Efficient and Secure Data Storage Operations for Mobile Cloud Computing

Zhibin Zhou and Dijiang Huang
{zhibin.zhou,dijiang}@asu.edu
Arizona State University

***Abstract*—Cloud computing is a promising technology, which is transforming the traditional Internet computing paradigm and IT industry. With the development of wireless access technologies, cloud computing is expected to expand to mobile environments, where mobile devices and sensors are used as the information collection nodes for the cloud. However, users' concerns about data security are the main obstacles that impede cloud computing from being widely adopted. These concerns are originated from the fact that sensitive data resides in public clouds, which are operated by commercial service providers that are not trusted by the data owner. Thus, new secure service architectures are needed to address the security concerns of users for using cloud computing techniques.**

In this paper, we present a holistic security framework to secure the data storage in public clouds with the special focus on lightweight wireless devices store and retrieve data without exposing the data content to the cloud service providers. To achieve this goal, our solution focuses on the following two research directions: First, we present a novel Privacy Preserving Cipher Policy Attribute-Based Encryption (PP-CP-ABE) to protect users' data. Using PP-CP-ABE, light-weight devices can securely outsource heavy encryption and decryption operations to cloud service providers, without revealing the data content and used security keys. Second, we propose an Attribute Based Data Storage (ABDS) system as a cryptographic access control mechanism. ABDS achieves information theoretical optimality in terms of minimizing computation, storage and communication overheads. Especially, ABDS minimizes cloud service charges by reducing communication overhead for data managements. Our performance assessments demonstrate the security strength and efficiency of the presented solution in terms of computation, communication, and storage.

I. Introduction

Existing cloud provides two main services: storage and computation. Users' concerns about data security are the main obstacles that prevent the public cloud from widely adopted. These concerns origin from the fact that sensitive data are stored and processed in public clouds, which are operated by commercial service providers and shared by various other customers. Along with the other customers who can be potential competitors or malicious attackers, these service providers, esp., the storage and computing service providers, are usually not trusted by the data owner. Moreover, the multi-tenant data architecture directly results in the risk that a user's data being exposed to business competitors or malicious attackers, who may compromise the data server shared among tenants.

Data confidentiality is a desired property when users outsource their data storage to public cloud service providers. To protect users' data, encryption is used to secure the data in the cloud. Recently, Ciphtertext Policy Attribute-Based Encryption (CP-ABE) schemes [3], [10], [7], [20] were proposed to facilitate key management and cryptographic access control in an expressive and efficient way. Under the construction of CP-ABE, an attribute is a descriptive string assigned to (or associated with) a user and each user may be tagged with multiple attributes. Multiple users may share common attributes, which allow message encryptors to specify a data access policy by composing multiple attributes through logical operators such as "AND", "OR", etc. To decrypt the message, the decryptor's attributes need to satisfy the access policy. These unique features of CP-ABE solutions make them appealing in the cloud data storage system that requires an efficient data access control for a large number of users belonging to different organizations.

With the fast development of wireless technology, mobile cloud has become an emerging cloud service model [18], in which mobile devices and sensors are used as the information collecting and processing nodes for the cloud infrastructure. This new trend demands researchers and practitioners to construct a trustworthy architecture for mobile cloud computing, which includes a large numbers of lightweight, resource-constrained mobile devices.

With the CP-ABE enabled cloud storage service, a new challenge is *how to incorporate wireless mobile devices, especially lightweight devices such as cell phones and sensors, into the cloud system.* This new challenge is originated from the fact that CP-ABE schemes always require intensive computing resources to run the encryption and decryption algorithms. To address this issue, an effective solution is to outsource the heavy encryption and decryption computation without exposing the sensitive data contents or keys to the cloud service providers. Our research described in this paper proposes such a solution for CP-ABE.

Another research challenge is *how to share encrypted data with a large number of users, in which the data sharing group can be changed frequently.* For example, when a user is revoked from accessing a file, he/she is not authorized to access any future updates of the file, i.e., the local copy (if exists) will get outdated. To this end, the updated data need to be encrypted by a new encryption key.

Furthermore, the third research challenge is *how to upload/download and update encrypted data stored in the cloud system.* For example, when changing certain data fields of an encrypted database, the encrypted data needs to be downloaded from cloud and then be decrypted. Upon finishing the updates, the files need to be re-encrypted and uploaded to the cloud system. Frequent upload/download operations will cause tremendous overhead for resource constrained wireless devices. Thus, it is desirable to design a secure and efficient cloud data management scheme to balance the communication and storage operational overhead incurred by managing the encrypted data.

To address the above described research challenges, in this paper, we present a holistic secure mobile cloud data management framework that includes two major components:

1) A Privacy Preserving CP-ABE (PP-CP-ABE) scheme;
2) An Attribute-Based Data Storage (ABDS) scheme that achieves information theoretical optimality.

Using PP-CP-ABE, users can securely outsource computation intensive CP-ABE encryption and decryption operations to the cloud without revealing data content and secret keys. In this way, lightweight and resource constrained devices can access and manage data stored in the cloud data store.

The ABDS system achieves scalable and fine-grained data access control, using public cloud services. Based on ABDS, users' attributes are organized in a carefully constructed hierarchy so that the cost of membership revocation can be minimized. Moreover, ABDS is suitable for mobile computing to balance communication and storage overhead, and thus reduces the cost of data management operations (such as upload, updates, etc.) for both the mobile cloud nodes and storage service providers.

The rest of this paper is organized as follows. Section II presents system models used in this paper. We present detailed PP-CP-ABE construction and ABDS design in Section III and IV, respectively. In Section V, we analyzed the security and discuss the performance of proposed schemes with comparison to several related works. We describe related works in Section VI. Finally, we conclude our work in Section VII.

II. System and Models

A. Notations

the notations used in this paper is listed in Table II-A:

TABLE I
NOTATIONS USED IN THIS PAPER.

Acronym	Descriptions
DO	Data Owner
ESP	Encryption Service Provider
DSP	Decryption Service Provider
SSP	Storage Service Provider
TA	Trust Authority
T	Access Policy Tree

B. System Model

In our proposed system, we denote the Data Owner as DO. A DO can be a mobile wireless device or a sensor that can request and/or store information from/in the Cloud. The data are secured by using our proposed PP-CP-ABE scheme. Other than DO, there are many data receivers who subscribe to the data owned by DO. The presented system model has the following properties:

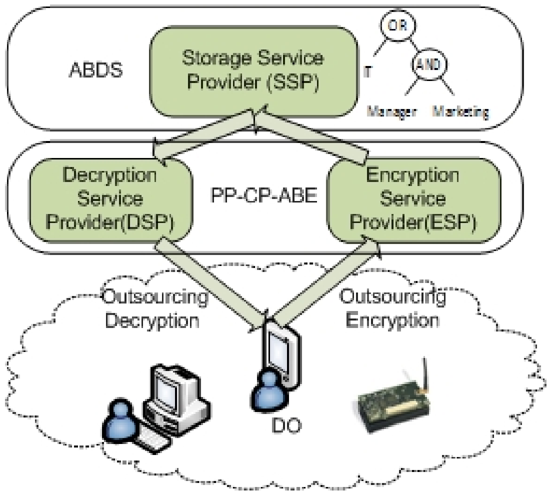

ABDS
Storage Service Provider (SSP)
OR
T
AND
Manager
Marketing
Decryption Service Provider(DSP)
PP-CP-ABE
Encryption Service Provider(ESP)
Outsourcing Decryption
Outsourcing Encryption
DO

Fig. 1. System Architecture of Our Proposed Framework.

1) The data must be encrypted before sending to storage service provider (SSP);
2) The encryption service provider (ESP) provides encryption service to the data owner without knowing the actual data encryption key (DEK);
3) The decryption service provider (DSP) provides decryption service to users without knowing the data content;
4) Even ESP, DSP and SSP collude, the data content cannot be revealed;

As shown in Figure 1, the SSP, ESP, and DSP form the core components of the proposed system. ESP and DSP provide PP-CP-ABE services and SSP, e.g., Amazon S3, provides storage services. The cloud is semi-trusted, in which the cloud only provides computing and storage services with the assistance on data security; however, the data is blinded to the cloud. In particular, more powerful PCs and Mobile Phones can works as communication proxy for sensors that collect information.

C. Attacking Models

We assume that the symmetric encryption algorithm and one-way hash function used in this paper is secure and the Discrete Logarithm Problem (DL) on both groups $\mathbb{G}_0$ and $\mathbb{G}_1$ is hard. In addition, the Trust Authority (TA) is responsible for distributing cryptographic keys, and it is well guarded and trustable. We consider the cloud service providers are honest but curious [12]. In other words, the service providers will perform in accordance to our proposed protocols and returns correct computation results. Misbehavior can be easily detected and punished by the customers. However, service providers will try to find out as much sensitive information (e.g., personal data, keys, etc.) as possible and may collude with malicious attackers.

The malicious attackers' goal is to reveal data in the cloud without authorization from DOs. Multiple attackers can combine their secrets to perform collusion attacks, in which they can try to decrypt the ciphertext and compromise the decryption keys that they are not authorized to access. One

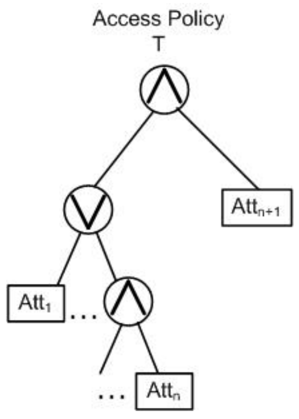


Fig. 2. Illustration of a sample access policy tree.

particular example of this collusion attack is that multiple users are revoked from accessing a file, and they try to collude to get updated files from the public cloud.

In particular, attackers want to break the *Forward Secrecy*, which is defined as follows: After a user is revoked from accessing a file, he/she may have a local copy of the file; however, the revoked user must not get any future updates on this file.

While data integrity and retrievability in the cloud are also important security requirements, they are not the focuses of this paper. Readers can refer to research works in the provable data possession (PDP) [1], [13].

D. Access Policy Tree

In this section, we briefly describe the model of an access policy tree used in PP-CP-ABE. The data access policy tree of PP-CP-ABE is composed by leaf nodes and internal nodes. Each leaf node represents an attribute, and each internal node is a logical gate, such as "AND", "OR", "n-of-m". Several functions and terms are defined as follows to facilitate the presentation of our solutions:

- $\mathbf{parent}(x)$: return the parent node of node x;
- $\mathbf{att}(x)$ denotes the attribute associated with the leaf node x in the data access tree;
- The access tree $\mathcal{T}$ composed by a set of leaf nodes (i.e., attributes) and internal nodes (i.e., logical gates) defines the data access policies, i.e., if a user *owns* a set of attributes that satisfy the logic operations of the tree to reach the root, it can access the secret secured by $\mathcal{T}$. Here *owns* means that the user has the private keys corresponding to the set of attributes. AND and OR are the most frequently used logical gates.
- num_x is the number of children of a node x. A child y of node x is uniquely identified by an index integer $\mathbf{index}(y)$ from 1 to num_x.
- The threshold value $k_x = num_x - 1$ when x is an *AND*, and $k_x = 0$ when x is an OR gate or a leaf node. k_x is used as the polynomial degree for node x using the threshold secret sharing scheme [26].

III. Privacy Preserving CP-ABE

In this section, we present the construction of PP-CP-ABE algorithm. In PP-CP-ABE, DOs outsource intensive computation of CP-ABE encryption and decryption to powerful cloud service providers (ESP and DSP) without disclosing their data content and secret keys.

A. Overview of the Construction

Essentially, the basic idea of PP-CP-ABE to outsource intensive but non-critical part of the encryption and decryption algorithm to the service providers while retain critical secrets. As we can prove later in this paper, the outsourcing of computation does not reduce the security level compared with original CP-ABE schemes, where all computations are performed locally.

The encryption complexity of CP-ABE grows linearly on the size of access policy. During the encryption, a master secret is embedded into ciphertext according to the access policy tree in a recursive procedure, where, at each level of the access policy, the secret is split to all the subtree of the current root. However, the security level is independent on the access policy tree. In other words, even if the ESP possesses secrets of most but not all parts of the access policy tree, the master secret is still information theoretically secure given there at least one secret that is unknown to ESP. Thus, we can safely outsource most part of encryption complexity to ESP by just retaining a small amount of secret information, which is processed locally.

As for the decryption, the CP-ABE decryption algorithm is computationally expensive since bilinear pairing operations over ciphertext and private key is a computational intensive operation. PP-CP-ABE addresses this computation issue by securely blinding the private key and outsourcing the expensive Pairing operations to the DSP. Again, the outsourcing will not expose the data content of the ciphertext to the DSP. This is because the final step of decryption is performed by the decryptors.

B. Background Information

Here, we present some preliminary knowledge about our construction. Our proposed PP-CP-ABE is constructed using bilinear pairing as well as secret sharing schemes, which are briefly presented as below.

1) Bilinear Pairing: Pairing is a bilinear map function $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_1$, where $\mathbb{G}_0$ and $\mathbb{G}_1$ are two multiplicative cyclic groups with large prime order p. The Discrete Logarithm Problem on both $\mathbb{G}_0$ and $\mathbb{G}_1$ are hard. Pairing has the following properties:

- *Bilinearity*:
$$e(P^a, Q^b) = e(P, Q)^{ab}, \quad \forall P, Q \in \mathbb{G}_0, \forall a, b \in \mathbb{Z}_p^*.$$
- *Nondegeneracy*:
$e(g, g) \neq 1$ where g is the generator of $\mathbb{G}_0$.
- *Computability*:
There exists an efficient algorithm to compute the pairing.

2) Secret Sharing: (t,n) secret sharing is used to divide a secret into n shares and any t shares can reconstruct the secret, while combining less than t shares will not disclose any information about the secret. As introduced by Shamir at el. in [26], in a $t-1$ degree polynomial, any t points on the polynomial be used to reconstruct the secret, i.e., the polynomial. We define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$:

$$\Delta_{i,S}(x) = \prod_{(j\in S, j\neq i)} \frac{x-j}{i-j}.$$

C. System Setup and Key Generation

The TA first setups the PP-CP-ABE system by choose a bilinear map: $e: \mathbb{G}_0 \times \mathbb{G}_0 \rightarrow \mathbb{G}_1$ of prime order δ_P with the generator g. Then, TA chooses two random α, $\beta \in \mathbb{Z}_p$. The public parameters are published as:

$$PK = \langle \mathbb{G}_0, g, h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha} \rangle. \quad (1)$$

The master key is $MK = (\beta, g^{\alpha})$, which is only known by the TA.

Each user needs to register with the TA, who authenticates the user's attributes and generates proper private keys for the user. An attribute can be any descriptive string that defines, classifies, or annotates the user, to which it is assigned. The key generation algorithm takes as input a set of attributes S assigned to the user, and outputs a set of private key components corresponds to each of attributes in S. The key generation algorithm takes the following operations:

1) Chooses a random $r \in \mathbb{Z}_p$,
2) Chooses a random $r_j \in \mathbb{Z}_p$ for each attribute $j \in S$.
3) Computes the private key as:

$$\begin{aligned} SK = &\langle D = g^{(\alpha+r)/\beta}; \\ &\forall j \in S: D_j = g^r \times H(j)^{r_j}; D'_j = g^{r_j} \rangle. \end{aligned}$$

4) Sends SK to the DO through a secure channel.

D. PP-CP-ABE Encryption

To outsource the computation of Encryption and preserve the data privacy, a DO needs to specify a policy tree $\mathcal{T} = \mathcal{T}_{ESP} \bigwedge \mathcal{T}_{DO}$, where $\bigwedge$ is an *AND* logic operator connecting two subtrees $\mathcal{T}_{ESP}$ and $\mathcal{T}_{DO}$. $\mathcal{T}_{ESP}$ is the data access policy that will be performed by the ESP and $\mathcal{T}_{DO}$ is a DO controlled data access policy. $\mathcal{T}_{DO}$ usually has a small number of attributes to reduce the computation overhead at the DO, in which it can be a sub-tree with just one attribute (see the example shown in Figure 3).

In practice, if $\mathcal{T}_{DO}$ has one attribute, DO can randomly specify an 1-degree polynomial $q_R(x)$ and sets $s = q_R(0)$, $s_1 = q_R(1)$, and $s_2 = q_R(2)$. Then DO sends $\{s_1, \mathcal{T}_{ESP}\}$ to ESP, which is noted as:

$$DO \xrightarrow{\{s_1, \mathcal{T}_{ESP}\}} ESP.$$

Here, we must note that sending s_1 and $\mathcal{T}_{ESP}$ will not expose any secret of our solution. We will prove this in Section V-A.

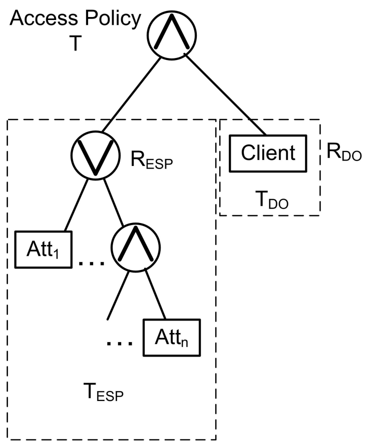


Fig. 3. Illustration of access policy $\mathcal{T} = \mathcal{T}_{ESP} \bigwedge \mathcal{T}_{DO}$.

ESP then runs the **Encrypt**$(s_1, \mathcal{T}_{ESP})$ algorithm, which is described below:

1) $\forall x \in \mathcal{T}_{ESP}$, randomly chooses a polynomial q_x with degree $d_x = k_x - 1$, where k_x is the secret sharing threshold value:
   a) For the root node of $\mathcal{T}_{ESP}$, i.e., R_{ESP}, Chooses a $d_{R_{ESP}}$-degree polynomial with $q_{R_{ESP}}(0) = s_1$.
   b) $\forall x \in \mathcal{T}_{ESP} \backslash R_{ESP}$ sets d_x-degree polynomial with $q_x(0) = q_{\mathbf{parent}(x)}(\mathbf{index}(x))$.
2) Generates a temporal ciphertext:

$$CT_{ESP} = \{\forall y \in Y_{ESP} : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\},$$

where Y_{ESP} is the set of leaf nodes in $\mathcal{T}_{ESP}$.

At the meantime, the DO performs the following operations:

1) Performs **Encrypt**$(s_2, \mathcal{T}_{DO})$ and derives:

$$CT_{DO} = \{\forall y \in Y_2 : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\}.$$

2) Computes $\widetilde{C} = Me(g,g)^{\alpha s}$ and $C = h^s$, where M is the message.
3) Sends $CT_{DO}, \widetilde{C}, C$ to the ESP:

$$DO \xrightarrow{\{CT_{DO}, \widetilde{C}, C\}} ESP.$$

On receiving the message from the DO, ESP generates the following ciphertext:

$$\begin{aligned} &CT = \langle \mathcal{T} = \mathcal{T}_{ESP} \bigwedge \mathcal{T}_{DO}; \widetilde{C} = Me(g,g)^{\alpha s}; C = h^s; \\ &\forall y \in Y_{ESP} \bigcup Y_{DO} : C_y = g^{q_y(0)}; C'_y = H(\mathbf{att}(y))^{q_y(0)} \rangle. \end{aligned}$$

Finally, the ESP sends CT to the SSP.

E. Outsourcing Decryption

CP-ABE decryption algorithm is computationally expensive since bilinear pairing is an expensive operation. PP-CP-ABE addresses this computation issue by outsourcing the expensive Pairing operations to the DSP. Again, the outsourcing will not expose the data content of the ciphertext to the DSP.

To protect the data content, the DO first blinds its private key by choosing a random $t \in \mathbb{Z}_p$ and then calculates $\widetilde{D} = D^t = g^{t(\alpha+r)/\beta}$. We denote the blinded private key as $\widetilde{SK}$:

$$\begin{aligned} \widetilde{SK} &= \langle \widetilde{D} = g^{t(\alpha+r)/\beta}, \\ &\quad \forall j \in S : D_j = g^r \cdot H(j)^{r_j}, D'_j = g^{r_j} \rangle. \end{aligned} \quad (2)$$

Before invoking the DSP, the DO first checks whether its owned attributes will satisfy the access policy $\mathcal{T}$. If so, the DO sends $\{\widetilde{SK}\}$ to the DSP, and requests the SSP to send the ciphertext to the DSP. On receiving the request, the SSP sends $CT' = \{\mathcal{T}; C = h^s; \forall y \in Y_1 \bigcup Y_2 : C_y = g^{q_y(0)}; C'_y = H(\mathbf{att}(y))^{q_y(0)}\}$ and $CT' \subset CT$ to the DSP:

$$SSP \xrightarrow{\{CT'\}} DSP. \quad (3)$$

Once the DSP receives both $\{\widetilde{SK}\}$ and CT', it then runs the **Decrypt**$(\widetilde{SK}, CT')$ algorithm as follows:

1) $\forall y \in Y = Y_{ESP} \bigcup Y_{DO}$ the DSP runs a recursive function $\mathrm{DecryptNode}(\mathrm{CT}', \widetilde{\mathrm{SK}}, \mathrm{R})$, where R is the root of $\mathcal{T}$. The recursion function is the same as defined in [3] and $\mathrm{DecryptNode}(CT', \widetilde{SK}, y)$ is proceeded as follows:

$$\begin{aligned} \mathrm{DecryptNode}(CT', \widetilde{SK}, y) &= \frac{e(D_i, C_y)}{e(D'_i, C'_y)} \\ &= \frac{e(g^r \cdot H(i)^{r_i}, g^{q_y(0)})}{e(g^{r_i}, H(i)^{q_y(0)})} \\ &= e(g,g)^{rq_y(0)} \\ &= F_y. \end{aligned}$$

The recursion is processed as follows: $\forall y$ is the child of x, it calls $DecryptNode(CT'; \widetilde{SK}; y)$ and stores the output as F_y. Let S_x be an arbitrary k_x-sized set of child nodes y, the DSP computes:

$$\begin{aligned} F_x &= \prod_{y \in S_x} F_y^{\Delta_{i,S'_x}(0)} \\ &= \prod_{y \in S_x} (e(g;g)^{r \cdot q_y(0)})^{\Delta_{i;S'_x}(0)} \\ &= \prod_{y \in S_x} (e(g;g)^{r \cdot q_{\mathbf{parent}(y)(\mathbf{index}(y))}})^{\Delta_{i;S'_x}(0)} \\ &= \prod_{y \in S_x} (e(g;g)^{r \cdot qx(i) \cdot \Delta_{i;S'_x}(0)} \\ &= e(g,g)^{rq_x(0)}, \end{aligned} \quad (4)$$

where $i = \mathbf{index}(z)$ and $S'_x = \{\mathbf{index}(z) : z \in S_x\}$. Finally, the recursive algorithm returns $A = e(g,g)^{rs}$.

2) Then, computes

$$e(C, \widetilde{D}) = e(h^s, g^{t(\alpha+r)/\beta}) = e(g,g)^{trs} \cdot e(g,g)^{t\alpha s}.$$

3) Sends $\{A = e(g,g)^{rs}, B = e(C, \widetilde{D}) = e(g,g)^{trs} \cdot e(g,g)^{t\alpha s}\}$ to the DO:

$$DSP \xrightarrow{\{A,B\}} DO.$$

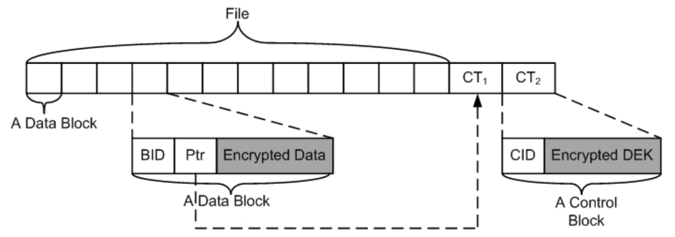


Fig. 4. Illustration a file organized into blocks with multiple control blocks.

On receiving $\{A, B\}$, DO calculates $B' = B^{1/t} = e(g,g)^{rs} \cdot e(g,g)^{\alpha s}$ and then it recovers the message:

$$M = \frac{\widetilde{C}}{(B'/A)} = \frac{Me(g,g)^{\alpha s}}{(e(g,g)^{rs} \cdot e(g,g)^{\alpha s})/e(g,g)^{rs}}.$$

IV. Attribute Based Data Storage

In this section, we present an Attribute Based Data Storage (ABDS) scheme that is based on PP-CP-ABE to enable efficient, scalable data management and sharing.

A. Data Management Overview

The frequent data updates will cause additional expense for file managements. For example, to update existing files, e.g., changing certain data fields of an encrypted database, in which the encrypted data need to be downloaded from SSP to DSP for decryption. Upon finishing the updates, the ESP needs to be re-encrypted and upload the data to the SSP. Thus, the re-encrypted process requires downloading and uploading the data, which may incur high communication and computation overhead, and as a result, will cost more for DOs.

To address the described cost issue, it is reasonable to divide a file into independent blocks that are encrypted independently. To update files, the DO can simply download the particular blocks to be updated. In this way, we can avoid re-encrypting the entire data. Moreover, data access control can be enforced on individual blocks using “lazy” re-encryption strategy. For example, when the data access memberships to a particular file are changed (i.e., the access tree is changed), this event can be recorded but no file changes are invoked. Until the data content needs to be updated, the re-encryption is then performed using the proposed PP-CP-ABE scheme.

Partitioning the data into multiple small blocks also introduces addition overhead. This is because the extra control information needs to be attached for each data block for data management. For example, the control message should include a block ID and a pointer to its corresponding data access tree $\mathcal{T}$. In Figure 4, we depicted a sample file stored in SSP. As shown in Figure 4, each file is divided into blocks. A block is a tuple {BID, Ptr, Encrypted Data}, where BID is the unique identification of the block; Ptr is the pointer to the control block CT; and data is encrypted with a DEK. A control block {CID, Encrypted DEK} has a control block ID, i.e., CID and DEK encrypted by using PP-CP-ABE scheme.

The ABDS system should take into the considerations the issues on what is the appropriate data block size to be partitioned with a known file size. In this work, our goal is to minimize the storage and communication overhead with the considerations the following simple assumptions:

1) Every data update should only affect a small amount of data, e.g., updating certain data fields in the Database;
2) In each unit time period, the number of blocks to be updated is known;
3) Each data block has the same probability to be updated;

One exemplary application scenario that complies with those assumptions is the traffic information collection, where lightweight devices, such as cell phone and sensors, serves as mobile or static data collection agents. Periodically, the devices update the corresponding data fields in encrypted databases.

Based on the above discussions, we can model the total cost C in a unit time period as follows:

$$C = 2nS_bC_c + \frac{F}{S_b}S_cC_s, \tag{5}$$

where n is the number of updated blocks in a unit time period and $2n$ stands for an update includes one encryption and one decryption that require two transmissions; S_b is the size of block; C_c is the cost rate of data transmission that is charged by both cloud storage providers and wireless communication service providers; F is the size of file; S_c is the size of control data for each data block, and C_s is the charging rate of storage. To minimize cost C, DO can minimize (5) and derive the optimal block size:

$$S_b \geq 2\sqrt{2nC_cFS_cC_s}.$$

B. Setup

PP-CP-ABE enables expressive policy with descriptive attributes to enforce data access control to the stored data. For example, if Alice wants to share a file to all CS students, she can specify the descriptive policy "CS *AND* Student". All the users whose attributes satisfy this policy can decrypt the data.

Besides the set of descriptive attributes enabled in the system, each user is assigned a unique binary ID: $b_0b_1 \ldots b_{n-2}b_{n-1}$. We can define the term "*bit-assignment* attribute" that is represented as "B_i" or "$\overline{B_i}$" to indicate the binary value at position i in the ID. B_i indicates the i'th bit of an ID is 1; $\overline{B}_i$ indicates the i'th bit of an ID is 0. If the length of an ID is n, then the total number of bit-assignment attributes is $2n$. This means that two binary values are mapped to one bit position (one for value 0 and one for value 1). Thus, a DO with ID u is uniquely identified by the set of bit-assignments S_u. Also, multiple DOs may have a common subset of bit-assignments. For example, a DO u_1's ID is 000 and a DO u_2's ID is 001, $S_{u_1} = \{\overline{B}_0, \overline{B}_1, \overline{B}_2\}$ and $S_{u_2} = \{\overline{B}_0, \overline{B}_1, B_2\}$ and $S_{u_1} \bigcap S_{u_2} = \{\overline{B}_0, \overline{B}_1\}$. Bit-assignment attributes can be used when the DO wants to share data to any arbitrary set of DOs. In this case, it may be hard to describe the set of DOs efficiently using descriptive attributes.

C. Upload New Files

Before uploading new files to the SSP, the ESP and DO need to determine the encryption parameters such as the block size. DO then invokes ESP with an access policy $\mathcal{T}_{ESP}$, which is the access policy to be enforced on the uploaded files.

We first define some terms used in the following presentations:

- *Literal*: A variable or its complement, e.g., b_1, $\overline{b_1}$, etc.
- *Product Term*: Literals connected by AND, e.g., $\overline{b}_2b_1\overline{b}_0$.
- *Sum-of-Product Expression (SOPE)*: Product terms connected by OR, e.g., $\overline{b}_2b_1b_0 + b_2$.

Given the set of shared data receivers S, the membership functions $f_S()$, which is in the form of SOPE, specifies the list of receivers:

$$f_S(b_1^u, b_2^u, \ldots, b_n^u) \quad = \quad \begin{cases} 0 & \text{iff} \quad u \in S, \\ 1 & \text{iff} \quad u \notin S. \end{cases}$$

For example, if the subgroup $S = \{000, 001, 011, 111\}$, then $f_S = \overline{b}_0\overline{b}_1\overline{b}_2 + \overline{b}_0\overline{b}_1b_2 + \overline{b}_0b_1b_2 + b_0b_1b_2$.

Then, the DO runs the Quine-McCluskey algorithm [21] to reduce f_S to minimal SOPE f_S^{min}. The reduction can consider *do not care* values $*$ on those IDs that are not currently assigned to any DO to further reduce number of product terms in the membership function. For example, if $S = \{000, 001, 011, 111\}$, $f_S^{min} = \overline{b}_0\overline{b}_1 + b_1b_2$.

Since f_S^{min} is in the form of SOPE, $\mathcal{T}_{ESP}$ can be formulated in disjunctive normal form (DNF). That is, for each product term E in f_S^{min}, the DO specifies an product term W using the following rules:

1) For positive literal $b_i \in f_S^{min}$, set B_i^+ in W.
2) For negative literal $\overline{b}_i \in f_S^{min}$, set B_i^- in W.

In consequence, the access policy $\mathcal{T}_{ESP}$ is in the following format: $T_{ESP} = W_1 \bigvee \cdots W_k$. For example, if $S = \{000, 001, 011, 111\}$, $f_S^{min} = \overline{b}_0\overline{b}_1 + b_1b_2$. We can find that f_S^{min} contains 2 product terms and T_{ESP} contains 2 AND gates connected by the root OR gate.

Finally, DO uploads the data blocks and the control block to SSP, where each data block is encrypted by the DEK and DEK is protected by the access policy in control block.

D. Data Updates

Now, we investigate into how to efficiently handle the data updates, i.e., how to modify encrypted data with or without changing data access control policy.

1) Data Updates With Access Policy Change: In Section IV-A, we described the "lazy" re-encryption strategy adopted by DOs. Using the "lazy" re-encryption scheme, the DO continuously records the revoked data receivers. When there is a need to modify the data, the DO will choose a new data access tree that can revoke all previously recorded data receivers.

When DO updates a data block with access policy change, we need to consider the following cases:

- If there is no control block associated with the latest access policy, i.e., no data updates occurred after the latest access policy change event, the DO encrypt a new random DEK associated with the latest access policy with PP-CP-ABE and attach a new control block to the end of the file, see Figure 4.

- If there exists a control block associated with the latest access policy, i.e., at least one data block was encrypted with the newest access policy, the DO can simply re-direct the control block pointer, see Figure 4, to the control block associated with the latest access policy.
- If a control block is not pointed by any data block, this control block should be deleted.

2) Updates Without Access Policy Change:: If no change is required to the access policy, DO can simply perform the PP-CP-ABE scheme and upload the updated data block in the SSP. The Block ID and the pointer to control the block are not changed.

V. Evaluation

In this section, we first present the security assessments of the presented solution. Then, we present the computation, communication, and storage performance evaluation.

A. Security Evaluation

The data structure of ciphertext and private key in PP-CP-ABE is the same as the original BSW CP-ABE[3], thus PP-CP-ABE can be viewed as a variation of CP-ABE. However, in PP-CP-ABE, the access policy tree is constructed by two sub-trees $\mathcal{T} = \mathcal{T}_{ESP} \bigwedge \mathcal{T}_{DO}$. In general, $\mathcal{T}_{DO}$ contains a single attribute to reduce the computation and communication overhead. Thus, DO randomly specifies a 1-degree polynomial $q(x)$ and sets $s = q(0)$, $s_1 = q(1)$ and $s_2 = q(2)$. The tuple $\{s_1, \mathcal{T}_{ESP}\}$ is sent to ESP. It is easy to prove that, based on the threshold secret sharing scheme [26], for a given 1-degree polynomial $q(x)$, knowing s_1, secrets s and s_2 are information theoretically secure.

Based on the security assumptions presented in Section II-C, ESP, DSP and SSP are untrusted but honest service providers that will perform according to protocol and returns correct results. In order to compromise users' secret information, the ESP, DSP and SSP can perform collusion attacks. In this scenario, an authorized user u' who satisfies the access tree $\mathcal{T}$ provides his blinded private key $\widetilde{SK}$ to the DSP for decryption. Then, ESP and DSP can try to utilize the blinded private key of u' to derive M from $Me(g,g)^{\alpha s}$. ESP has s_1, and thus it can easily derive $e(g,g)^{\alpha s_1}$. This is because $e(g,g)^{\alpha}$ is available from the public parameters presented in (1). As the user u' satisfies the access policy $\mathcal{T}_{DO}$, DSP can derive the following values $e(g,g)^{r's_1}$, $e(g,g)^{r's_2}$, $e(g,g)^{r's}$, and $e(g,g)^{t\alpha s + tr's}$ through the F_x function (see (4)) without knowing $alpha$ and r'. In the following table, we listed all rational terms that are available to ESP and DSP.

ESP	s_1	$e(g,g)^{\alpha s_1}$	$g^{\beta s_1}$	$g^{s_1/\beta}$
DSP	$e(g,g)^{r's_1}$	$e(g,g)^{r's_2}$	$e(g,g)^{r's}$	$e(g,g)^{t\alpha s+tr's}$

As we can see, ESP has the values s_1 and $e(g,g)^{\alpha s_1}$, but it is unaware of values s_2 or s. DSP possesses more terms as well as the blinded private key $\widetilde{SK}$ of u' (see (2)). We must note that $\widetilde{SK}$ is not a valid CP-ABE private key, since the $\widetilde{D} = g^{t(\alpha+r')/\beta}$ is embedded with tr' and $t\alpha$, and the rest of all private key components $\{\forall j \in S : D_j = g^{r'} \cdot H(j)^{r_j}, D'_j = g^{r_j}\}$ are embedded with r'. Essentially, this blinded private key can be a valid CP-ABE private key when (i) the master key is $MK = \{\beta, g^{t\alpha}\}$; (ii) a colluding user contributes $D = g^{t(\alpha+r')/\beta}$, which is a valid component embedded with tr'; and (iii) a colluding user contributes $\{\forall j \in S : D_j = g^{r'} \cdot H(j)^{r_j}, D'_j = g^{r_j}\}$, which are binded by a random r', which is different from tr' in D. Since the t is the exponent of the generator g, deriving it is equivalent to solve the DLP problem, which is considered to be hard. Thus, given the security of secret sharing and hardness of DLP on $\mathbb{G}_0$ and $\mathbb{G}_1$, ESP and DSP cannot derive $e(g,g)^{\alpha s_2}$ or $e(g,g)^{\alpha s}$ even if they collude.

B. Performance Evaluation

1) Computation Performance of PP-CP-ABE:: To evaluate the performance of the presented PP-CP-ABE scheme, we evaluate the computation overhead of service providers and users based on both theoretical analysis and experimental results.

Firstly, we analysis the number of expensive cryptographic operations over $\mathbb{G}_0$ and $\mathbb{G}_1$, i.e., pairing, exponentiation, multiplication, performed by service providers and users' devices. In our analysis we assume that the access policy T_{ESP} has a_1 attributes connected by an AND logical gate and T_{DO} only has 1 attribute. In addition, the root node is an AND gate.

In the following table, we compared the number of exponentiations, multiplications and hash to $\mathbb{G}_0$ operations incurred on ESP side and user side in the encryption outsourcing, where a_1 is the number of attributes in T_{ESP}:

	Exp $\mathbb{G}_0/\mathbb{G}_1$	Mul $\mathbb{G}_1$	Hash to $\mathbb{G}_0$
ESP	$2a_1/0$	0	a_1
User	3/1	1	1

We also provide a comparison of the number of exponentiations, multiplications, inversion, and pairing operations incurred by decryption outsourcing on DSP side and user side as shown in the following table, where a_1 is the number of attributes in T_{ESP}:

	Exp $\mathbb{G}_1$	Mul $\mathbb{G}_1$	Inv $\mathbb{G}_1$	Pairing
DSP	a_1	$2a_1$	a_1	$2a_1 + 1$
User	1	2	1	0

From the above analysis, we can see that the computation overhead is linear for service providers (ESP and DSP) and constant for the user. Among all operations, pairing is most computationally intensive.

We also conduct the experimental evaluation of cryptographic pairing and ECC operations on a wireless Mote sensor (8 bit-7.37 MHZ ATMega128L, 4KB RAM). The testing environments and results are listed in the following table:

	Pairing	Exp $\mathbb{G}_0$	Mul $\mathbb{G}_0$
Sensor	31250 ms	10720 ms	196 ms

Apart from the theoretical analysis, we also performed experimental measurements. Based on the CP-ABE open source project [3], we implemented and evaluated the PP-CP-ABE on a PC with 1.6GHz Intel Atom processor running Linux

2.6.32. The computation time is measured using clock ticks returned by `clock_t clock(void)` function in standard C library. To illustrate that most of the computation overhead is outsourced to service providers, we run the user and server on the same platform and recorded the number of clock ticks are recorded. In the Figure 5, we compared computation overhead incurred on service providers and users in encryption and decryption outsourcing. The computation overhead is calculated in terms of 10 based logarithms, i.e., $\log_{10}$, of thousands (K) clocks ticks. As we can see from the figure, more than 90% of encryption and more than 99% of decryption computation are performed by the service providers.

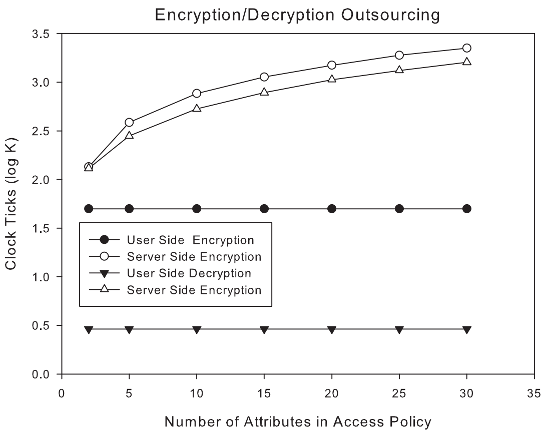


Fig. 5. Performance evaluation of the encryption and decryption outsourcing.

2) Storage Performance of ABDS: We analyze the storage performance of ABDS and compare it with several related cryptographic access control solutions: broadcast encryption schemes (Subset-Diff) [14], BGW broadcasting encryption [6], access control polynomial (ACP) scheme [29].

The performance is assessed in terms of cipher-text storage overhead, key storage overhead (system parameters and public/private keys stored on the users and TA). We denote the total number of users in the system with N and a user wants to share a file to any given set of receivers in the system. The comparative results are presented in Table II.

Ciphertext Storage Overhead In Subset-Diff scheme, the size of ciphertext is $O(t^2 \cdot log^2 t \cdot \log N)$, with t as maximum number of colluding users to compromise the ciphertext. For BGW scheme, the ciphertext size is $O(1)$ or $O(N^{\frac{1}{2}})$ as reported in [6]. In ACP scheme, the size of message depends on the degree of access control polynomial, which equals to the number of current receivers. Thus, the message size is $O(N)$. To control a set of receivers S using ABDS, the size of ciphertext depends on the number of product terms in the f_S^{min} (see IV-C). In [25], the authors derived an upper bound and lower bound on the average number of product terms in a minimized SOPE. Experimentally, the average number of message required is $\approx \log(N)$ [9].

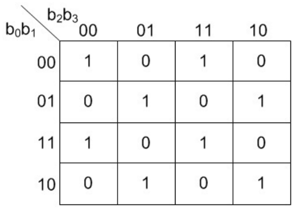
We examine some cases when ciphertext storage overhead is maximized.

Lemma 1 (multiple data receivers worst case): The worst case of sharing a file with multiple data receivers happens when both of following conditions hold: 1) The number of distinct receivers is $N/2$; 2) the Hamming distance between IDs of any two receivers is at least 2. In the worst case, the number of key updating messages is $N/2$. $\square$

Proof 1: Please refer to [8] for complete proof. $\square$
In this case, the number of product terms is $N - N/2 = N/2$ using ABDS. However, we can see that the worst cases happens in extremely low probability.

Lemma 2 (worst case possibility): When sharing a file with all data receivers in uniform probability, the worst case scenario happens with probability $\frac{1}{2^{N-1}}$. $\square$

Proof 2: In the worst case, the Hamming distance of IDs of $N/2$ receivers should be at least 2. As shown in the Karnaugh table in Figure 6, each cell represents an ID. For any cell marked 0 and any cell marked 1, the Hamming distance is at least 2. Thus, the worst case happens in two cases: (i) the encryptor wants to reach $N/2$ receivers marked 1 in Figure 6; (ii) the encryptor wants to reach $N/2$ receivers marked 0 in Figure 6 $\square$.



b_0b_1 \ b_2b_3	00	01	11	10
00	1	0	1	0
01	0	1	0	1
11	1	0	1	0
10	0	1	0	1

Fig. 6. Worst cases of broadcast encryption to $N/2$ receivers

To investigate the average case, we simulated ABDS in a system with 512 users and 1024 users, and the number of messages required are shown in Figure 7(a) and Figure 7(b) respectively. In the simulation, we consider the cases of 0%, 5%, 25%, 50% IDs are not assigned (i.e., *do not care* value). For each case, different percentages of receivers are randomly selected from the group. We repeat 100 times to average the results. Experimentally, the message size in CP-ABE starts at about 630 bytes, and each additional attribute adds about 300 bytes. Since the number of attributes in the access policy is bounded by $\log N$, we can conclude that the ciphertext storage overhead of ABDS is in the order of $O(\log^2 N)$.

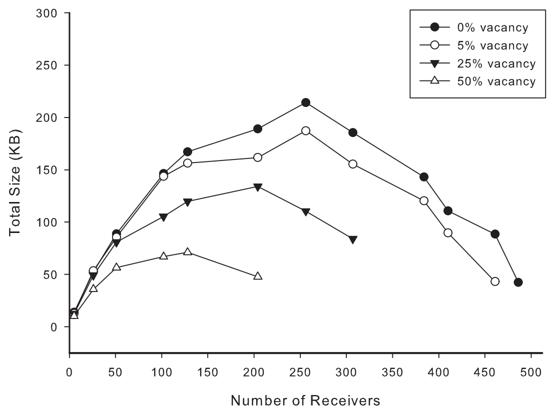
Key Storage Overhead Compared with Broadcast Encryption schemes, ABDS greatly reduced the Key Storage Overhead of the TA and users' devices. In ABDS, the PK and MK is of constant size. Also, a user needs to store $\log(N)$ bit-assignment attributes. Thus, the storage overhead is $O(\log N)$, assuming a user does not store any IDs of the data receivers. Although the DO may need the list of data receivers' IDs along with the list of *do not care* IDs to perform Boolean function minimization, we can argue that this does not incur extra storage overhead.

- The data publishers do not need to store the receiver's IDs after the broadcast; thus, the storage space can be released.

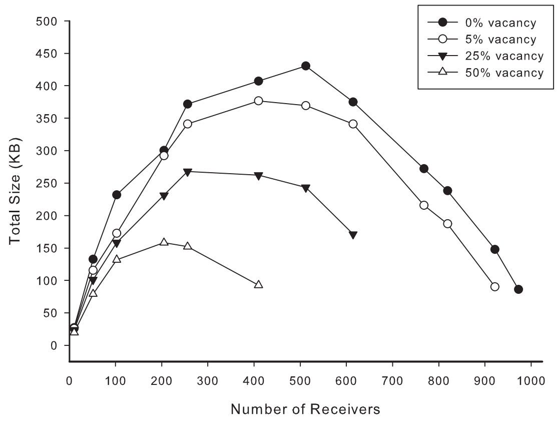
TABLE II
COMPARISON OF CIPHERTEXT STORAGE OVERHEAD AND KEY STORAGE OVERHEAD IN DIFFERENT CRYPTOGRAPHIC ACCESS CONTROL SCHEMES

Scheme	Ciphertext Storage		Key Storage	
	single data receiver	multiple data receivers	TA	User
ABDS	$O(\log N)$	$\approx O(\log^2 N)$	$O(1)$	$O(\log N)$
Subset-Diff	$O(t^2 \cdot log^2 t \cdot \log N)$	$O(t^2 \cdot log^2 t \cdot \log N)$	$O(N)$	$O(t \log t \log N)$
BGW_1	$O(1)$	$O(1)$	N/A	$O(N)$
BGW_2	$O(N^{\frac{1}{2}})$	$O(N^{\frac{1}{2}})$	N/A	$O(N^{\frac{1}{2}})$
ACP	$O(N)$	$O(N)$	$O(N)$	$O(1)$

N: the number of group members; t: maximum number of colluding users to compromise the ciphertext.



(a)



(b)

Fig. 7. (a) Size of ciphertext for group size 512; (b) Size of ciphertext for group size 1024.

- The TA can periodically publish the minimized SOPE of all *do not care* IDs, which can be used by data publishers to further reduce number of messages.
- If IDs are assigned to users sequentially, i.e., from low to high, TA can simply publish the lowest unassigned IDs to all users, who can use the all higher IDs as *do not care* values.
- Even if a user needs to store N IDs, the space is merely $N \log N$ bits. If $N = 2^{20}$.
- If a data publisher cannot utilize *do not care* values to further reduce the membership function in SOPE form, the ciphertext storage overhead might be a little higher. As shown in Figure 7(a)and Figure 7(b), the curve of 0% vacancy can also be used as ciphertext storage overhead required if a data publisher does not know the *do not care* IDs.

VI. RELATED WORKS

Existing works related to our proposed schemes includes (i) attribute based encryption and (ii) cryptographic access control over untrusted storage.

Attribute Based Encryption (ABE) was first proposed as a fuzzy version of IBE in [23], where an identity is viewed as a set of descriptive attributes. There are two main variants of ABE proposed so far, namely Key Policy Attribute Based Encryption (KP-ABE [17]) and Ciphertext Policy Attribute Based Encryption (CP-ABE [3]). In KP-ABE, each ciphertext is associated with a set of attributes and each user's private key is embedded with an access policy. Decryption is enabled only if the attributes on the ciphertext satisfy the access policy of the user's private key. In CP-ABE [3], [10], [20], [27], each user has a set of attributes that associate with user's private key and each ciphertext is encrypted by an access policy. To decrypt the message, the attributes in the user private key need to satisfy the access policy. CP-ABE is more appealing since it is conceptually closer to the Role Based Access Control (RBAC) [24] model.

Cryptographic access control over untrusted storage is investigated in both cryptography community and networking community. In cryptography community, Broadcast Encryption (BE) was introduced by Fiat and Naor in [15]. Compared with traditional one-to-one encryption schemes, BE is very efficient. Based on tradeoffs between key storage and ciphertext storage overhead, existing BE schemes can be generally categorized into the following classes: (i) constant ciphertext, linear public and/or private key on number of total receivers [5]; (ii) linear ciphertext on number of revoked receivers, constant (or logarithm) public and/or private key, [11], [22], [4]; (iii) sub-linear ciphertext, sub-linear public and/or private key [5]. In this work, we proposed a new construction of ABDS scheme to address the deficiency of all 3 class existing works. Particularly, ABDS supports any arbitrary number of receivers with much lower complexity of storage and communication.

In networking community, various encrypted file systems [19], [16], [2], [12] were proposed to secure data over untrusted storage. Particularly, in [2], the authors proposed a distributed storage scheme where users outsource encryption to a semi-trusted re-encryption server. However, if the server colludes with some malicious user, the data secrecy will be compromised completely. Compared with this scheme, our proposed PP-CP-ABE is secure even if service providers and malicious users collude. Recently, Yu et al. [28] proposed a security framework for cloud computing based on CP-ABE. Compared with our work, their solution requires the users to disclose part of original private key to the cloud while our solution only send blinded private keys. Moreover, our solution specially considers mobile cloud environments and their work.

VII. Conclusion

In conclusion, we proposed a holistic security framework for cloud data storage services to secure the data management in public clouds. Especially, our solution enables lightweight wireless devices to securely store and retrieve their data in public cloud with minimal cost. To this end, we proposed a novel Privacy Preserving Cipher Policy Attribute-Based Encryption (PP-CP-ABE) to protect users' encrypted data. Using PP-CP-ABE, light-weight devices can securely outsource intensive encryption and decryption operations to cloud service providers, without revealing the data content and used security keys. Also, we proposed an Attribute Based Data Storage (ABDS) system as a cryptographic access control mechanism. ABDS achieve information theoretically optimal in terms of minimizing computation, storage and communication overheads. Especially, ABDS minimize cloud costs charged by cloud service providers as well as communication overhead for data managements. Our performance assessments demonstrate the security strength and efficiency of our solution in terms of computation, communication, and storage.

Currently, PP-CP-ABE is based on BSW CP-ABE scheme[3], which suffers from linearly growing ciphertext size. We are investigating a CP-ABE scheme that is of constant ciphertext size and trying to propose a privacy preserving outsourcing scheme of the new CP-ABE scheme. Moreover, further performance improvements can be achieved by precomputing and caching some mostly used access policy trees from ESP. Another important future work will be implementation of a user space secure file system based on popular public cloud storage such that users can secure their cloud storage transparently.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, pages 598–609. ACM, 2007.

[2] G. Ateniese, K. Fu, M. Green, and S. Hohenberger. Improved proxy re-encryption schemes with applications to secure distributed storage. *ACM Trans. Inf. Syst. Secur.*, 9(1):1–30, 2006.

[3] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute-based encryption. In *SP '07: Proceedings of the 2007 IEEE Symposium on Security and Privacy*, pages 321–334, Washington, DC, USA, 2007. IEEE Computer Society.

[4] D. Boneh, X. Boyen, and E.J. Goh. Hierarchical identity based encryption with constant size ciphertext. *Advances in Cryptology–EUROCRYPT 2005*, pages 440–456, 2005.

[5] D. Boneh, C. Gentry, and B. Waters. Collusion resistant broadcast encryption with short ciphertexts and private keys. In *Advances in Cryptology–CRYPTO 2005*, pages 258–275. Springer, 2005.

[6] D. Boneh, A. Sahai, and B. Waters. Fully collusion resistant traitor tracing with short ciphertexts and private keys. pages 573–592, 2006.

[7] D. Boneh and B. Waters. Conjunctive, subset, and range queries on encrypted data. pages 535–554. Springer, 2007.

[8] I. Chang, R. Engel, D. Kandlur, D. Pendarakis, D. Saha, I.B.M.T.J.W.R. Center, and Y. Heights. Key management for secure lnternet multicast using Boolean functionminimization techniques. *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, 2, 1999.

[9] L. Cheung, J. Cooley, R. Khazan, and C. Newport. Collusion-Resistant Group Key Management Using Attribute-Based Encryption. Technical report, Cryptology ePrint Archive Report 2007/161, 2007. http://eprint. iacr. org.

[10] L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *CCS '07: Proceedings of the 14th ACM conference on Computer and communications security*, pages 456–465, New York, NY, USA, 2007. ACM.

[11] C. Delerablée, P. Paillier, and D. Pointcheval. Fully collusion secure dynamic broadcast encryption with constant-size ciphertexts or decryption keys. *Pairing-Based Cryptography–Pairing 2007*, pages 39–59.

[12] S. D. C. di Vimercati, S. Foresti, S. Jajodia, S. Paraboschi, and P. Samarati. Over-encryption: management of access control evolution on outsourced data. In *VLDB '07: Proceedings of the 33rd international conference on Very large data bases*, pages 123–134. VLDB Endowment, 2007.

[13] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. In *Proceedings of the 16th ACM conference on Computer and communications security*, pages 213–222. ACM, 2009.

[14] A. Fiat and M. Naor. Broadcast Encryption, Advances in Cryptology-Crypto93. *Lecture Notes in Computer Science*, 773:480–491, 1994.

[15] A. Fiat and M. Naor. Broadcast Encryption, Advances in Cryptology-Crypto93. *Lecture Notes in Computer Science*, 773:480–491, 1994.

[16] E.J. Goh, H. Shacham, N. Modadugu, and D. Boneh. SiRiUS: Securing remote untrusted storage. In *Proc. Network and Distributed Systems Security (NDSS) Symposium 2003*, pages 131–145, 2003.

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

[18] D. Huang, X. Zhang, M. Kang, and J. Luo. Mobicloud: A secure mobile cloud framework for pervasive mobile computing and communication. In *Proceedings of 5th IEEE International Symposium on Service-Oriented System Engineering*, 2010.

[19] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu. Plutus: Scalable secure file sharing on untrusted storage. In *FAST '03: Proceedings of the 2nd USENIX Conference on File and Storage Technologies*, pages 29–42, 2003.

[20] J. Katz, A. Sahai, and B. Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In *EUROCRYPT'08: Proceedings of the theory and applications of cryptographic techniques 27th annual international conference on Advances in cryptology*, pages 146–162, Berlin, Heidelberg, 2008. Springer-Verlag.

[21] E.J. McCluskey. Minimization of Boolean functions. *Bell System Technical Journal*, 35(5):1417–1444, 1956.

[22] D. Naor, M. Naor, and J. Lotspiech. Revocation and tracing schemes for stateless receivers. *Lecture Notes in Computer Science*, pages 41–62, 2001.

[23] A. Sahai and B. Waters. Fuzzy Identity-Based Encryption. In *Advances in Cryptology–Eurocrypt*, volume 3494, pages 457–473. Springer.

[24] RS Sandhu, EJ Coyne, HL Feinstein, and CE Youman. Role-based access control models. *Computer*, 29(2):38–47, 1996.

[25] T. Sasao. Bounds on the average number of products in the minimum sum-of-products expressions for multiple-value input two-valued output functions. *Computers, IEEE Transactions on*, 40(5):645–651, May 1991.

[26] A. Shamir. How to Share a Secret. *Communications of the ACM*, 22(11):612–613, 1979.

[27] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. *ePrint report*, 290, 2008.

[28] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing. *INFOCOM'1010. Conference of the IEEE Computer and Communications Societies. Proceedings. IEEE*, pages 1–9, 2010.

[29] X. Zou, Y.S. Dai, and E. Bertino. A Practical and Flexible Key Management Mechanism For Trusted Collaborative Computing. *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*, pages 538–546, 2008.