponent $\beta \in \mathbb{Z}_p$, set $h = g_1^\beta$ and $h' = g_2^\beta$. We will use this notation to denote such a pair and also this procedure in our ABE instantiation.

For $r \in \mathbb{Z}_p$ and a set of group elements $\{g_i\}$, we write $\{g_i\}^r$ to refer to $\{g_i^r\}$.

Finally, we use $[i]$ to denote the set of integers from 1 to i.

4.3 Distributed Pseudorandom Functions

Pseudorandom functions (PRFs), informally, is a class of function family such that no polynomial-time adversary can distinguish between a randomly chosen function among this family and a truly random function (whose outputs are sampled uniformly and independently at random), with significant advantage relative to the security parameter. Each PRF takes a secret seed which also serves as an index to determine which function in the family to be used. We will use a set of distributed PRFs (DPRF) with outputs always multiply to the identity element on any input, as used in some existing multi-authority ABE schemes [5, 19].

Our framework expects the DPRF which takes an input of bit strings and outputs an element in group $\mathbb{G}_1$. In more details, looking ahead, there is a one-time setup cost among the AA's to setup this DPRF. Each pair of AA's, say, indexed by (j, k), shares a PRF seed $seed_{jk}$, say via a (non-interactive) key exchange. With these $O(N^2)$ PRFs in total, we set the PRF of the k-th AA, $F_k(\mathsf{id})$, to be a multiplication of $(N-1)$ basic PRFs on the same input id, each weighted by either 1 (for multiplication) or -1 (for division).

By choosing between multiplication and division accordingly, all these PRF values can cancel each other when $F_k(\mathsf{id})$ for different k are multiplied together. The final composite PRF is computed as $F_k(\mathsf{id}) = \prod_{j<k} \mathsf{prf}_{jk}(\mathsf{id}) / \prod_{j>k} \mathsf{prf}_{jk}(\mathsf{id})$, where $\mathsf{prf}_{jk}(\cdot)$ is the notation for a PRF which takes the secret seed shared by parties j and k. Informally, such a "product-of-PRF" construction still looks pseudorandom to any adversary controlling less than $(N-2)$ other authorities.

5. PARTITIONED ABE FROM PAIRINGS

Now we are ready to start the exposition of our framework of "upgrading" a single-authority ABE. Underlying our extensions for supporting multi-authority, attribute revocation, and outsourced decryption is the possibility of partitioning the master key and the ciphertext into two parts. One part is attribute-independent, and the other is attribute-dependent. This is a natural property in most existing schemes. Key partition will be one of the properties useful for our extension for multi-authority, which is also leveraged to support outsourced decryption. Ciphertext partition, together with other properties (in Section 7), will be useful for our extension to support attribute revocation.

5.1 Key Partition

The attribute-independent part of the master key corresponds to the authority-specific part of the key. Each authority manages a set of attributes, and correspondingly generates the attribute-dependent part of the master key.

For brevity of exposition, we abuse the term "key" to either refer to the public key and the corresponding private key collectively, or either one of them. Which one we mean should be clear from the context. For example, the key used to derive a ciphertext should be the public key, and the key used to derive a user secret key should be the private key.

We also abuse the notation of $\mathsf{KGen}()$ a bit to refer to the two conceptual sub-parts of

$$(\{L_k\}, \{K_i\}) \xleftarrow{\$} \mathsf{KGen}(\mathsf{gsk}, \{\mathsf{ask}_i\}, \mathbb{A}, \mathsf{id}; u)$$

as defined by the definition below.

Definition 4. *An ABE scheme is said to be having key partition if*

1. *(Partition of Master Keys:) The master key can be partitioned into two following parts.*
   (a) *An attribute-independent part:* $\mathsf{gpk}/\mathsf{gsk}$.
   (b) *An attribute-dependent part:* $\{\mathsf{apk}_i/\mathsf{ask}_i\}$.
2. *(Partition of User Secret Keys:) Each user secret key can be partitioned into two parts. Specifically,*
   (a) *An attribute-independent part* $\{L_k\}$ *that is derived from* gsk*, i.e.,*
   $$\{L_k\} \xleftarrow{\$} \mathsf{KGen}(\mathsf{gsk}, \mathsf{id}; u),$$
   where u *is the randomness used by* KGen *and* k *is a counter which enumerates the terms in* $\{L_k\}$ *according to the scheme.*
   (b) *An attribute-dependent part* $\{K_i\}$ *that is derived from* $\{\mathsf{ask}_i\}$*, i.e.,*
   $$\{K_i\} \xleftarrow{\$} \mathsf{KGen}(\{\mathsf{ask}_i\}, \mathbb{A}, \mathsf{id}),$$
   where i *is a counter which enumerates the attributes.*

5.2 Ciphertext Partition

For efficient revocation, we require the underlying ABE scheme to produce ciphertext featuring partition property similar to what we have defined for key partition.

Definition 5. *An ABE scheme is said to be having ciphertext partition if a ciphertext consists of the following components:*

1. *An attribute-dependent part* $\{D_i\}$ *that is derived from* $\{\mathsf{apk}_i\}$.
2. *An attribute-independent part*[5] $\{C_k\}$ *that is derived from* gpk.

Since we partition all of the master public/private key, user secret key, and ciphertext into different components, for brevity we overload each function for referring to the part of the algorithm which processes the partitioned version, *i.e.*, taking part of the inputs and returning part of the outputs.

5.3 Pairing-Based Attribute-Based Encryption

Here we define a template for a pairing-based ABE scheme.

Definition 6. *In our framework, a pairing-based attribute-based encryption is defined to have the structures below.*

1. *We assume that there is a trusted initializer, which sets up the basic system parameters including the bilinear groups context* param[6].
2. *The attribute-independent part of master public key,* gpk*, is in* $\mathbb{G}_T$*, and that of the master secret key* gsk *is in* $\mathbb{G}_1$*. (It can be extended to be multiple elements.)*
3. *A user secret key is in the form of* $(\{K_k\}, \{L_i\})$*, all of them are* $\mathbb{G}_1$ *elements.*
4. *Each element in the set* $\{\mathsf{apk}_i\}$ *is a group* $\mathbb{G}_2$ *element, which is for encoding the attribute involved in the policy to be specified during encapsulation.*
5. *A ciphertext is in the form of* $(\{C_k\}, \{D_i\})$*, all of them are* $\mathbb{G}_2$ *elements.*
6. *(Pairing-Product Equation for Decryption:) The decryption algorithm recovers the encapsulated key by the equation* $\prod \hat{e}(C_k, L_k) \prod \hat{e}(D_i, K_i)^{\omega_i}$*, where* ω_i *can be derived from the ciphertext and the decryption key.*

It is natural for decryption to be relying on a pairing-product equation. Since, after all, the confidentiality provided by the ciphertext relies on a pairing-based assumption often related to the indistinguishability of a target group element from random. Thus, for an honest user, decryption should be done by pairing up the group elements from the private key with those group elements from the ciphertext. This also justifies the assumption that gpk is a $\mathbb{G}_T$ element.

There could be ABE schemes which are outside the above formulation, for example, having a $\mathbb{Z}_p$ element in the decryption key or a $\mathbb{G}_T$ element in the ciphertext.

6. MULTI-AUTHORITY DISTRIBUTION

With the above abstraction of partitionable ABE, we start with our first extension of turning a single-authority ABE scheme into a multi-authority ABE scheme. The high-level goal here is to create a user decryption key sharing the same structure as that of the underlying single-authority ABE, as if all the attributes are managed by a single authority.

The homomorphic structure of discrete-logarithm-based cryptosystems often makes the task of distributing the keys easier. For MA-ABE, we want to distribute the master keys, such that this distribution carries over to the user secret key (to be modelled by the homomorphic properties of user secret keys below) and the encapsulated key (to be modelled by the homomorphic property of encapsulated key below).

ABE should be collusion-resistant, such that any "mix-and-match" attacks across different user decryption keys cannot be successful as the key components from different colluding users are randomized in different ways. To ensure that, one way is to apply a pseudorandom function on the identity of the user which is supposed to be unique (and assumed to be certified by some means) [4, 5, 17]. For our extension, we use a DPRF as reviewed in Section 4.3.

Another complication is that, the user decryption key generation of the underlying single-authority ABE can be probabilistic. A reason is that the randomization can protect the master secret key, such that it cannot be recovered easily from the user decryption key. That means any random factor chosen by one authority cannot be simply passed to another authority. On the other hand, as discussed, collusion-resistance expects the decryption key from different authorities for the same user to be randomized (or if one resorts to the help of a random oracle [17, 23], it can just be user-specific instead of being randomized). The consequence is that, different authorities need to agree upon the same randomness whenever issuing keys to the same user.

These observations motivate an additional property in our framework, termed as inter-dependency of the user secret key. This property requires that the attribute-independent part of the user secret key can serve as a "carrier" for the randomness, such that the attribute-dependent part can still be generated with respect to the randomness chosen by the other authorities, without directly transferring any random coin used to create the attribute-independent part.

[5]While it is rare, this part can be absent from the ciphertext.
[6]Note that this is a common requirement in existing ABE schemes and even MA-ABE schemes [5, 17].

Definition 7. *An ABE scheme supports master key distribution/extension if it possesses of the following properties:*

1. *(Inter-dependency of the User Secret Key:) There exists a polynomial time algorithm* AGen *such that, for every* K_i *in the attribute-dependent part of the user secret key, it can be computed from* AGen *by taking the following two inputs, namely, the attribute-dependent part of the master secret key* ask_i*, and the attribute-independent part of the same user secret key* $\{L_k\}$*, i.e.,*

$$K_i \xleftarrow{\$} \mathsf{AGen}(\mathsf{ask}_i, \{L_k\}),$$

where $\langle \{K_i\}, \{L_k\} \rangle \xleftarrow{\$} \mathsf{KGen}(\mathsf{gsk}, \{\mathsf{ask}_i\}, \mathbb{A})$*,* $i \in \mathbb{A}$*.*

2. *(Homomorphic Properties of the User Secret Key:) For every* $\mathsf{gsk}, \mathsf{gsk}' \in \mathbb{G}_2$*,* $\{\mathsf{ask}_i\}$*, attribute sets* $\mathbb{A}$*,* $\mathsf{id} \in \mathcal{ID}$*, and randomness* u*,* u'*, we have*

$$\begin{aligned} & \mathsf{KGen}((\mathsf{gsk}, \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u) \\ \cdot \quad & \mathsf{KGen}((\mathsf{gsk}', \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u) \\ = \quad & \mathsf{KGen}((\mathsf{gsk} \cdot \mathsf{gsk}', \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u), \\ \textit{and} \quad & \mathsf{KGen}((\mathsf{gsk}, \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u) \\ \cdot \quad & \mathsf{KGen}((\mathsf{gsk}, \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u') \\ = \quad & \mathsf{KGen}((\mathsf{gsk}, \{\mathsf{ask}_i\}), \mathbb{A}, \mathsf{id}; u + u'). \end{aligned}$$

3. *(Homomorphic Property of the Encapsulated Key:) For every* $\mathsf{gpk}, \mathsf{gpk}' \in \mathbb{G}_T$*, policy* $\mathcal{P}$*, and randomness* s*, if*

   - $(\mathfrak{C}, \mathfrak{K}_0) \leftarrow \mathsf{Encap}(\mathsf{gpk}, \mathcal{P}; s)$*,*
   - $(\mathfrak{C}, \mathfrak{K}_1) \leftarrow \mathsf{Encap}(\mathsf{gpk}', \mathcal{P}; s)$*, and*
   - $(\mathfrak{C}, \mathfrak{K}) \leftarrow \mathsf{Encap}(\mathsf{gpk} \cdot \mathsf{gpk}', \mathcal{P}; s)$*;*

   then $\mathfrak{K} = \mathfrak{K}_0 \cdot \mathfrak{K}_1$*. This implicitly requires that the encapsulated key* $\mathfrak{K}$*, is only dependent on the attribute-independent part of master key* gpk*.*

6.1 Construction

- Setup:
  1. Given the security parameter, the bilinear groups context param is established via the Setup algorithm of the underlying ABE.
  2. According to param, each authority j executes the master key generation part of Setup algorithm of the underlying ABE to obtain master public key $(\mathsf{gpk}_j, \{\mathsf{apk}_i\})$ and master private key $(\mathsf{gsk}_j, \{\mathsf{ask}_i\})$.
  3. The resulting gpk of our MA-ABE will be set to $\prod_j \mathsf{gpk}_j$.
  4. All authorities execute the setup stage of the distributed PRF [5] as described in Section 4.3. After this setup, each authority j obtains the secret seed to compute $F_j(\cdot)$, as defined in Section 4.3.

     Here, we utilize a set of DPRF's with $\mathbb{G}_1$ as its range. The instantiation of such DPRF under XDH assumption has also been previously used [5].
  5. Each authority also sets up a key pair for a secure signature scheme, such that any signature from one authority can be verified by another authority. For example, the public key for the signature scheme can be certified by the CA.
- KGen:
  1. For user id, each authority j uses its own gsk_j, and its own randomness u_j, to generate $\{L_{k,j}\}$, the attribute-independent part of usk, *i.e.*,

     $$\{L_{k,j}\} \xleftarrow{\$} \mathsf{KGen}(\mathsf{gsk}, \mathsf{id}; u_j).$$

     For each group element in the above set, authority j blinds it with $F_j(\mathsf{id}||k)$ where $||$ denotes string-concatenation, *i.e.*, $\{L_{k,j} \cdot F_j(\mathsf{id}||k)\}$. Finally, it is signed and the signature is also sent to the user.
  2. In the second round of this protocol, the user has collected the set of $\{L_{k,j}\}$ for all $j \in [N]$.

     The user then combines the response from each authority and outputs $\{\prod_{j \in [N]} L_{k,j} \cdot F_j(\mathsf{id}||k)\}$, which will result in set $\{L_k\}$ due to the DPRF. The user passes the set of $\{L_{k,j}\}$ for all $j \in [N]$ and the signatures collected to each authority.
  3. Upon successful verification of the signature, which means each authority j obtained an authenticated copy of $\{L_k\}$, it then uses the $\{\mathsf{ask}_i\}$ part of its own master secret key gsk_j to generate the attribute-dependent part of user key for each i, *i.e.*,

     $$K_i \xleftarrow{\$} \mathsf{AGen}(\mathsf{ask}_i, \{L_k\}).$$

  4. The user then combines the response from each authority and outputs

     $$\langle \{L_k\}, \cup_{j \in [N]} \{K_i\} \rangle.$$
- Encap and Decap remain the same.

6.2 Analysis

6.2.1 Correctness Analysis

- First, by the construction of DPRF, for each k we have

$$\prod_{j \in [N]} L_k \cdot F_j(\mathsf{id}||k) = \prod_{j \in [N]} L_k.$$

- By construction, $\prod_{j \in [N]} (\mathsf{gpk}_j)$ is gpk. With the first homomorphic property of the user secret key, the role of $\prod_{j \in [N]} L_k$ with respect to gpk is similar to that of $L_{k,j}$ with respect to gpk_j for the j^{th} copy of the single-authority ABE. A user thus recovers the attribute-independent part of user secret key with respect to gpk.
- Different authorities can use different randomness u_j in their key generation. The second homomorphic property of user secret key is crucial here, such that when the keys are multiplied together, $\sum u_j$ will play the role of the randomness as in the underlying ABE scheme.
- Now we turn our focus to the attribute-dependent part $\{K_i\}$, which will probably be determined by the random factor u. Yet, the $\{L_{k,j}\}$ component from each authority j will also have u_j embedded. With the inter-dependency of the user secret key, the set $\{L_{k,j}\}$ serves as useful material embedding information of $u = \sum_{j \in [N]} u_j$. By using the AGen algorithm, the $\{K_i\}$ parts can be created without learning $\{u_j\}$ explicitly.

6.2.2 Security Analysis

Intuitively, security follows from the DPRF-based security proof strategy [5] and a specific property which we require from the underlying ABE schemes for AGen. Namely, we require that the (part of the) key to be generated by the AGen algorithm can be simulated by the simulator for the security proof of the underlying scheme. While it seems that we cannot do a "clean" security reduction and need to look into the details of the underlying simulator, we expect that the use of DPRF can help us to ensure such a property.

It is often the case that, the simulator in the security proof for an ABE construction can return a correct key, as long as the key can be seen as chosen uniformly at random (recall that our formulation requires a probabilistic key generation algorithm). The constraint is that the exact choice of the randomness is unknown. In pairing-based constructions, such randomness is in the form of discrete logarithms, which is unknown to the simulator and cannot be revealed to the adversary. That is the reason our framework requires an additional AGen algorithm, which serves as a "carrier" of the randomness contributed by other authorities.

Our security reduction requires that, no matter what are the intermediate outputs (*i.e.*, the attribute-independent part $\{L_k\}$) from all the other corrupted authorities controlled by the adversary, we can still simulate the scheme.

The use of DPRF allows us to match with the key output by the underlying simulator by "adjusting" the term accordingly. Specifically, the required part of the key to be returned as an intermediate output can be obtained via dividing the key from the underlying simulator by the corresponding $(N-2)$ key components output by adversary. Such adjustment looks indistinguishable to the adversary due to the pseudorandomness of the DPRF.

For an adversary which corrupt $(N-1)$ of the authorities, in the static corruption model, we can set the remaining honest authority to be the one which executes the underlying single-authority ABE. The simulation can guess the correct choice of the honest authority with probability of $1/N$.

6.2.3 Stronger Security Model

For security against a stronger model, where the honest authority is not necessarily the last one to "complete" the user key generation, we need to leverage a universally composable commitment scheme [3] with an extra round of communication for key generation. The commitment key is generated as part of the system setup. The extra round occurs at the beginning, which requires each authority to commit to the attribute-independent part $\{L_k\}$, and send the commitment to the users instead. In the next round, the committed value $\{L_k\}$ is revealed. Upon successful verification, the user proceeds with the rest of the protocol as usual.

In the security proof, the simulator owns the trapdoors for equivocation and extraction. Even in this special mode, the commitment key is indistinguishable to the adversary due to the security of the commitment scheme. The rest is the usual trick which extracts the committed values from the authorities controlled by the adversary, adjusts the set $\{L_k\}$ accordingly, and uses the equivocation feature to explain the commitment for the case that it opens to "another" value which fits with the adjustment.

Due to the requirement of having AGen inside KGen to be dependent on the first-round output of all the authorities, the resulting KGen protocol requires more than one round. For security, concurrent execution should not be performed.

6.3 Discussion

While previous MA-ABE schemes have used the DPRF techniques [5, 19], our usage allows the attributes in a clause of the policy to be cross-authority; and without relying on the random oracle [17, 23].

One may consider our framework to be a generalization of the specific scheme proposed by Liu *et al.* [21]. However, their setting considers a set of "central authorities" which goes beyond the role of a typical certificate authority and needs to be involved in the user decryption key generation.

7. ATTRIBUTE-LEVEL REVOCATION

The key idea for enabling attribute revocation is that, the partition structure of the master key carries over to the user secret key and ciphertext, *i.e.*, both the user secret key and the ciphertext can also be partitioned into two parts. The first part is derived from the attribute-independent key and the second part is derived from the attribute-dependent key.

To revoke a certain attribute, we can update the corresponding attribute key by exponentiating it with a random factor. This change should be propagated to the ciphertexts associated with that attribute, and the user secret keys which we are not going to revoke. We just apply the random exponent for the update on the ciphertext, and apply the inverse of the same exponent to the secret key component. For decryption, the pairings pair-up a ciphertext component and a secret key component, so the decryption results remain the same for non-revoked users. This allows an updated ciphertext to be decryptable by an updated key.

Our attribute revocation extension only applies on ABE schemes with small universe of attributes, and can be seen as a generalization of an existing scheme [30].

7.1 ABE with Attribute-Level Revocation

We define the following ABE with partition for extension to support efficient attribute revocation.

Definition 8. *A pairing-based ABE scheme is said to be revocable if it possess of the following properties:*

1. *The scheme is ciphertext-partitionable (Definition 5).*
2. *(Homomorphic Property of Attribute-Dependent Part of the Ciphertext:) Suppose the ciphertext $\mathfrak{C}$ is generated by the encapsulation algorithm which takes the master key $(\mathsf{gpk}, \{\mathsf{apk}_i\})$ and a policy $\mathcal{P}$ as input, i.e.,*

$$\mathfrak{C} \xleftarrow{\$} \mathsf{Encap}((\mathsf{gpk}, \{\mathsf{apk}_i\}), \mathcal{P}),$$

if $\{D_i\}$ is the attribute-dependent part of $\mathfrak{C}$, then there is a (not necessarily proper) subset $\{D_{i,0}\}$ of $\{D_i\}$, such that for a random $r \in \mathbb{Z}_p$, $\{D_{i,0}\}^{1/r}$ follows the same distribution as the same subset of the attribute-dependent part of

$$\mathfrak{C}' \xleftarrow{\$} \mathsf{Encap}((\mathsf{gpk}, \{\mathsf{apk}_i\}^r), \mathcal{P}).$$

(We remark that one can generalize this definition to require $\{D_{i,0}\}^{f(r)}$ instead, where $f(r) := 1/r$, and replace it with other publicly computable function.)

3. *(Public Randomizability of the Ciphertext:) There exists a probabilistic polynomial time algorithm* $\mathsf{Rand}()$ *which takes in a ciphertext and outputs a randomized version of it following the same distribution as the output of* Encap*, but encapsulating the same key as the input ciphertext.*

7.2 Construction

We follow the framework implicitly defined in Section 3.3, which supplements an ABE scheme with three algorithms:

- To perform the system-update SU, for each attribute i, the authority controlling it picks a random element r_i from $\mathbb{Z}_p$, then, using the attribute-dependent component of the existing master key, computes $\mathsf{apk}_i^{r_i}$ as the attribute-dependent component of the new master public key. The corresponding attribute-dependent component of the new master secret key will be $\mathsf{ask}_i \cdot r_i$.
- To perform the key-update KU, if the user is entitled to have the same attribute i in the next time period, the K_i component will be updated to $K_i^{r_i}$.
- To perform the ciphertext-update CU, take the old ciphertext, re-randomize it by $\mathsf{Rand}()$, then update the attribute-dependent component $\{D_i\}$ of the re-randomized ciphertext by setting those to $\{D_i^{1/r_i}\}$.

7.3 Analysis

Correctness of all the updates can be easily seen from all the homomorphic properties we require. Note that the update of the ciphertext is possible since we require every ciphertext to be accompanied by the policy. Correctness of decryption is ensured by the way we perform the updates of ciphertext and private key, the format of the decryption as a pairing product equation, and the bilinearity of the pairing function which makes the corresponding exponents from the ciphertext and the private key cancel each other.

Security analysis follows from a reductionist argument to the security of the underlying scheme. Note that we are considering a static corruption model. Our construction basically replicates the attribute set supported by the basic scheme into multiple copies, one supporting each time domain. It is easy to simulate the user secret key, given there are only polynomially many time periods. For challenge ciphertext, it is about truncating a long ciphertext to one which only contains the policy related to the challenge time period. By applying only the exponent $1/r_i$, with respect to the same attribute, parts of the ciphertext for different time period may be correlated. The public re-randomization procedure of the ciphertext breaks this possible correlation.

8. OUTSOURCED DECRYPTION

With our extension for multi-authority and revocation, outsourced decryption is readily available. Specifically, the user will take the role of the $(N+1)^{\text{th}}$ authority, and delegate the part of the key from the N authorities to the mediator.

8.1 Construction

1. User randomly generates $(\mathsf{gpk}', \mathsf{gsk}')$ as in Setup.
2. User delegates to the mediator $\mathsf{usk}' = \langle \{L_k \cdot L_i'\}, \{K_i\} \rangle$ where $\mathsf{usk} = \langle \{L_k\}, \{K_i\} \rangle$ is the original decryption key, and $\{L_i'\} \xleftarrow{\$} \mathsf{KGen}(\mathsf{gsk}', \mathsf{id})$ will be kept as a secret.
3. Mediator decrypts $\mathfrak{C}$ by running $\mathfrak{K}' \leftarrow \mathsf{Decap}(\mathsf{usk}', \mathfrak{C})$. (Same as the last extension, we also expect a non-anonymous ABE scheme, in which the ciphertext is always appended with its associated policy.)
4. User computes $\frac{\mathfrak{K}'}{\prod \hat{e}(L_k', C_k)}$ to get the encapsulated key.

8.2 Analysis

Correctness is easy to see from the homomorphic property of the encapsulated key.

The user side computation is efficient. Most of the computations are outsourced since the user side decryption is as efficient as that of a single-authority ABE which "involves no attribute" at all, *i.e.*, only the attribute-independent part of the key for the last authority "run" by the user is used. Note that the number of pairing required is the number of elements in $\{L_k\}$. This is a scheme-specific parameter, and is independent of the number of attributes owned by the user, or the number of authorities. For example, the instantiation to be presented in the next section only requires two pairing operations (since the set only contains $\{L_0, L_1\}$). On the other hand, most pairing-based ABE constructions which support fine-grained access control policy often require a number of pairing operations which is linear in the number of attributes (except those restricted to conjunctive policy, which often allow ciphertext parts related to different attributes to be aggregated together).

Security follows from that of MA-ABE since we are conceptually introducing one more authority. Without the "last" authority, which is the user in this case, security preserves even if all the other authorities collude.

8.3 Discussion

The original work by Green *et al.* [12] on outsourcing the decryption of ABE ciphertexts extended two specific ABE constructions, one is from the collection of CP-ABE schemes by Waters [28] and another is the LSSS-generalization of the original KP-ABE scheme by Goyal *et al.* [11]. A major trick there is similar to what we did in the last section for updating the keys for a particular attribute. In more details, every term in the original decryption key will be exponentiated with the same random factor, and the result will be served as the outsourced key. The random factor will then be kept as the user-side secret for completing the decryption.

While this makes the user side decryption as efficient as a regular ElGamal decryption; the security proofs for both schemes need to be re-done from scratch [12]. Their basic constructions are proven in the selective model. It was mentioned [12] that the dual system encryption framework (*e.g.*, [17]) can be used for adaptively-secure extension. However, it was also acknowledged [12] that any security argument for such approach is difficult to be a black box reduction to the underlying scheme.

The idea of having an extra "default" authority can be seen as an abstraction of an existing idea of "default attribute" [18]. Yet, there are two important differences. First, the previous work [18] essentially assumes a setting which the trusted authority generates both the outsourced key and the user side secret (as far as the context of this paper is concerned). Moreover, only a specific construction was given [18]. Their construction conceptually splits the single authority into two for each decryption key generation. This approach requires proving the whole scheme from scratch.

9. PUTTING IT ALL TOGETHER

9.1 Sample Schemes within Our Framework

Our case study takes Waters efficient and expressive CP-ABE scheme [28]. We remark that all but one of the schemes there [28] can fit with our framework[7].

An adaptively-secure CP-ABE scheme based on composite order groups [16] has strong resemblance to a scheme in the work of Waters [28] as acknowledged by the authors [16]. This scheme [16] also fits with our framework. The resulting MA-ABE scheme naturally resembles to the specific MA-ABE scheme by Liu *et al.* [21], although in a different system model as discussed in Section 6.3.

Some schemes may satisfy one of our extensions but not the others. As an example, the threshold ABE scheme of Li *et al.* [18], which already support outsourced decryption, can be extended for attribute-level revocation.

- Setup:

  1. Pairing-groups parameterized by param are first chosen according to the security parameter λ.
  2. A group-$\mathbb{G}_1$ element g_1^a is randomly chosen. Note that the knowledge of $a \in \mathbb{Z}_p$ is never required in any step in any of the algorithms below, except in the form of g_2^a in KGen.
  3. The master key for authority j consists of:
     (a) An attribute-independent part: $\mathsf{gsk}_j = g_2^{\alpha_j}$ where $\alpha_j \in \mathbb{Z}_p$, $\mathsf{gpk}_j = \hat{e}(g_1, g_2)^{\alpha_j}$.
     (b) An attribute-dependent part: For each attribute i, $\mathsf{ask}_i = \beta_i$ where $\beta_i \in \mathbb{Z}_p$, $\mathsf{apk}_i = h_i = g_1^{\beta_i}$. We may also pre-compute $h'_i = g_2^{\beta_i}$ as part of the attribute secret key. Note that $h'_i = \psi^{-1}(h_i)$.

  The "partition of master keys" into gpk and $\{\mathsf{ask}_i\}$ has already been done. System-Update SU can be easily done too.

- $\mathsf{Encap}(\mathsf{gpk}, \{\mathsf{apk}_i\}, \mathcal{P}; s)$: We first describe the original Encap algorithm of Waters-ABE.

  1. The policy is expressed in the form of an LSSS access structure $(\mathbb{M}, \rho)$. $\mathbb{M}$ is an $l \times n$ LSSS matrix and ρ maps each row of $\mathbb{M}$ to an attribute $\rho(x)$.
  2. This algorithm chooses a random vector $\vec{v} \in \mathbb{Z}_p^n$ denoted by $(s, v_2, \cdots, v_n)$, where s is a secret exponent used in encapsulated key $\mathfrak{K}$.
  3. For each row vector $\mathbb{M}_i$, this algorithm calculates $\lambda_i = \mathbb{M} \cdot \vec{v}$ and chooses random $t_i \xleftarrow{\$} \mathbb{Z}_p$.
  4. The ciphertext is
     $$\mathfrak{C} = \langle C_0 = g_1^s, \{D_{i,0} = g_1^{t_i}, D_{i,1} = g_1^{a\lambda_i} h_{\rho(i)}^{-t_i}\}_{\forall i \in [\ell]} \rangle$$
     and the encapsulated key is $\mathfrak{K}_j = \mathsf{gpk}_j^s$.

  To see how it fits with our framework:

  – Knowing the policy $(\mathbb{M}, \rho)$, it is easy to re-randomize the ciphertext by $\{D_{i,0} \cdot g_1^{t'_i}, D_{i,1} h_{\rho(i)}^{-t'_i}\}$ for $t'_i \in \mathbb{Z}_p$.
  – As we have already partitioned, C_0 is the attribute-independent part and $\{D_{i,0}, D_{i,1}\}$ is the attribute-dependent part.
  – For the homomorphic property of the attribute-dependent part, we only concern its subset which is just $D_{i,0}$. To see, when $\mathsf{apk}_i = h_i$ is exponentiated to $\mathsf{apk}'_i = \mathsf{apk}_i^r = h_i^r$. Keeping the same $D_{i,1}$ will implicitly change t_i to $t'_i = t_i/r$, *i.e.*,
    $$D_{i,1} = g_1^{a\lambda_i} h_i^{-t_i} = g_1^{a\lambda_i} (h_i^r)^{-t_i/r}.$$
    The attribute-dependent part $D_{i,0}$ can then be exponentiated with $1/r$ to obtain $D_{i,0}^{1/r} = g_1^{t_i/r} = g_1^{t'_i}$ to match with the t'_i in $D_{i,1}$. Note that it satisfies the distribution for a ciphertext w.r.t. $\mathsf{apk}'_i = \mathsf{apk}_i^r = h_i^r$.
    This also explains how ciphertext-update CU works.
  – The homomorphic property regarding the encapsulated key is easy to be seen.

- KGen: The key generation algorithm takes in the master secret key of an authority and a set of attribute keys of the authority; outputs $\mathsf{usk}_{\mathsf{id},j}$ defined by:
  $$\langle L_0 = \mathsf{gsk}_j \cdot g_2^{au_j} = g_2^{\alpha_j} g_2^{au_j}, L_1 = g_2^{u_j}, \{K_x = {h'_x}^{u_j}\} \rangle$$
  where $\{x\}$ is the set of attributes for user named id, and u_j is randomly chosen from $\mathbb{Z}_p$.

  To see how it fits with our framework:

  – As already partitioned, $\{L_0, L_1\}$ is the attribute-independent part, the set $\{K_x\}$ is the attribute-dependent part. With this structure, the key can be easily updated via the key-update KU.
  – For homomorphic properties of user secret key,
    $$\begin{aligned} L_{0,j} \cdot L_{0,k} &= (\mathsf{gsk}_j \cdot g_2^{au_j}) \cdot (\mathsf{gsk}_k \cdot g_2^{au_k}) \\ &= (g_2^{\alpha_j} g_2^{au_j}) \cdot (g_2^{\alpha_k} g_2^{au_k}) \\ &= g_2^{(\alpha_j + \alpha_k)} g_2^{a(u_j + u_k)}, \\ L_{1,j} \cdot L_{1,k} &= g_2^{u_j} \cdot g_2^{u_k} = g_2^{(u_j + u_k)}, \\ K_{x,j} \cdot K_{x,k} &= {h'_x}^{u_j} \cdot {h'_x}^{u_k} = {h'_x}^{(u_j + u_k)}. \end{aligned}$$
  – For inter-dependency, AGen outputs $L_1^{\beta_x} = g_2^{u_j \beta_x} = {h'_x}^{u_j} = K_x$.

- Decap: The decryption algorithm takes in a ciphertext encrypted under the access structure of $(\mathbb{M}, \rho)$, and a decryption key for the set $\mathbb{A}$.

  Let $\mathbb{I}$, be a subset of $\{1, 2, \cdots, \ell\}$ as defined by $\mathbb{I} = \{i : \rho(i) \in \mathbb{A}\}$. The algorithm computes constants $\omega_i \in \mathbb{Z}_p$ such that $\sum_{i \in \mathbb{I}} \omega_i \mathbb{M}_i = (1, 0, \cdots, 0)$, which is possible, and may not be unique, when $\mathbb{A}$ satisfies $(\mathbb{M}, \rho)$.

  Then it decrypts the ciphertext by recovering the encapsulated key as
  $$\mathfrak{K} = \frac{\hat{e}(C_0, L_0)}{\prod_{i \in \mathbb{I}} (\hat{e}(D_{i,1}, L_1) \hat{e}(D_{i,0}, K_{\rho(i)}))^{\omega_i}},$$
  which is obviously a pairing production equation.

[7] The remaining one can also be fitted if we further generalize our framework, which we decided not to, for better understanding of the essence by avoiding definitional clumsiness.

9.2 Our Instantiation

With reference to Section 9.1, we spell out how to instantiate our framework by using Waters CP-ABE [28].

- Setup: It follows directly from our framework and the Setup of the underlying ABE scheme. How $\{\mathsf{apk}_i/\mathsf{ask}_i\}$ can be updated also follows exactly as our framework.
- Encap: Different from the underlying Encap algorithm is that, for each attribute i, one needs to take the term $\mathsf{apk}_i = h_i$ from the master key of the corresponding AA's. The resulting ciphertext is

  $$\mathfrak{C} = \langle C_0 = g_1^s, \{D_{i,0} = g_1^{t_i}, D_{i,1} = g_1^{a\lambda_i} h_{\rho(i)}^{-t_i}\}_{\forall i \in [\ell]} \rangle.$$

  The encapsulated key is defined accordingly as:

  $$\mathfrak{K} = \prod_j \hat{e}(g_1, g_2)^{\alpha_j s} = \hat{e}(g_1, g_2)^{\sum_j \alpha_j s} = \hat{e}(g_1, g_2)^{\alpha s}.$$

  Both outputs are in the same format as the underlying ABE scheme. Re-randomization is easy and we omit Rand(). Update CU can be carried out by only updating the $\{D_{i,0}\}$ part as explained in the last subsection.
- KGen:
  1. User named id interacts with each authority j and obtains $\langle L_{0,j}, L_{1,j} \rangle$ which is defined by:

     $$\begin{aligned} L_{0,j} &= F_j(\mathsf{id}||0) \cdot \mathsf{gsk}_j \cdot g_2^{au_j} = F_j(\mathsf{id}||0) \cdot g_2^{\alpha_j} g_2^{au_j} \\ L_{1,j} &= F_j(\mathsf{id}||1) \cdot g_2^{u_j}. \end{aligned}$$

  2. User id eventually obtains:

     $$\langle L_0 = \prod_{j \in [N]} L_{0,j} = g_2^{\alpha} g_2^{au}, L_1 = \prod_{j \in [N]} L_{1,j} = g_2^{u} \}\rangle.$$

     where $\alpha = \sum_{j \in [N]} \alpha_i$ and $u = \sum_{j \in [N]} u_j$.
  3. User id sends L_1 to each authority j and obtains

     $$\{K_x = L_1^{\beta_x} = g_2^{u\beta_x} = {h'_x}^u\},$$

     which computes the keys $\langle L_0, L_1, \{K_x\} \rangle$.

  The involvement of signature is omitted here. Again, this is in the same format as the underlying ABE scheme. Key-update KU can be applied on $\{K_x\}$.
- Decap: With the same format of key and ciphertext, normal decryption is the same as the underlying Decap.

  We do not repeat the details of outsourcing a decryption key and how to perform the corresponding partial decryption (which is the same as the Decap) and final decryption (which is just one pairing operation).

9.3 Security Analysis

Security follows from our generic framework. As mentioned, we need to argue that the key constructed from AGen algorithm can match with what is constructed by the simulator for proving the security of the underlying ABE scheme. The underlying simulator will output a key for a specific random exponent, say u^*. Note that we need to match with the random factor u in $L_1 = \prod_{j \in [1,N]} L_{1,j} = \prod_{j \in [1,N]} g_2^{u_j}$. To simulate the L_1 term, the authority of concern, say k^*, can set $u_{k^*} = g_2^{u^*} / \prod_{j \in [1,N] \setminus \{k^*\}} g_2^{u_j}$. For the L_0 term, it appears to require the knowledge of the discrete logarithm a. However, this can be circumvented as explained in the analysis of our framework in Section 6.2.2. Specifically, the L_{0,k^*} component can be just a random group element, such that when all $L_{0,j}$'s are multiplied together, it will match with the key component L_0 returned by the underlying simulator.

10. EVALUATING A CRYPTOSYSTEM

We want to make two remarks regarding performance evaluation. While performance analysis on a prototype appear to be crucial to demonstrate the deployability in general, these analyses may not add much retention value for scientific use for cryptosystems which are neither sensitive to the system model (*e.g.*, many rounds of interaction between possibly dynamic number of entities) nor to the input (*e.g.*, the distribution of the plaintext to be encrypted). Specifically, for ABE, the interaction requires at most a constant number of rounds. Also, it is typical to use hybrid encryption such that ABE just encapsulates a symmetric-key, where the possibly large plaintext is encrypted by a fast symmetric-key encryption scheme. As a result, counting the number of operations almost directly suggests the computation time.

Another remark is that, the same cryptosystem design can always be instantiated with different security parameter, which should be chosen according to the tightness of the proof and the underlying assumption. As clearly demonstrated in the class of ABE schemes of Waters [28], apparently more efficient schemes can be built by making stronger assumptions. A stronger assumption means that it can be easier to solve the underlying hard problem, and hence the scheme should be instantiated with a larger security parameter. Doing so probably offset the efficiency gain of the newer design. Unfortunately, quite a few existing performance evaluations of cryptosystems, mostly from non-cryptography venues, did not take these into considerations. This renders the so-called performance analysis rather superficial.

To conclude, one should always consider the assumptions, no matter the performance evaluation is order analysis or benchmark. Evaluation should be done with security parameters chosen appropriately according to the assumptions.

11. CONCLUSION AND FUTURE WORK

We propose an abstraction of ABE which can be extended to support a set of nice features desirable for practical usage of ABE, namely, multi-authority extension, attribute-level revocation, and outsourced decryption. Instead of tailor-making a construction we define a set of properties which explains the correctness and security of our approach.

Our work opens an avenue of research in designing new extension frameworks for ABE. Specific goals include a new framework for features such as online/offline ABE [13]; or for the same set of features but with some requirements removed, for example, setting up the authorities in advance, or appending the access control policy to the ciphertext. We also note that for user-level revocation, sub-linear size of keying materials could be possible. However, for attribute-level revocation, how to reduce the key-update materials appears to be challenging, especially in the multi-authority setting. Another problem is to investigate if outsourced decryption for anonymous ABE can be done, in which the policy associated with the ciphertext remains unknown to any mediator.

Acknowledgement

The author would like to thank Hannes Hartenstein, Russell W. F. Lai, Ying-Kai Tang, and anonymous reviewers for their helpful comments which improve the paper.

Sherman Chow is supported by the Direct Grant (4055018) of the Chinese University of Hong Kong, Early Career Award and the Early Career Scheme (CUHK 439713), and General Research Funds (CUHK 14201914) of the Research Grants Council, University Grant Committee of Hong Kong.

12. REFERENCES

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-Policy Attribute-Based Encryption. In *IEEE Security and Privacy*, pages 321–334, 2007.

[2] A. Boldyreva, V. Goyal, and V. Kumar. Identity-Based Encryption with Efficient Revocation. In *ACM Computer and Communications Security*, pages 417–426, 2008.

[3] R. Canetti and M. Fischlin. Universally Composable Commitments. In *CRYPTO*, pages 19–40, 2001.

[4] M. Chase. Multi-Authority Attribute Based Encryption. In *TCC*, pages 515–534, 2007.

[5] M. Chase and S. S. M. Chow. Improving Privacy and Security in Multi-Authority Attribute-Based Encryption. In *ACM Computer and Communications Security*, pages 121–130, 2009.

[6] S. S. M. Chow. *New Privacy-Preserving Architectures for Identity-/Attribute-Based Encryption*. PhD thesis, New York University, 2010.

[7] S. S. M. Chow, Y. J. He, L. C. K. Hui, and S.-M. Yiu. SPICE - Simple Privacy-Preserving Identity-Management for Cloud Environment. In *ACNS*, pages 526–543, 2012.

[8] S. S. M. Chow, J. K. Liu, and J. Zhou. Identity-Based Online/Offline Key Encapsulation and Encryption. In *ACM AsiaCCS*, pages 52–60, 2011.

[9] S. S. M. Chow, V. Roth, and E. G. Rieffel. General Certificateless Encryption and Timed-Release Encryption. In *Security and Cryptography for Networks (SCN)*, pages 126–143, 2008.

[10] A. Ge, J. Zhang, R. Zhang, C. Ma, and Z. Zhang. Security Analysis of a Privacy-Preserving Decentralized Key-Policy Attribute-Based Encryption Scheme. *IEEE Trans. Parallel Distrib. Syst.*, 24(11):2319–2321, 2013.

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

[12] M. Green, S. Hohenberger, and B. Waters. Outsourcing the Decryption of ABE Ciphertexts. In *USENIX Security*, 2011.

[13] S. Hohenberger and B. Waters. Online/Offline Attribute-Based Encryption. In *Public-Key Cryptography (PKC)*, pages 293–310, 2014.

[14] J. Hong, K. Xue, and W. Li. Comments on "DAC-MACS: Effective Data Access Control for Multiauthority Cloud Storage Systems". *IEEE Trans. Information Forensics and Security*, 10(6):1315–1317, 2015.

[15] S. Jahid, P. Mittal, and N. Borisov. EASiER: Encryption-Based Access Control in Social Networks with Efficient Revocation. In *ACM AsiaCCS*, pages 411–415, 2011.

[16] A. B. Lewko, T. Okamoto, A. Sahai, K. Takashima, and B. Waters. Fully Secure Functional Encryption: Attribute-Based Encryption and (Hierarchical) Inner Product Encryption. In *EuroCrypt*, pages 62–91, 2010.

[17] A. B. Lewko and B. Waters. Decentralizing Attribute-Based Encryption. In *EuroCrypt*, pages 568–588, 2011.

[18] J. Li, X. Chen, J. Li, C. Jia, J. Ma, and W. Lou. Fine-Grained Access Control System Based on Outsourced Attribute-Based Encryption. In *ESORICS*, pages 592–609, 2013.

[19] J. Li, Q. Huang, X. Chen, S. S. M. Chow, D. S. Wong, and D. Xie. Multi-Authority Ciphertext-Policy Attribute-Based Encryption with Accountability. In *ACM AsiaCCS*, pages 386–390, 2011.

[20] M. Li, S. Yu, Y. Zheng, K. Ren, and W. Lou. Scalable and Secure Sharing of Personal Health Records in Cloud Computing Using Attribute-Based Encryption. *IEEE Trans. Parallel Distrib. Syst.*, 24(1):131–143, 2013.

[21] Z. Liu, Z. Cao, Q. Huang, D. S. Wong, and T. H. Yuen. Fully Secure Multi-authority Ciphertext-Policy Attribute-Based Encryption without Random Oracles. In *ESORICS*, 2011.

[22] H. Ma, R. Zhang, and W. Yuan. Comments on "Control Cloud Data Access Privilege and Anonymity with Fully Anonymous Attribute-Based Encryption". *IEEE Trans. Information Forensics and Security*, 11(4):866–867, 2016.

[23] Y. Rouselakis and B. Waters. Efficient Statically-Secure Large-Universe Multi-Authority Attribute-Based Encryption. In *Financial Cryptography and Data Security*, pages 315–332, 2015.

[24] A. Sahai, H. Seyalioglu, and B. Waters. Dynamic Credentials and Ciphertext Delegation for Attribute-Based Encryption. In *CRYPTO*, pages 199–217, 2012.

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

[26] Y. Tong, J. Sun, S. S. M. Chow, and P. Li. Cloud-Assisted Mobile-Access of Health Data With Privacy and Auditability. *IEEE J. Biomedical and Health Informatics*, 18(2):419–429, 2014.

[27] M. Wang, Z. Zhang, and C. Chen. Security Analysis of a Privacy-Preserving Decentralized Ciphertext-Policy Attribute-Based Encryption Scheme. *Concurrency and Computation: Practice and Experience*, 28(4):1237–1245, 2016.

[28] B. Waters. Ciphertext-Policy Attribute-Based Encryption: An Expressive, Efficient, and Provably Secure Realization. In *Public Key Cryptography*, pages 53–70, 2011.

[29] K. Yang, X. Jia, K. Ren, and B. Zhang. DAC-MACS: Effective Data Access Control for Multi-Authority Cloud Storage Systems. In *Infocom*. IEEE, 2013.

[30] S. Yu, C. Wang, K. Ren, and W. Lou. Attribute Based Data Sharing with Attribute Revocation. In *ACM AsiaCCS*, pages 261–270, 2010.