Secure Fine-Grained Access Control of Mobile User Data through Untrusted Cloud

Kai Zhou and Jian Ren
Department of ECE, Michigan State University, East Lansing, MI 48824-1226
Email: {zhoukai, renjian}@msu.edu

***Abstract*—Cloud computing enables data owners to outsource their computationally intensive tasks and store private data to the shared cloud. To enhance the security while preserving the flexibility of data sharing, Attribute Based Encryption (ABE) was introduced to provide a fine-grained access control. A key issue in ABE based systems is the high computational overhead, which could be prohibitive for resource constrained mobile devices. In this paper, we propose a scheme to securely and efficiently outsource the computationally intensive access control operations of ABE to the shared cloud, thus reliving the computational burden of mobile users which can greatly improve the battery lifetime. In a high level view, data owners only need to specify access policies on the encrypted data so that access control can be done automatically by the cloud. Our proposed scheme guarantees that it is computationally infeasible for the untrusted cloud to recover the encrypted file and that the cloud is enforced to complete the full functionality of access control, even in situations where the cloud may be compromised by malicious data users. Our theoretical analysis and experiment results both demonstrate that our scheme can achieve high performance gain for resource constrained mobile devices.**

***Index Terms*—Attribute based encryption, fine-grained access control, outsourcing, mobile cloud computing.**

I. Introduction

With the fast development of cloud computing, there is a trend for mobile users to upload and share their data through the cloud platform for unrestricted access regardless of the time and location. A typical scenario would be the Personal Healthcare System [1], where patients upload their personal health records to a public cloud that are accessible to the authorized physicians. In these scenarios, data owners are concerned about the security and privacy of their own data. On one hand, the untrusted cloud should be prevented from knowing the content of data. On the other hand, data owners should control who can access the data.

Attribute Based Encryption (ABE) cryptosystem turns out to be a promising candidate to address the above issue. As a generalization of Identity Based Encryption (IBE) [2], [3], ABE can be categorized into two types: Key-Policy ABE (KP-ABE) and Ciphertext-Policy ABE (CP-ABE) [4]–[6] depending on where the policy is specified. A typical CP-ABE system consists of message senders, message receivers and an Attribute Authority (AA) that is responsible for generating secret keys. Each user (sender or receiver) is tagged with a set of attributes (e.g., {Student, ECE, Enrolled}). In message encryption, the sender can specify an access policy (e.g., {Professor $\wedge$ (ECE $\vee$ CSE)}) associated with the ciphertext. As a result, only those receivers whose attributes (e.g., {Professor, ECE}) satisfy the policy can decrypt the message. In contrast, KP-ABE tags ciphertext with sets of attributes and associates policies with decryption keys. The fact that CP-ABE enables fine-grained access control of encrypted data makes it suitable for various cloud based applications.

However, one concern in ABE is its relatively high computational complexity in encryption and decryption. For instance, message encryption will take two modular exponentiations for each attribute in the policy [5]. And a message receiver will conduct two pairing operations in bilinear groups for each attribute in decryption. As a result, the number of modular exponentiations and pairing operations involved in the encryption and decryption process is linear to the number of attributes in the access policy. The high computational overhead of ABE seems to be prohibitive for resource constrained mobile devices. In some recent research [7], [8], the performance of ABE has been measured on mobile phones. It is shown that, at a security level of 128 bits[1], the average execution time for encryption with 10 attributes on android smartphones is around 10 seconds. Although it is feasible to implement ABE on mobile devices, it is highly inefficient in terms of execution time and energy consumption.

To address this problem, some researchers [10] propose an online/offline computation scheme where the message is pre-encrypted without specifying a policy during the offline phase. Then based on the pre-computed results, mobile users can encrypt messages offline by a specified policy with relatively low computational overhead. Some other research works focus on secure outsourcing of ABE systems [11]–[13]. However, they either focus on a specific access structure [12] or rely on a rather strong security assumption of cloud [13].

In this paper, we propose to outsource the whole access control functionality to the cloud. This is achieved based on a key observation that the message encryption part and access control part are highly detached in ABE. To be specific, the ciphertext of ABE can be roughly divided into a payload section and a control section. The payload section is simply message encryption $\mathsf{Enc}(m, s)$ using a secret key s. The

[1]The description of different security levels can be found in [9]

978-1-5090-2279-3/16/$31.00 ©2016 IEEE

control section is a linear secret sharing scheme that shares the secret key s to certain parties (i.e., attributes). Roughly speaking, in the decryption process, a receiver with certain attribute set that satisfies the policy can recover the secret key s. Thus the receiver can further decrypt the message m.

In our proposed scheme, we let the cloud generate the control section of the ciphertext and a mobile user only needs to conduct message encryption. However, it is not a trivial task. First, to share the secret key s, the cloud has to know the secret beforehand. In this case, the cloud can easily decrypt the payload section. Second, when the cloud is fully in charge of the access control, how to enforce the cloud to deny invalid access is a critical problem. Because in some situations, the cloud may be compromised by malicious users or the cloud may actively collude with data users. We address these problems by exploring the linear property of secret sharing schemes. Basically, the mobile users encrypt message with a secret s_1 while the cloud distributes a different secret s_2 through a secret sharing scheme. The control section in the ciphertext and the decryption algorithm are then modified in such a way that only valid users can recover the secret s_1.

In summary, our proposed scheme enables mobile users to conduct only normal encryption and specify an access control policy on the ciphertext. The access control is then honestly fulfilled by the cloud. Moreover, the cloud will conduct partial decryption based on which the data users can recover the message through normal decryption. In this way, attribute based encryption becomes transparent to mobile users and cloud can provide access control of encrypted data as a service.

The reset of this paper is organized as follows. In Section II, we introduce the system model and threat model. In Section III, we provide a high level overview of the ABE scheme. The detailed construction of our outsourced ABE scheme is given in Section IV. We analyze the security and complexity of the proposed scheme in Section V and present some numeric results in Section VI. We introduce some typical scenarios where our proposed scheme can be applied in Section VII. The paper is concluded in Section VIII.

II. System Model and Threat Model

A. System Model

We consider a file sharing system consisting of four parties: an Attribute Authority (AA), data owners, data users (who could be data users simultaneously) and the cloud. The AA is responsible for setting up the system, authenticating users' attributes and generating secret keys when users request keys with their attributes. Data owners desire to upload their data either for storage or sharing. They also want to specify access policies on the encrypted data in order to control who can access the data. Data users that are tagged with attributes want to decrypt the data they can access by requesting secret keys from the AA. The cloud will provide storage service as well as access control service. It will generate control section for data owners' uploaded data and help to partially decrypt the data when requested by data users. The system model is shown in Fig. 1.

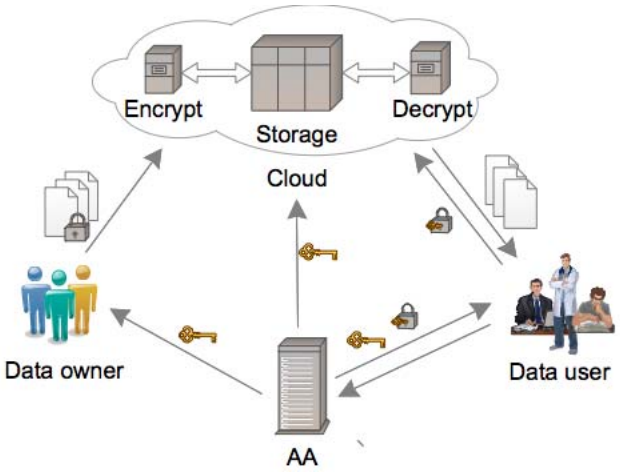

Fig. 1. System Model of Outsourced ABE

Our proposed outsourced system can be modeled as the following functions.

1) $\mathsf{Setup}(\lambda) \rightarrow \{\mathsf{PK}, \mathsf{MSK}\}$: the AA generates public parameter PK and master secret key MSK using a security parameter λ. The public parameter PK is known to all the users as well as the cloud.
2) $\mathsf{KeyGen}(\mathsf{PK}, S) \rightarrow \mathsf{SK}$: when a data user with a set of attributes S requests his secret key, the AA first authenticates that all the attributes in S are valid. Then AA generates and distributes the secret key SK associated with S to the data user through a secure channel. We note that the secret key SK is composed of SK_L and SK_S, where SK_L is kept secret by data user and SK_S is prepared for the cloud for partial decryption.
3) $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$: a data owner encrypts the message m to produce the ciphertext CT_1 and specifies an access policy T. Meanwhile, some auxiliary information is produced for further encryption. $\{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ is outsourced to the cloud.
4) $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$: the cloud enforce the access policy T to produce partial ciphertext CT_2 based on the auxiliary information aux. The ciphertext is formed as $\mathsf{CT} = \{\mathsf{CT}_1, \mathsf{CT}_2\}$ that is stored in the cloud.
5) $\mathsf{Dec}_S(\mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$: when a data user requests to access data CT, he sends the secret key SK_S to the cloud. The cloud decrypts the ciphertext CT to obtain partially decrypted ciphertext CT_S, which is then transmitted to the data user.
6) $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$: the data user decrypts the ciphertext CT_S to obtain the message m using his secret key SK_L.

B. Threat Model

In the system, we assume that AA is fully trusted as is the case in typical ABE systems. We consider a *malicious*

cloud model. On one hand, the cloud is curious in learning the message stored in cloud storage. On the other hand, the cloud can be compromised by malicious users. Invalid users without proper attribute sets may collude with the cloud in order to obtain the access right. The three functional modules (encryption, storage and decryption) are logically considered to be implemented in one single cloud environment. In this case, the cloud will have knowledge of the ciphertext CT as well as the secret key SK_S. We note that one highlight of our scheme is that we do not impose the strong assumption as in most related works that the cloud is only curious-but-honest. Later, we show that our scheme supports public auditing of all the operations conducted by the cloud. As a result, the cloud is enforced to honestly conduct the required operations.

III. A High Level Overview of CP-ABE

In this section, we introduce some necessary preliminaries of CP-ABE. We utilize the scheme in [5] as the underlying CP-ABE scheme for its highly expressive access structure. We emphasize on the realization of access control in ABE and model it as the process of encoding and decoding a secret.

A. Bilinear Pairing

A bilinear pairing is a mapping function $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, where $\mathbb{G}_1$ and $\mathbb{G}_2$ are two multiplicative cyclic groups with prime order p. Let g be the generator of $\mathbb{G}_1$. The bilinear pairing e has the following properties:

1) The pairing e is bilinear. That is, $\forall g_1, g_2 \in \mathbb{G}_0, a_1, a_2 \in \mathbb{Z}_p, e(g_1^{a_1}, g_2^{a_2}) = e(g_1, g_2)^{a_1 a_2}$.
2) The pairing e is non-degenerate. That is $e(g, g) \neq 1$.
3) The paring e and the group operation in $\mathbb{G}_1$ are both efficiently computable.

B. Linear Secret Sharing Scheme

A secret sharing scheme is a method for a secret holder to distribute a secret to different parties and only under certain criteria can the parties reconstruct the secret. A secret sharing scheme is *linear* in that the mapping between a secret and the secret shares are realized by linear functions [14].

A typical construction of linear secret sharing scheme is Shamir's $(t-n)$-threshold secret sharing scheme [15]. In his construction, the secret and secret shares are all elements in a finite field $\mathbb{F}_q$. To share a secret s, the secret holder chooses a polynomial $f(x)$ of degree $(t-1)$ over $\mathbb{F}_q$. The coefficients $c_1, c_2, \cdots, c_{t-1}$ are randomly chosen from $\mathbb{F}_q$ and the constant term is set to be s. A secret share is a point $(a_i, f(a_i))$ on the polynomial, where $a_i \in \mathbb{F}_q$ is randomly generated. The basic idea is that for a polynomial of degree $(t-1)$, if t or more points on the polynomial are given, one can fully reconstruct the polynomial (coefficients) using polynomial interpolation. Suppose there are a total of n secret shares distributed to n parties. If there are t or more parties collude with each other, they can reconstruct the secret. Thus, this scheme is called the $(t-n)$-threshold secret sharing scheme.

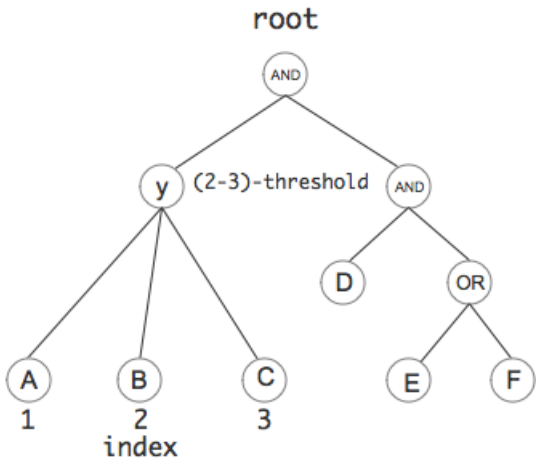

Fig. 2. Access Tree

C. Access Tree

In ABE schemes, a data owner needs to describe an access policy on encrypted data. An access tree is a kind of access structure that describing access policies. In the tree T, a leaf node x stands for an attribute and we let A_x denote its attribute. A non-leaf node y represents a $(t_y - n_y)$-threshold gate, where $1 \leq t_y \leq n_y$ is its threshold value and n_y is the number of its children. Especially, when $t_y = 1$ node y represents an $\mathsf{OR}(\vee)$ gate and when $t_y = n_y$ it represents an $\mathsf{AND}(\wedge)$ gate. We denote the parent node of y as $p(y)$. The children of y are indexed from 1 to n_y and for a specific child x of y, a function $\rho(x)$ returns the index of x.

Let T_R denote a tree rooted at node R. For a set of attributes S and a leaf node x, we say set S satisfies T_x if and only if $A_x \in S$ and it is written as $\mathsf{T}_x(S) = 1$. Then $\mathsf{T}_R(S)$ can be calculated in a recursive way. We utilize Fig. 2 as an example to illustrate the access tree. In Fig. 2, there are 6 attributes $\{A, B, C, D, E, F\}$ in the access tree. The non-leaf node y represents a $(2-3)$-threshold gate. Thus the access tree describe a policy 2-out-of-$(A, B, C) \wedge D \wedge (E \vee F)$. An attribute set $S_1 = \{A, B, D, F\}$ will satisfy this policy while a set $S_2 = \{A, B, C, D\}$ will not.

D. Access Control: Encoding and Decoding a Secret

As stated previously, the ciphertext of ABE is divided into a payload section and a control section that are denoted as CT_e and CT_c , respectively. The data owner encrypts a message m by some normal encryption algorithm to obtain

$$\mathsf{CT}_e = \mathsf{Enc}(m, s),$$

where s is the secret key. The control section can be modeled as the ciphertext of the access tree T encrypted with the secret key s, which is denoted as

$$\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T}).$$

In decryption, given a secret key SK, a user can decode the secret key as

$$s = \mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}).$$

Then from s, the user can decrypt the message as

$$m = \mathsf{Dec}(\mathsf{CT}_e, s).$$

Now, we illustrate the encoding and decoding process in detail.

1) Encoding a Secret: The basic idea of encoding a secret according to an access tree T is to share the secret to each leaf node of the tree. Then, the secret share distributed to a leaf node is disguised by the attribute of the leaf node. Since each non-leaf node is a $(t-n)$-threshold gate, the secret is shared from the root of the node using $(t-n)$-threshold secret sharing schemes.

To be specific, given an access tree T, each node y (including the leaf node) is associated with a randomly generated polynomial f_y of degree $d_y = t_y - 1$. These polynomials are generated in a recursive manner starting from the root node R. First, for root R, the constant term of f_R is set to be the secret key s, that is $f_R(0) = s$. Then another d_y points are chosen to fully define the polynomial f_R. For each child x, set $f_x(0) = f_{p(x)}(\rho(x))$ and another d_x points are randomly chosen to define polynomial f_x. In the end, for each leaf node l, we will obtain an associated secret share $f_l(0)$.

The next step is to disguise the secret share with the attribute for each leaf node $l \in L$, where L denotes the set of all leaf nodes. Especially, $\forall l \in L$, compute a pair of ciphertext $\mathsf{CT} = (C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)})$, where g is a public parameter and $h(\cdot)$ is a hash function that maps an arbitrary attribute to an element in $\mathbb{G}_1$. In summary, the secret encoding process can be modeled as a function

$$\begin{aligned} & \mathsf{Encode}(s, \mathsf{T}) \\ = \ & \mathsf{CT} \\ = \ & \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}. \end{aligned}$$

2) Decoding a Secret: The task of decoding is to reconstruct the secret key s encoded to the access tree T based on user's secret key SK. In ABE system, the secret key of each user is issued by the AA based on the user's attribute set. For a user with attribute set S, the secret key is

$$\mathsf{SK} = \{(D_{j1} = g^r \cdot h(j)^{r_j}, D_{j2} = g^{r_j}), \forall j \in S\},$$

where $r, r_j \in \mathbb{Z}_p$ are random numbers.

The decoding process is a recursive algorithm consisting of decoding each node in T. Intuitively, the secret reconstruction starts from the leaf nodes and goes towards the root. Let

$$\mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}, S, x) \rightarrow c_x$$

be a function to decode a node x. When x is a leaf node, if $A_x \in S$, we can compute

$$\begin{aligned} c_x &= \frac{e(D_{A_x 1}, C_{x1})}{e(D_{A_x 2}, C_{x2})} \\ &= \frac{e(g^r \cdot h(A_x)^{r_{A_x}}, g^{f_x(0)})}{e(g^{r_{A_x}}, h(A_x)^{f_x(0)})} \\ &= \frac{e(g,g)^{r f_x(0)} \cdot e(h(A_x), g)^{r_{A_x} f_x(0)}}{e(h(A_x), g)^{r_{A_x} f_x(0)}} \\ &= e(g,g)^{r f_x(0)}. \end{aligned}$$

If $A_x \notin S$, set $c_x = \perp$.

When x is a non-leaf node, denote its children as a set S_{cx}. For each child $z \in S_{cx}$, the decoding result is c_z. If the number of children whose decoding result $c_z \neq \perp$ is smaller than the threshold value t_x, it means that the threshold gate associated with x is not satisfied. Then we set $c_x = \perp$. Otherwise, we can form a valid set S_{vx} of size t_x where each children node $z \in S_{vx}$ has a valid decoding result c_z. Let $I_{vx} = \{\rho(z) | z \in S_{vx}\}$ be the index set for the corresponding nodes in S_{vx}. For index $i \in I_{vx}$, denote the Lagrange coefficient be

$$L_i(u) = \prod_{j \in I_{vx}, j \neq i} \frac{u - i}{j - i}.$$

Then we can compute the decoding of node x as

$$\begin{aligned} c_x &= \prod_{z \in S_{vx}} c_z^{L_i(0)} \\ &= \prod_{z \in S_{vx}} e(g,g)^{r f_z(0) L_i(0)}. \end{aligned}$$

Since in encoding, the secret associated with node z is set as $f_z(0) = f_{p(z)}(\rho(z)) = f_x(i)$, c_x can be further written as

$$\begin{aligned} c_x &= \prod_{z \in S_{vx}} e(g,g)^{r f_x(i) L_i(0)} \\ &= e(g,g)^{r \sum_{i \in I_{vx}} f_x(i) L_i(0)}. \end{aligned}$$

From the Lagrange interpolation, we have

$$f_x(0) = \sum_{i \in I_{vs}} f_x(i) L_i(0).$$

Thus, we have

$$c_x = e(g,g)^{r f_x(0)}.$$

By repeating this decoding process, we can obtain

$$c_R = e(g,g)^{rs}.$$

We note that the secret decoding will not give the secret key s in clear form. Instead, the decoding result is a disguised form $e(g,g)^{rs}$ that is further used in message decryption. In summary, the decoding process can be modeled as a function

$$\mathsf{Decode}(\mathsf{CT}_c, \mathsf{SK}, \mathsf{T}, S) = e(g,g)^{rs}.$$

IV. Construction of Outsourced CP-ABE

In this section, we introduce the detailed construction of the outsourced CP-ABE system. We first present some intuitive ideas.

A. Exploring Linearity of Secret Sharing

In ABE system, the message m is encrypted as $\mathsf{CT}_e = \mathsf{Enc}(m, s)$ with a secret key s. The secret key s is then encoded as $\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T})$. From the above analysis of encoding process, we can see that most of the computational burden lies in encoding. This motivates us to outsource the whole encoding portion to the cloud. However, if the cloud is in charge of encoding, it will inevitably learn the secret key s thus decryption the message. Moreover, the decoding process will produce the decoded secret as $e(g,g)^{rs}$. In an effective access control system, only the valid users can successfully decode the secret. Thus, it is crucial to prevent a compromised cloud and a malicious user to learn the decoding result $e(g,g)^{rs}$. These are the two critical issues in our design.

We address the above issues by exploring the linearity of secret sharing. Basically, the secret encoding process is a series of $(t-n)$-threshold secret sharing schemes according to the access tree T. In a secret sharing scheme, suppose the polynomial is

$$f(x) = s + c_1 x + c_2 x^2 + \cdots + c_d x^d.$$

Then a secret share is $(i, f(i))$. If we change all the secret shares to $(i, f(i)+\delta)$ with δ being a constant, then after the Lagrange interpolation, the recovered polynomial becomes

$$g(x) = (s+\delta) + c_1 x + c_2 x^2 + \cdots + c_d x^d.$$

That is, the shared secret changes from s to $s+\delta$. We define this linear relation between secret the secret shares as the linearity of secret sharing schemes.

The above analysis gives some insight of our proposed scheme. In ABE, the secret is encoded as

$$\mathsf{CT}_c = \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}.$$

Similarly, we can modify each pair of ciphertext (C_{l1}, C_{l2}) as $(C'_{l1} = g^{f_l(0)+\delta}, C'_{l2} = h(A_l)^{f_l(0)+\delta})$. Then the decoding process will reconstruct the secret as $e(g,g)^{r(s+\delta)}$. However, this modification requires data owners to calculate g^δ and $h(A_l)^\delta$ for each leaf node l in the access tree T. The computational overhead is still too expensive for mobile users. Instead, we propose to only modify the first part of the ciphertext pair. That is, set $(C'_{l1} = g^{f_l(0)+\delta}, C'_{l2} = C_{l2} = h(A_l)^{f_l(0)})$. We further re-design the decoding process such that the recovered secret would be $e(g,g)^{r(s+\delta)}$. The details will be provided in the construction of our outsourced ABE system.

In summary, the data owner will encryption the message as

$$\mathsf{CT}_e = \mathsf{Enc}(m, s+\delta).$$

The cloud will encode the secret key s (known to the cloud) according to an access tree T as

$$\mathsf{CT}_c = \mathsf{Encode}(s, \mathsf{T}).$$

On requesting, the cloud will decode CT_c to reconstruct the secret in the form of $e(g,g)^{r(s+\delta)}$. Based on the decoding result, the data user can decrypt the message m.

B. System Setup

The attribute authority executes $\mathsf{Setup}(\lambda) \rightarrow \{\mathsf{PK}, \mathsf{MSK}\}$.

1) AA generates a bilinear mapping $e : \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$, where $\mathbb{G}_1$ is a group of prime order p with generator g.
2) AA chooses two random exponents $\alpha, \beta \in \mathbb{Z}_p$ and calculates $h = g^\beta$ and $e(g,g)^\alpha$. The public key is published as
$$\mathsf{PK} = \{e, g, h = g^\beta, e(g,g)^\alpha\}.$$
3) AA sets the master key as
$$\mathsf{MSK} = \{\beta, \alpha\}.$$

C. Key Generation

When a user with attribute set S requests his secret key, the attribute authority executes $\mathsf{KeyGen}(\mathsf{PK}, S) \rightarrow \mathsf{SK}$ and delivers the secret key SK to that user.

1) AA chooses random integers $r, z \in \mathbb{Z}_p$ and computes
$$\begin{aligned}\mathsf{SK}_S \;=\; & \{D_1 = g^{(\alpha+r)/z}, \\ & (D_{j1} = g^r \cdot h(j)^{r_j}, D_{j2} = h(j)^{r_j/\beta}, \\ & D_{j3} = g^{r_j}), \forall j \in S\}.\end{aligned}$$
2) AA computes $\mathsf{SK}_L = \{\frac{z}{\beta}\}$.
3) AA sets $\mathsf{SK} = \{\mathsf{SK}_L, \mathsf{SK}_S\}$. We note that SK_S is for the cloud to partially decrypt the ciphertext and SK_L is for the user to fully recover the plaintext.

D. Encryption

The encryption process consists of two phases: local side encryption $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ and cloud side encryption $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$.

$\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$:

1) Data owner chooses random integers $s_1, s_2 \in \mathbb{Z}_p$ and computes
$$C_1 = m \cdot e(g,g)^{\alpha(s_1+s_2)}, C_2 = h^{s_1}, C_3 = g^{s_1}.$$
2) Data owner sets $\mathsf{aux} = \{s_2, C_2, C_3\}$ and $\mathsf{CT}_1 = C_1$.
3) Data owner specifies an access policy T and uploads $\{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$ to the cloud.

$\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$:

1) Given the secret s_2 and access tree T, the cloud follows the encoding process introduced in Section III-D1. The secret s_2 is encoded as
$$\begin{aligned} & \mathsf{Encode}(s_2, \mathsf{T}) \\ = \; & \mathsf{CT}_c \\ = \; & \{(C_{l1} = g^{f_l(0)}, C_{l2} = h(A_l)^{f_l(0)}), \forall l \in L\}.\end{aligned}$$
2) The cloud computes $C_4 = C_2 \cdot h^{s_2} = h^{s_1+s_2}$ and $\forall l \in L$,
$$(C'_{l1} = C_{l1} \cdot C_3 = g^{f_l(0)+s_1}, C_{l2} = h(A_l)^{f_l(0)}).$$

3) The cloud sets

$$\mathsf{CT}_2 \quad = \quad \{C_4, (C'_{l1}, C_{l2}), \forall l \in L\}.$$

4) The whole ciphertext is $\mathsf{CT} = \{\mathsf{T}, \mathsf{CT}_1, \mathsf{CT}_2\}$, which is stored in the cloud storage.

E. Decryption

The decryption process consists of two phases: cloud side decryption $\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$ and local side decryption $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$.

$\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$ *:*

1) When a data user wants to access encrypted data, he first sends part of his secret key SK_S to the cloud.
2) The cloud decodes the ciphertext as $\mathsf{Decode}(\mathsf{T}, \mathsf{CT}, \mathsf{SK}_S) \rightarrow m_1$. If the attribute set S associated with SK_S satisfies the access tree T, the decoding result is $m_1 = e(g,g)^{r(s_1+s_2)}$. Otherwise, $m_1 = \perp$ indicating that the data user does not have the right to access the encrypted data. The correctness of this decoding algorithm is given in Section IV-F.
3) The cloud computes

$$\begin{aligned} m_2 &= e(D_1, C_4) \\ &= e(g^{(\alpha+r)/z}, h^{s_1+s_2}) \\ &= e(g,g)^{(s_1+s_2)(\alpha+r)\beta/z}. \end{aligned}$$

4) The cloud sets $\mathsf{CT}_S = \{m_1, m_2\}$, which is then delivered to the data user.

$\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$*:*

1) The data user decrypts the message as

$$\begin{aligned} \frac{C_1 \cdot m_2}{m_1^{\mathsf{SK}_L}} &= \frac{m \cdot e(g,g)^{\alpha(s_1+s_2)} e(g,g)^{r(s_1+s_2)}}{(e(g,g)^{(s_1+s_2)(a+r)\beta/z})^{z/\beta}} \\ &= \frac{m \cdot e(g,g)^{\alpha(s_1+s_2)} e(g,g)^{r(s_1+s_2)}}{e(g,g)^{(s_1+s_2)(a+r)}} \\ &= m. \end{aligned}$$

F. Proof of Correctness

In this section, we prove that given an attribute set S that satisfies the access tree T, the decoding process will decode the ciphertext as $m_1 = e(g,g)^{r(s_1+s_2)}$.

The decoding $\mathsf{Decode}(\cdot)$ is a recursive algorithm by decoding each node in T. Let $\mathsf{Decode}(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S, x) \rightarrow m_x$ be a function to decode a node x in T.

When x is a leaf node, the attribute associated with x is A_x. If $A_x \in S$, we can compute

$$\begin{aligned} m_x &= \frac{e(D_{A_x 1}, C_{x1})}{e(D_{A_x 3}, C_{x2}) \cdot e(D_{A_x 2}, C_2)} \\ &= \frac{e(g^r \cdot h(A_x)^{r_{A_x}}, g^{f_x(0)+s_1})}{e(g^{r_{A_x}}, h(A_x)^{f_x(0)}) \cdot e(h(A_x)^{r_{A_x}/\beta}, h^{s_1})} \\ &= \frac{e(g,g)^{r(f_x(0)+s_1)} \cdot e(h(A_x), g)^{r_{A_x}(f_x(0)+s_1)}}{e(g, h(A_x))^{r_{A_x} f_x(0)} \cdot e(g, h(A_x))^{s_1 r_{A_x}}} \\ &= e(g,g)^{r(f_x(0)+s_1)}. \end{aligned}$$

If $A_x \notin S$, set $m_x = \perp$.

When x is a non-leaf node, denote its children as a set S_{cx}. For each child $z \in S_{cx}$, the decoding result is m_z. If the number of children whose decoding result $m_z \neq \perp$ is smaller than the threshold value t_x, it means the threshold gate associated with x is not satisfied. Then we set $m_x = \perp$. Otherwise, we can form a valid set S_{vx} of size t_x where each children node $z \in S_{vx}$ has a valid decoding result m_z. Let $I_{vx} = \{\rho(z) | z \in S_{vx}\}$ be the index set for the corresponding nodes in S_{vx}. For index $i \in I_{vx}$, denote the Lagrange coefficient as

$$L_i(z) = \prod_{j \in I_{vx}, j \neq i} \frac{u-i}{j-i}.$$

Then we can compute the decoding of node x as

$$\begin{aligned} m_x &= \prod_{z \in S_{vx}} m_z^{L_i(0)} \\ &= \prod_{z \in S_{vx}} (e(g,g)^{r(f_y(0)+s_1)})^{L_i(0)}. \end{aligned}$$

Since in encoding, the secret associated with node z is set as $f_z(0) = f_{p(z)}(\rho(z)) = f_x(i)$, m_x can be further written as

$$\begin{aligned} m_x &= \prod_{z \in S_{vx}} (e(g,g)^{r(f_x(i)+s_1)})^{L_i(0)} \\ &= e(g,g)^{r(\sum_{i \in I_{vx}} f_x(i) L_i(0) + s_1 \sum_{i \in I_{vx}} L_i(0))} \\ &= e(g,g)^{r(f_x(0)+s_1)}. \end{aligned}$$

The second equality comes from polynomial interpolation and the fact that the sum of Lagrange coefficients equals to 1. When the root of the tree is decoded, we can get

$$m_1 = m_R = e(g,g)^{r(f_R(0)+s_1)} = e(g,g)^{r(s_1+s_2)}.$$

Thus we have proved that

$$\mathsf{Decode}(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) = m_1 = e(g,g)^{r(s_1+s_2)}$$

if and only if the attribute set S satisfies T. Otherwise, $m_1 = \perp$.

V. Complexity and Security Analysis

A. Complexity Analysis

The encryption and decryption process in ABE system contains mainly two computationally expensive operations: modular exponentiation (Exp) and bilinear pairing (Pair). Thus we utilize the number of Exp and Pair as measurements to evaluate the performance of our scheme.

In ABE system [5], the complexity of encryption and decryption is determined by the access policy. Suppose there are n attributes involved in the access tree T. For each attribute, the computation of the control section CT_c takes 2 Exp. To compute the encryption section CT_e in the ciphertext, the user has to conduct 2 Exp. Thus the complexity of encryption can be measured as $(2 + 2n)$ Exp. To decode a leaf node, the user needs to conduct 2 Pair. The complexity of decoding the

tree T is also determined by the threshold gates in the non-leaf nodes. To decode a threshold gate with threshold value t, the user needs to first select t valid children to decode and calculate t Exp. Denote the sum of the threshold values of all the non-leaf nodes as K. The whole decryption process takes $(2+2K)$ Pair and K Exp.

In comparison, at local encryption $\mathsf{Enc}_L(m, \mathsf{PK}) \rightarrow \{\mathsf{CT}_1, \mathsf{T}, \mathsf{aux}\}$, the data user needs to conduct 3 Exp. For cloud side encryption $\mathsf{Enc}_S(\mathsf{PK}, \mathsf{T}, \mathsf{aux}) \rightarrow \mathsf{CT}_2$, it takes $(1+2n)$ Exp for the cloud. In decryption, $\mathsf{Dec}_S(\mathsf{T}, S, \mathsf{CT}, \mathsf{SK}_S) \rightarrow \mathsf{CT}_S$ takes the cloud $(1+3K)$ Pair and K Exp. The message decryption $\mathsf{Dec}_L(\mathsf{CT}_S, \mathsf{SK}_L) \rightarrow m$ takes the user 1 Exp. We note that in the above analysis, the do not count in the complexity to share the secret according the access tree in the original ABE scheme. In our scheme, the user only needs to specify the access policy and the cloud will conduct the secret sharing.

In terms of communication, in the original ABE, the data owner has to compute the encryption section CT_e and the control section CT_c first and upload them to the cloud together with an access policy T. We denote the size of an element in group $\mathbb{G}_1$ and $\mathbb{G}_2$ as Len_1 and Len_2, respectively. The access policy T is simply a string and denote its size as Len_T. Thus the size of the total ciphertext is $1\mathsf{Len}_2 + (1+2n)\mathsf{Len}_1 + \mathsf{Len}_\mathsf{T}$. When a data user wants to access the encrypted data, he has to download the ciphertext from the cloud storage. Thus, the total communication overhead for encryption and decryption is $2\mathsf{Len}_2 + (2+4n)\mathsf{Len}_1 + 2\mathsf{Len}_\mathsf{T}$.

In comparison, in our proposed outsourcing scheme, a data owner only needs to calculate and upload the encryption section CT_e and specify an access tree T. Thus the communication overhead for encryption is $2\mathsf{Len}_1 + \mathsf{Len}_2 + \mathsf{Len}_T$. A data owner needs to download the partially decrypted message CT_S with size $2\mathsf{Len}_2$. Thus the total communication overhead is $2\mathsf{Len}_1 + 3\mathsf{Len}_2 + \mathsf{Len}_T$.

The computational and communication complexity of the original ABE scheme and our outsourced ABE scheme is summarized in Table I. From the table, we can see that in general, our scheme introduce $\mathcal{O}(n)$ performance gain for mobile users in both encryption and decryption. In total, the user and cloud together will do more computation. However, the increase of computation on cloud side which should be insignificant for the cloud.

B. Security Analysis

In this section, we analyze the security of our proposed scheme mainly from three aspects. That is (i) confidentiality of the encrypted file; (ii) honest access control and (iii) collusion tolerance. In our analysis, we let a single cloud server provide the partial encryption and decryption service, which is quite a strong adversary model. As a result, the cloud can hold the partial encryption and decryption key at the same time. We also assume that the cloud can be compromised. For example, malicious users can collude with the cloud in order to gain illegal access to certain files.

1) Confidentiality: Confidentiality is the most basic security requirement for ABE systems. It requires that the cloud cannot learn the encrypted message. In our scheme, the message m is encrypted as $C_1 = m \cdot e(g,g)^{\alpha(s_1+s_2)}$, where s_2 is known to the cloud while s_1 is kept secret by the user. Thus the only way for the cloud to recover message m is to obtain the secret s_1. For each encryption, the user will generate a different secret s_1 independently. As a result, the cloud can only try to learn s_1 from the partial encryption key g^{s_1}. However, given g^{s_1}, finding the discrete logarithm s_1 is computationally infeasible. Thus, we conclude that our scheme is secure in protecting the confidentiality of the message.

2) Honest access control: Besides confidentiality, the users are also concerned whether the cloud can honestly complete access control. One key feature of our scheme is that all the computations conducted by the cloud is not "security-sensitive". That is the input and output of the algorithms implemented at the cloud side can be publicly known and will not degrade the security. This feature enables the possibility of public audition. A trusted third party can always repeat the computations conducted by the cloud. As a result, it decreases the chance for the cloud to successfully cheat in computation.

On the other hand, access control is marked as a service provided by the cloud. Since the access structure is embedded in ciphertext, a user thus can decide whether he is qualified to decrypt even before decryption. If a qualified user cannot decrypt the ciphertext, he can report an error thus degrading the quality of service.

3) Collusion tolerance: Honest access control enables users to access files as long as their attributes satisfy the policy. A secure access control scheme should also prevent users whose attributes do not satisfy the policy from accessing the files. In some situations, malicious users may collude with the cloud in order to gain illegal access. In this case, a malicious user will give his user secret key $SK_L = \frac{z}{\beta}$ to the cloud. We note that successfully decoding of access tree will produce $e(g,g)^{r(s_1+s_2)}$. Similar to the previous analysis, it is computationally infeasible for the cloud to derive (s_1+s_2) since s_2 is kept secret by the data owners. As a consequence, the only way to derive $e(g,g)^{r(s_1+s_2)}$ is through decoding of access tree with valid set of attributes.

VI. Numeric Results

In this section, we present some experiment results to demonstrate the efficiency of our proposed scheme. The experiment is conducted in an Android phone Samsung GT-I9100 with Android 4.1.2 operating system (we note that it is a relatively old android phone). The CPU is Dual-core 1.2 GHz Cortex-A9 with 1 GB RAM.

We implement our outsourced ABE based on Java Pairing-Based Cryptography Library (JPBC) [16]. We compare the average CPU time for both encryption and decryption of our

TABLE I
COMPLEXITY COMPARISON

	Encryption		Decryption		Communication Overhead
	User	Cloud	User	Cloud	
Original ABE [5]	$(2+2n)$ Exp	–	$(2+2K)$ Pair + K Exp	–	$2\mathsf{Len}_2 + (2+4n)\mathsf{Len}_1 + 2\mathsf{Len}_\mathsf{T}$
Outsourced ABE	3 Exp	$(1+2n)$ Exp	1 Exp	$(1+3K)$ Pair + K Exp	$2\mathsf{Len}_1 + 3\mathsf{Len}_2 + \mathsf{Len}_\mathsf{T}$

proposed scheme with that of the benchmark implemented in [16]. In the implementation, each attribute policy is a string where attribute names are connected by "t-of-n" gates. For example, a formal policy "A B 1-of-2 C D 2-of-3" represents at least 2 of $\{A \vee B, C, D\}$ should be satisfied. To measure the average performance of decryption, we utilize a simple access policy in the testing. We let all the leaf nodes be connected by a single root node and set the root as a $(\frac{n}{2}-n)$-threshold gate. We let the message m be a single element in $\mathbb{G}_1$. In practice, this m can be the symmetric encryption key for a group of files. Any data user who can recover the key m can access those files.

In our experiments, we utilize three different security levels as shown in Table II. And we let the number of attributes vary from 1 to 30.

TABLE II
SECURITY LEVELS OF ABE

Security level bits	$L_1(80)$	$L_2(112)$	$L_3(128)$
Bit length of r	160	224	256
Bit length of q	512	1024	1536

The experiment results are shown in Fig. 3 and Fig. 4. In the figures, "enc" and "dec" stands for encryption and decryption, respectively. "sos" indicates secure outsourcing. L_1, L_2, L_3 represents three different security levels. "gain" means the performance gain of outsourcing in terms of execution time. We can see that the encryption and decryption time for the original ABE are both approximately linear to the number of attributes. In comparison, the time for encryption and decryption at the local side almost remains constant regardless of the number of attributes. For instance, for the most secure level L_3, encrypting a message on mobile phone takes around 2 seconds and decryption takes around 0.2 seconds. In comparison, at level L_3, encryption in the original ABE will take more than 100 seconds for 15 attributes and decryption will take around 28 seconds. Generally, encryption is more time consuming than decryption under the access policy specified in the experiment.

From the theoretical analysis in Table I, we can see that at the local side, both the data owner and the data user need to do a constant number of Exp. While in the original ABE, the computational overhead of encryption and decryption are linear to the number of attributes n or the sum of the threshold K. Thus the experiment results are consistent with the theoretical analysis.

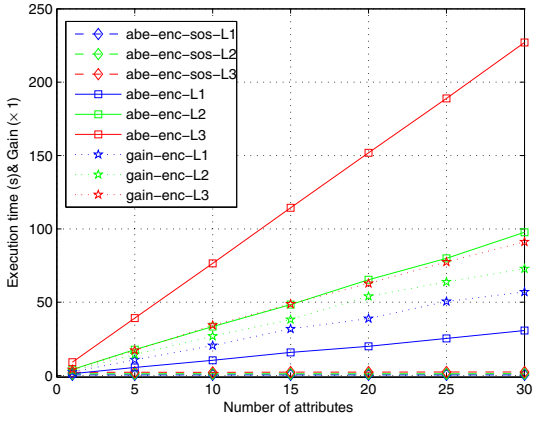

Fig. 3. Performance Comparison of Outsourced Encryption

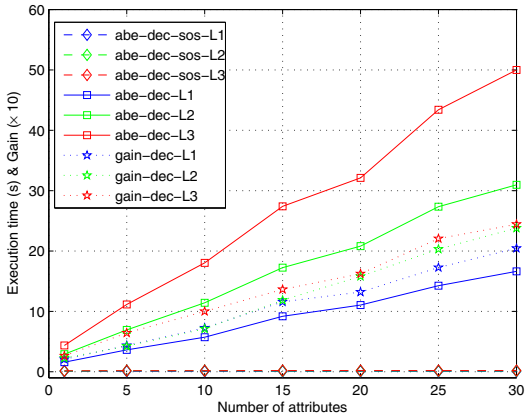

Fig. 4. Performance Comparison of Outsourced Decryption

VII. APPLICATION SCENARIOS

In this section, we introduce some application scenarios where our proposed scheme can be utilized.

From a theoretic point of view, we introduce a systematic way to securely outsource ABE. We modeled ABE as two highly detached functionalities: encryption and encoding. We further point out that encoding is a series of cascaded linear secret sharing schemes. Thus, we propose to utilize the linearity of secret sharing schemes to modify the secret shares in such a way that users can control the secret to

be shared. Currently, most ABE systems utilize linear secret sharing schemes to construct the expressive access structure. For instance, the access tree structure as discussed in this paper and the monotone span programs [17] utilized in [6]. We believe that our proposed scheme can be applied in such kind of ABE systems utilizing linear system sharing schemes.

In practice, out scheme is suitable for the scenarios where mobile users need to store and share their private data in a fine-grained manner. More and more applications rely on mobile devices to collet sensitive data and upload the data to a sharing environment such as cloud. Then qualified users can utilized the data under the control of the data owner. For instance, in mobile healthcare system, wearable devices can sense and collect user's health related data that may be private. In this case, data owner can encrypt the data using efficient symmetric encryption schemes. A symmetric key may be related to a group of data files. The secret keys then can be encrypted using our proposed scheme. The whole encryption files are then uploaded to the cloud. The data owner can specify policies on the encrypted files controlling who can access the data. For example, he can specify that only his family or physicians in certain departments of certain hospitals can access the data. Compare to the traditional public key cryptosystems, the data owner can encrypt the data to a class of users he may not know in advance instead of a certain person he already knows. Moreover, data users do not have to request keys from the data owners. This enables the data owners to manage data off-line.

VIII. Conclusion

In this paper, we introduce a systematic way to outsource attribute based encryption (ABE) systems by utilizing the linearity of secret sharing schemes. Our scheme enables mobile users to encrypt their data using normal encryption schemes and specify access policies on encrypted data to control who can access the data. The whole access control part is outsourced to the cloud. In addition, the cloud can partially decrypt the data if requested by the user. Thus the user can decrypt the data efficiently. The proposed scheme can guarantee security of the outsourced data. Moreover, the untrusted cloud is enforced to honestly complete the task of access control even in situation that the cloud may collude with malicious users. The experiment results show that our scheme is highly efficient. Mobile users only need to conduct constant time computation regardless of the complexity of access policies. We also provide some application scenarios where our scheme can be utilized.

In summary, our proposed outsourcing scheme makes attribute based encryption transparent to mobile users. The nice property of ABE to provide fine-grained access control is preserved. Meanwhile, mobile users can enjoy dramatically performance gain without sacrificing security.

References

[1] 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," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 1, pp. 131–143, 2013.

[2] D. Boneh and M. Franklin, "Identity-based encryption from the weil pairing," in *Advances in Cryptology–CRYPTO 2001*, pp. 213–229, Springer, 2001.

[3] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Advances in Cryptology–EUROCRYPT 2005*, pp. 457–473, Springer, 2005.

[4] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proceedings of the 13th ACM conference on Computer and communications security*, pp. 89–98, Acm, 2006.

[5] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Security and Privacy, 2007. SP'07. IEEE Symposium on*, pp. 321–334, IEEE, 2007.

[6] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *Public Key Cryptography–PKC 2011*, pp. 53–70, Springer, 2011.

[7] M. Ambrosin, M. Conti, and T. Dargahi, "On the feasibility of attribute-based encryption on smartphone devices," in *Proceedings of the 2015 Workshop on IoT challenges in Mobile and Industrial Systems*, pp. 49–54, ACM, 2015.

[8] X. Wang, J. Zhang, E. M. Schooler, and M. Ion, "Performance evaluation of attribute-based encryption: Toward data privacy in the iot," in *Communications (ICC), 2014 IEEE International Conference on*, pp. 725–730, IEEE, 2014.

[9] M. Brown, D. Hankerson, J. López, and A. Menezes, *Software implementation of the NIST elliptic curves over prime fields*. Springer, 2001.

[10] S. Hohenberger and B. Waters, "Online/offline attribute-based encryption," in *Public-Key Cryptography–PKC 2014*, pp. 293–310, Springer, 2014.

[11] M. Green, S. Hohenberger, and B. Waters, "Outsourcing the decryption of abe ciphertexts.," in *USENIX Security Symposium*, vol. 2011, 2011.

[12] 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 *Computer Security–ESORICS 2013*, pp. 592–609, Springer, 2013.

[13] J. Li, C. Jia, J. Li, and X. Chen, "Outsourcing encryption of attribute-based encryption with mapreduce," in *Information and Communications Security*, pp. 191–201, Springer, 2012.

[14] A. Beimel, "Secret-sharing schemes: a survey," in *Coding and cryptology*, pp. 11–46, Springer, 2011.

[15] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[16] A. De Caro and V. Iovino, "jpbc: Java pairing based cryptography," in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011*, (Kerkyra, Corfu, Greece, June 28 - July 1), pp. 850–855, IEEE, 2011.

[17] E. F. Brickell, "Some ideal secret sharing schemes," in *Advances in Cryptology EUROCRYPT 89*, pp. 468–475, Springer, 1989.