Method for Providing Secure and Private Fine-grained Access to Outsourced Data

Mosarrat Jahan[1,2] Mohsen Rezvani[1] Aruna Seneviratne[1,2] Sanjay Jha[1,2]

[1] University of New South Wales, Australia
{mjahan,mrezvani,sanjay}@cse.unsw.edu.au
[2] National ICTA (NICTA), Australia
Aruna.Seneviratne@nicta.com.au

Technical Report
UNSW-CSE-TR-201505
May 2015

School of Computer Science and Engineering
The University of New South Wales
Sydney 2052, Australia

Abstract

Outsourcing data to the cloud for computation and storage has been on rise in recent years. In this paper we investigate the problem of supporting write operation on the outsourced data for clients using mobile devices. Due to security concerns, data in the cloud is expected to be stored in encrypted form and associated with access control mechanism. In this work, we consider the Attribute based Encryption (ABE) scheme as it is well suited to support access control in outsourced cloud environment. Currently there is a gap in the literature on providing write access on the data encrypted with ABE. Moreover, since ABE is computationally expensive, it imposes processing burden on resource constrained mobile devices. Our work has two fold advantages. Firstly, we extend the single authority Ciphertext Policy Attribute based Encryption (CP-ABE) scheme to support write operations. We define a group among the set of authorized users for a ciphertext that can perform the write operation, while the remaining users can perform only read operation. Secondly in achieving this goal, we move some of the expensive computations to a manager and remote cloud server by exploiting their high-end computational power. Our security analysis demonstrates that the security properties of system are not compromised.

Keywords: Security; fine-grained access control; attribute based encryption (ABE); write operation.

1 Introduction

With the advent of cloud technology, users and enterprises are able to use infrastructure, storage and application services from third parties, namely Cloud Service Providers (CSPs). CSPs minimise the cost users get access to an infrastructure as required without having to build and maintain the infrastructure themselves. Moreover, cloud computing provides a number of advantages like higher availability and protection of data in case of disaster that can affect inhouse computing system. However, cloud services suffer from many security and privacy issues[8][14]. For example, data cannot be stored in the cloud in clear text as may be subject to the unauthorised or malicioius access. Efficient mechanisms are needed to read and write the outsourced data. There is an extensive study in literature to provide efficient access mechanisms and most of them only support read operations [1][8]. This becomes even a greater problem when information access needs to be provided to users with different privileges. For example, assume that a document needs to be shared between a doctor, nurse, physiotherapist and a social worker where the doctor has complete read-write access to the whole document, the nurse and the physiotherapist have access to different parts of the document, and the social worker only has read access.

Data encryption and access control mechanisms are closely related as encryption should be done by keeping in mind who will get the access of data. Access control on encrypted data is difficult due to the generation and management of keys and ciphertexts. In this scenario, Attribute based Encryption (ABE) is a promising solution to provide access control on outsourced data. It is an efficient cryptographic tool to establish secured communication among a group of users and ensure privacy of the participants. Compared to traditional Public Key Encryption (PKE) schemes, ABE stores a single copy of the ciphertext. Moreover, keys can be generated and managed by a single or multiple attribute authorities. In ABE, a user is represented as having a set of properties or attributes that ensures user privacy by hiding user identity. A user will be able to decrypt a ciphertext if and only if he can satisfy the access requirement of the ciphertext with the attributes he owns. Many work utilized ABE to provide fine-grained access control to outsourced data. However, most of them only support read operations[1][8].

Moreover with the rise of mobile devices, most of the users are accessing data using smart phones, tablets, etc that are constrained in terms of storage, processing and battery power. Thus there is an urgent need to design light weight applications that can be executed in the mobile devices and can access and process outsourced data from mobile devices.

In this paper, we present an efficient mechanism for performing fine-grained read and write operations on outsourced data that will enable a number of users access and manipulate the shared data. This paper makes the following contributions:

- Shows that its is possible for the owner to retain control over the data by prohibiting writers to modify any access policy;

- Provides a method to delegate expensive bilinear pairing operation to the cloud and manager servers without compromising security of the system thus enabling the use of ABE in mobile environments;

- Present a security analysis that shows the security of the system is maintained.

The rest of the paper is organized as follows. Section II lists the related work. Section III presents the system model and assumptions we make. Section IV describes the proposed algorithm in detail, while discussion and the security analysis are given in Section V and Section VI, respectively. Finally some conclusions and future directions are presented in Section VII.

2 Related Work

ABE is extensively studied in literature. Dan Boneh et al. proposed the notation of fully functional identity-based encryption (IBE) in 2001 [2]. Sahai and Waters developed the concept of Fuzzy-IBE in 2005 where the identity of a user is represented as a set of strings known as attributes [15]. They proposed an access control mechanism where a recipient can decrypt a ciphertext if there is match of d_k attributes between the ciphertext and the recipient where d_k is a predefined value. The idea of ABE emerged from the concept of Fuzzy IBE. Goyal et al. [7] proposed the concept of Key Policy Attribute Based Encryption (KP-ABE) where an access structure is included in the decryption key of a user while the ciphertext is associated with a set of attributes. A user can decrypt a ciphertext if and only if the attributes of the ciphertext can satisfy the access policy of the decryption key. In KP-ABE, the key generator will determine the types of data a user can access. Ciphertext Policy Attribute Based Encrytion (CP-ABE) is proposed by Bethencourt's et al. in 2007 [1]. CP-ABE is a complementary concept of KP-ABE as the access structure is associated with ciphertext and set of attributes are inscribed into the decryption key. In CP-ABE the data owner is solely responsible for determining who will access the data. Bethencourt's CP-ABE is secure in generic group model. Many works were done to make CP-ABE secure in standard model but with cost of expressiveness of access structure [5]. Among all the work, B. Water's CP-ABE is prominent as he proposed an expressive CP-ABE in standard complexity model [17]. In all of the above schemes, a user has to go to a single trusted authority to prove his identity to obtain the decryption key. This approach is not secured as the compromise of the single authority will make all the ciphertexts belonging to the system easily decryptable because the central authority contains the master secret key used to generate all the decryption keys. Thus research is conducted to divide the responsibility of key generations into a number of attribute authorities. Chase et al. first proposed the concept of Multi-Authority Attribute Based Encryption (MA-ABE) by extending the single authority Fuzzy IBE to handle multiple Attribute Authorities (AA) [3]. This scheme does not ensure user privacy as user identity GID is used to authenticate and corrupted AAs can link the related attribute keys issued by different AAs for a single user using the GID. Chase and Chow addressed the problem of [3] by distributing the operation of CA among a number of AAs and by generating different pseudonyms for a single user when he contacts with different AAs [4]. The shortcoming of the scheme is the communication and computational overhead during setup and key generation phase due to co-operations among the AAs. Han et al. [9] proposed a privacy preserving multi-authority KP-ABE scheme. Each AA works independently and can generate secret key for a user without co-operation with other AAs and each authority has no information about user identity. Hur et al.[10] proposed a multi-authority CP-ABE (MA-CPABE) scheme for Disruption Tolerant Networks (DTN). In this scheme a number of AAs and users contribute to generate the decryption key under the control of a CA. This involves communication overhead among the CA, AAs and users. Lewko et al. [11] proposed a decentralized CP-ABE scheme without any CA and their scheme is applicable for expressive access policy.

Both forms of the ABE schemes proposed only support read operations [1][7][8]. Recently, Zhao et al. [18] proposed the design of flexible read/write framework for outsourced data utilizing the concept of CP-ABE[1] and an Attribute Based Signature (ABS) scheme[13]. In the proposed scheme, the data owner encrypts the file using an access policy $T_{decrypt}$ and then signs the file by using signature access policy T_{sign}. An authorized reader verifies the signature and decrypts the ciphertext. A writer can encrypt the modified file with a new access policy $T_{decrypt_1}$ and sign the file with T_{sign}. The CSP will verify the new sign SG_1 with T_{sign} and the verification key. However this is based on a single authority CP-ABE and ABS and they allow any writer to change the access policy for the data. Ruj et al. [14] extended this work to provide a decentralized version by combining Lewko's decentralized ABE scheme [11] with multiple authority ABS proposed by Maji et al.[13]. This enables users of the data to both modify the T_{sign} and $T_{decrypt}$, as a result will ultimately reduce data owner's control over the data. More recently Dong et al. [6] utilized hierarchical Identity Based Encryption (IBE) to provide secure, efficient and scalable data sharing mechanism where once again any writer can specify the access policy. In this scheme, a root-PKG generates system parameters, a secret key and private keys for the lower levels L-PKGs and D-PKGs. Each L-PKGs selects a secret key for itself and private keys for the L-PKGs and D-PKGs under its control. D-PKG is responsible for generating private keys for users in the same domain. Data owner encrypts data using IBE scheme and the public keys of the recipients that can also modify the data. A user with the decryption key can decrypt and update the data if it's public key belongs to the encrypted data. Liu et al. [12] proposed a privacy preserving data access control mechanism by combining CP-ABE [1] with the hierarchical structure of multiple authorities. Their scheme supports write operation using the concept of Attribute Based Signature (ABS) [13] scheme. In this mechanism, a certificate authority (CA) will perform system setup and generates system public key and master key. When a new Attribute Authority (AA_i) with identity aa joins the system, CA selects the secret key for AA_i using key generation rule of the single authority CPABE [1]. Top level AA_i will generate keys for the users or the underlying AAs by using the concept of key delegation of the single authority CPABE [1]. Data encryption and decryption is performed using the same rule of single authority CPABE [1]. A user who wants to perform write operation re-encrypts M to M' and sign the key to generate a signature σ. The cloud server will check this σ with respect to T_{sign} and it verifies successfully, the cloud will accept the ciphertext. In all of these schemes, the data owner's control is not fully maintained as writers can modify the access policy and they are not suitable for

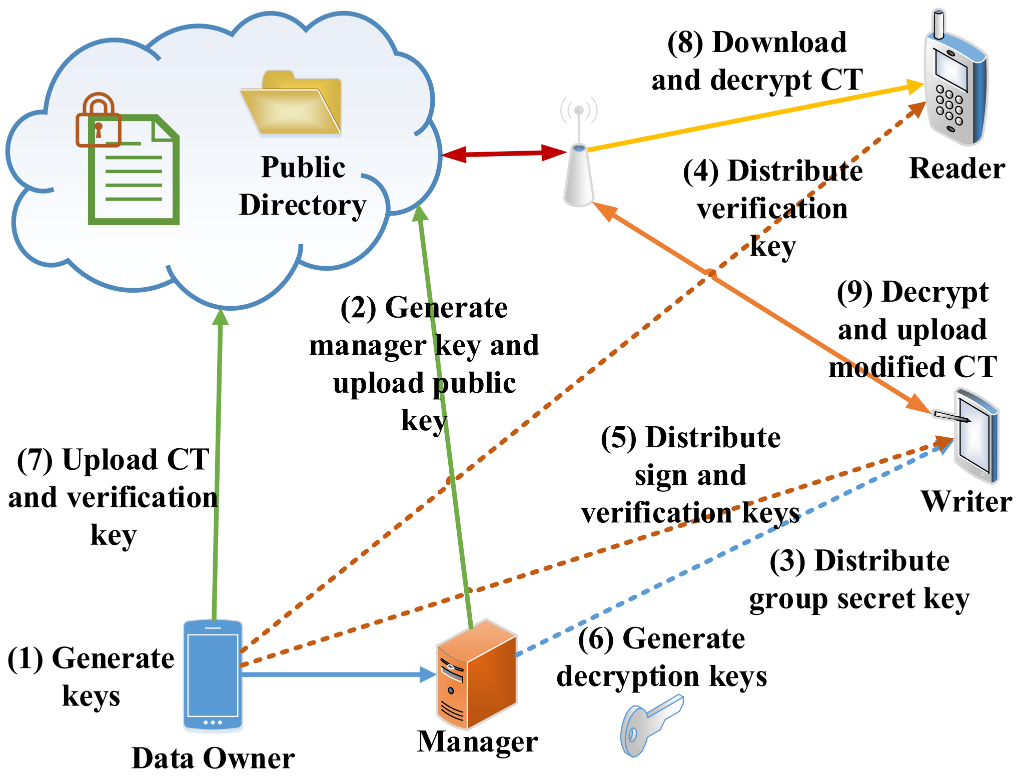
(8) Download and decrypt CT
Public Directory
Reader
(4) Distribute verification key
(9) Decrypt and upload modified CT
(2) Generate manager key and upload public key
(7) Upload CT and verification key
(5) Distribute sign and verification keys
Writer
(3) Distribute group secret key
(6) Generate decryption keys
(1) Generate keys
Data Owner
Manager

Figure 3.1: Our system model.

processing data in mobile devices because of the high computational overheads associated with the building blocks. We address these issues in our proposed work.

3 System Architecture

In this section, we describe the system architecture and define the security model.

3.1 System Description and Assumptions

Figure 3.1 shows the system architecture for collaborative data sharing among a group of users within a company. Our scheme extend Bethencourt's CP-ABE[1] to support this operation. The notations used in the description are summarized in Table 3.1. The system model consists of the following entities:

1. **Data Owner**, who is responsible for originating an outsourced data that is shared with a number of users, each allowed to either read, write or do both operations. The **data owner** generates owner secret key, OSK and owner public key, OPK. The **data owner** also generates group secret key, GSK and group public key, GPK, what will indicate who is authorized to write on the outsourced data. The OPK and GPK will become part of a public key, PK. For each outsourced data the **data**

Table 3.1: List of notations

Symbol	**Description**
OSK	Owner secret key
OPK	Owner public key
GSK	Group secret key
GPK	Group public key
Sk	Signing key
Vk	Verification key
MSK	Manager secret key
CK	Cloud key
CSP	Cloud service provider
CT	Ciphertext

owner also generates a signing key Sk and a verification key Vk. Either **data owner** can share GSK with the group users or delegate a managing entity to do this. The **data owner** shares Vk with CSP as well as with all authorized users whereas shares Sk only with the writers. We assume that all communications among the data owner, where appropriate the manager and users will be done using a secure channel, for example SSH and the authorized users will not collude.

2. **Manager** is an internal trusted server running within the owner's network and generates decryption keys on behalf of the data owner[16]. The proposed system, exploits the capabilities of the **manager** to reduce computational burden on user side.

3. **Cloud Server** stores encrypted data and provides access to the ciphertext, CT to the users with valid Vk. It is assumed that **cloud server** is honest-but-curious, namely it may try to gather information on the encrypted data but will not tamper it [16]. In the proposed system, the processing power of the **cloud server** is used to securely perform computationally expensive bilinear mapping operations.

4. **Users** access encrypted data using their mobile devices and can decrypt if authorized. Moreover, only a subset of the authorized users will be able to perform write operations.

3.2 Threat Model and Security Requirements

- **Data Confidentiality:** Unauthorized users who do not satisfy the access policy should not be able to decrypt the ciphertexts. Data should be protected against CSP and manager server.

- **Collusion Resistant:** Collusion attacks are possible only when two or more unauthorized users can combine their attributes to satisfy the access policy and get the access of data. Our scheme should be protected against such type of attacks.

- **Replay Attack:** Replay attack is performed when a malicious user having the access of old copy of CT replaces the valid CT with the obsolete one. As our scheme supports write operation, the old CT in the cloud should be replaced by the valid CT written by an authorized writer and this scheme should be protected against replay attack.

- **Prevent Unauthorized Write Operation:** The system should be secured against write operation by unauthorized users, CSP and manager.

- **Resistant against Compromising GSK:** In our scheme GSK is used to define the group of writers among the set of authorized users. Thus compromise of GSK creates provision for revoked writers to perform write operations and such situations should be avoided.

4 Proposed Scheme

In this section we describe the construction to accomplish the data sharing operation in the scenario depicted in Figure 3.1. Data owner forms a group by selecting users among the authorized users using the group formation technique described in [16]. This includes a subset of authorized users able to perform write operations.

4.1 Scheme Construction

Let G_0 be a bilinear group of prime order p with generator g and $e : G_0 \times G_0 \rightarrow G_1$ be a bilinear map. We also utilize a one way hash funcion H that maps an attribute to a random group element in G_0. Our construction consists of eight steps following the execution sequence depicted in Figure 3.1.

Data Owner Setup(): The data owner generates owner secret key, $OSK = \{\beta\}$ and owner public key, $OPK = \{g^{\beta}, g^{1/\beta}\}$, where $\beta \in Z_p$. The data owner also selects a group of users for write operation and generates group secret key, $GSK = u_0$ and group public key, $GPK = g^{u_0}$ that will become part of Public Key, PK, where $u_0 \in Z_p$. For each outsourced data the data owner selects Sk and Vk (Step 1 in Figure 3.1).

Manager Setup(): The manager creates manager secret key, $MSK = (\alpha, g^{\alpha}, \alpha_1, \alpha_2)$ and calculates $e(g, g)^{\alpha}$, where $\alpha = \alpha_1 + \alpha_2$ and $\alpha, \alpha_1, \alpha_2 \in Z_p$. The manager also generates a cloud key, $CK = (g^{\alpha_2/\beta}, V_{no})$ (Step 2 in Figure 3.1). Finally the public key, PK is generated using both a group public key, GPK, supplied by the data owner, and a cloud key, CK, provided by the manager as:

$$PK = \{G_0, g, h = g^{\beta}, f = g^{1/\beta}, e(g, g)^{\alpha}, GPK, CK, V_{no}\},$$

where G_0 is a bilinear group of prime order p with generator g and V_{no} is the version number of PK.

KeyDistribution(): The data owner or manager can distribute GSK to every member of the group using a secure communication channel (Step 3). The data owner transmits Vk to the cloud and every authorized user (Step 4) and Sk to the group members (Step 5) again using a secure communication channel.

KeyGen(PK, MSK, A)$\rightarrow$ DSK: The manager generates data secret key, DSK on behalf of the data owner by selecting a random $r \in Z_p$ for a particular user and $r_j \in Z_p$ for each attribute $j \in A$ (Step 6). DSK is generated as follows:

$$DSK = (D = g^{(\alpha_1+r)/\beta}, \forall j \in A : D_j = g^r.H(j)^{r_j}, D'_j = g^{r_j}),$$

where H is hash function that maps an attribute to a random group element in G_0.

Encrypt(PK, GPK, M, T)$\rightarrow$ CT: The data owner determines an access tree T representing the access policy following the method described in [1]. A polynomial q_x is generated for each node $x \in T$. A randomly selected secret value $s \in Z_p$ is associated with the root R of T by setting $q_R(0) = s$. The data owner also attaches a time stamp τ to prevent replay attacks[14]. Then the ciphertext, CT is generated as follows:

$$\begin{aligned} CT =& \{v = 0, \tau, T, C = M.e(g, g)^{\alpha s}, C_{0_{grp}} = g^{u_0 s}, C'' = g^{\beta s}, \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(atty(y))^{q_y(0)}\}. \end{aligned}$$

where Y is the set of leaf nodes of T and v is the version number of CT.

The manager shares α with the group members by encrypting it with GPK. The CSP also computes $e(g, g)^{u_0 s}$ using $C_{0_{grp}}$ and g and generates C'= $M.e(g, g)^{\alpha s}.e(g, g)^{u_0 s}$.The final CT generated by the data owner is,

$$\begin{aligned} CT =& \{v = 0, \tau, T, C' = M.e(g, g)^{\alpha s}.e(g, g)^{u_0 s}, \\ & C = M.e(g, g)^{\alpha s}, C_{0_{grp}} = g^{u_0 s}, C'' = g^{\beta s}, \\ & \forall y \in Y : C_y = g^{q_y(0)}, C'_y = H(att(y))^{q_y(0)}\}. \end{aligned}$$

The data owner signs CT using Sk and uploads the CT to the cloud (Step 7 in Figure 3.1). To further improve security, the data owner doesn't provide the manager any access to CT.

UploadCTCloud(): Cloud first verifies τ and then proceeds to check the validity of the signature on CT using Vk before storing CT.

Decrypt(CT, DSK, PK, GSK)$\rightarrow$ M: A user requiring access to CT presents Vk to the cloud, and if it matches with the Vk present in the cloud, then he gets to download CT. The user verifies the signature on CT. If it is a valid, the user will try to match it's attributes with the leaf nodes of T by evaluating the following equation for each leaf node x:

$$\begin{aligned} DecryptNode(CT, DSK, x) =& e(D_i, C_x)/e(D'_i, C'_x) \\ =& e(g^r H(i)^{r_i}, g^{q_x(0)})/e(g^{r_i}, H(i)^{q_x(0)}) \\ =& e(g, g)^{rq_x(0)}, \end{aligned}$$

where $i = att(x)$. The values obtained from the leaf nodes are then used recursively to evaluate the value of non-leaf nodes as described in [1]. Finally if the user attributes are able to satisfy the access policy, the value of the root node is determined as follows:

$$A = DecryptNode(CT, DSK, R) = e(g, g)^{rs}$$

The decryption operation proceeds as follows:

1)Read Operation by the Authorized Users (Step 8 in Figure 3.1) (If the user does not have Sk from the data owner): User evaluates the following equation:

$$\begin{aligned} M &= C.A/e(C'', D)e(C'', CK) \\ &= M.e(g,g)^{\alpha s}.e(g,g)^{rs}/e(g^{\beta s}, g^{(\alpha_1+r)/\beta})e(g^{\beta s}, g^{\alpha_2/\beta}) \\ &= M.e(g,g)^{(\alpha s+rs)}/e(g,g)^{(\alpha_1+r)s}e(g,g)^{\alpha_2 s} \\ &= M.e(g,g)^{(\alpha s+rs)}/e(g,g)^{(\alpha_1 s+rs+\alpha_2 s)} \\ &= M.e(g,g)^{(\alpha s+rs)}/e(g,g)^{(\alpha s+rs)}. \end{aligned}$$

The computation of $e(C'', D)$ is done by the manager and user computes $e(C'', CK)$ and $e(C'', D)e(C'', CK)$.

2) Read and Write Operation by the Authorized Group of Users (Step 9 in Figure 3.1) (If the user has Sk from the data owner): Message is decrypted as follows:

$$\begin{aligned} M &= C'.A/e(C'', D).e(C'', g^{u_0/\beta}).e(C'', CK) \\ &= M.e(g,g)^{\alpha s}.e(g,g)^{u_0 s}.e(g,g)^{rs}/e(g^{\beta s}, g^{(\alpha_1+r)/\beta}).e(g^{\beta s}, g^{u_0/\beta}).e(g^{\beta s}, g^{\alpha_2/\beta}) \\ &= M.e(g,g)^{(\alpha s+rs+u_0 s)}/Me(g,g)^{(\alpha_1+r)s}.e(g,g)^{u_0 s}.e(g,g)^{\alpha_2 s} \\ &= M.e(g,g)^{(\alpha s+rs+u_0 s)}/e(g,g)^{(\alpha s+rs+u_0 s)} \end{aligned}$$

The computations of $e(C'', D)$ and $e(C'', CK)$ are done by the manager, the computation of $e(C'', g^{u_0/\beta})$ and $e(C'', D)e(C'', CK)e(C'', g^{u_0/\beta})$ are performed by the user.

Suppose M is modified as M' and to re-encrypt M' we need to generate $C' = M'.e(g,g)^{\alpha s}e(g,g)^{u_0 s}$. The user performs the following operations:

1. User calculates pairing of C'' (contained in CT) and $g^{1/\beta}$(contained in PK) and this generates $e(g^{\beta s}, g^{1/\beta}) = e(g,g)^s$.
2. User computes $(e(g,g)^s)^\alpha = e(g,g)^{\alpha s}$
3. CSP calculates $e(g,g)^{u_0 s}$ from $C_{0_{grp}} = g^{u_0 s}$
4. User calculates $e(g,g)^{\alpha s}e(g,g)^{u_0 s}$.
5. User generates $C' = M'.e(g,g)^{\alpha s}e(g,g)^{u_0 s}$ and $C = M'.e(g,g)^{\alpha s}$
6. User selects a time stamp τ.
7. To generate new CT user extracts $T, C'', C_{0_{grp}}, \forall y \in Y : C_y, C'_y$ from the CT received, thus preventing the user from generating a new access tree T. The version number of CT will be increased by 1.
8. User will sign the CT using Sk.

KeyReGen(): When a member in the group is revoked, a new user is added or a key is compromised, the manager selects $\alpha' \in Z_p$ and chooses a new $CK = g^{\alpha'_2/\beta}$ where $\alpha'_2 = \alpha' - \alpha_1$. The new $PK = \{G_0, g, h = g^\beta, f = g^{1/\beta}, e(g,g)^{\alpha'}, GPK, CK, V_{no}\}$ and $MSK = (\alpha', g^{\alpha'}, \alpha_1, \alpha'_2, V_{no})$ where V_{no} is increased by 1. We keep α_1 fixed and modify α' and α'_2 to prevent the need

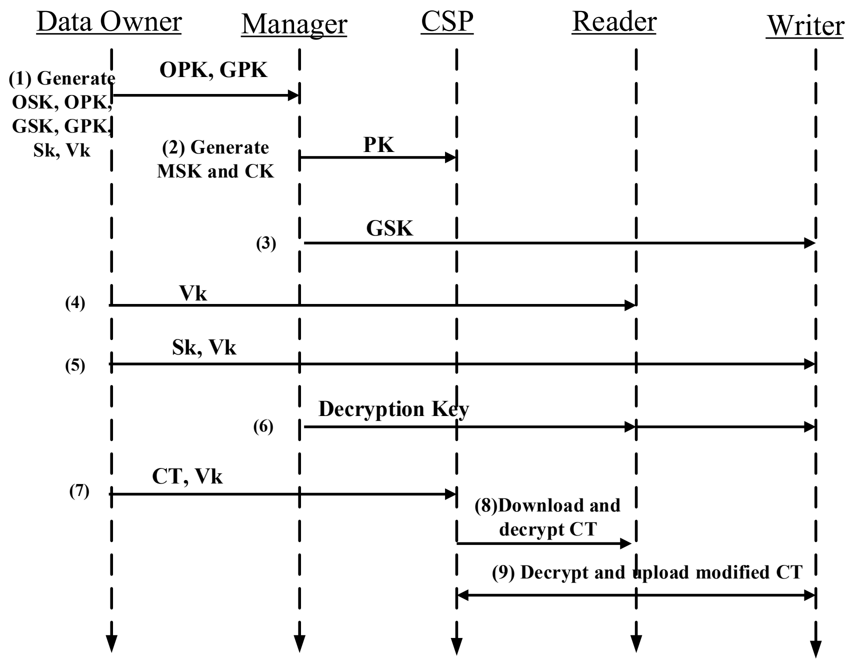

Figure 4.1: An illustration of outsourced data sharing among a group of users

to modify all the decryption keys generated so far using α_1. α_1 and α'_2 will ultimately produce α' during the decryption operation.

The data owner updates $GSK = u_x$ and $GPK = g^{u_x}$. GSK is distributed among the group members using secure channel. The manager also distributes encrypted value of α' using GPK. Data owner modifies C as $M.e(g,g)^{\alpha' s}$ and sends C and $C_{x_{base}} = g^{u_x s}$ to the cloud. To prevent CT from being decrypted by the revoked users (writer), the CT is re-encrypted by the CSP using the re-encryption key. The re-encryption key $RK_{0\to x}$ can be calculated by the data owner as well as the holder of GSK as $g^{\{u_x/u_0\}}$ and is sent to the cloud. The cloud re-encrypts $\tilde{C_x} = C.e(C_{0_{grp}}, RK_{0\to x}) = M.e(g,g)^{\alpha' s}.e(g^{u_0}s, g^{u_x/u_0}) = M.e(g,g)^{\alpha' s}.e(g,g)^{u_x s}$. The data owner combines the final ciphertext as follows:

$$CT_x = \{v = x, T, \tilde{C_x}, C, C'', C_{x_{base}}, \forall y \in Y : C_y, C'_y\}$$

Finally data owner changes the Sk and Vk whenever there is a change in group membership.

The cryptographic operations performed in time order is shown in Figure **??**

5 Discussion

Traditional CP-ABE[1] uses α as a secret to generate the decryption keys. Any one having the access of α will be able to generate decryption keys and thus able to decrypt all ciphertexts. In our case α can be leaked as it is shared with a number of users. As a safety measure, we divide α into two parts α_1 and α_2 and utilize α_1 to generate the decryption keys. Thus even if α is compromised, decryption keys generated so far using α_1 remain safe. Our scheme adopts new values α' and α_2' by keeping α_1 fixed. The data owner modifies C and $\tilde{C_x}$ using α', manager computes $e(g,g)^{\alpha'}$ and a new CK using α_2'. The decryption keys will be keep generated using α_1 and combined with α_2' during decryption phase to generate α'. Thus when α is compromised, our scheme only needs to generate α', CK, C, $\tilde{C_x}$, Sk and Vk, eliminating the need for modifying the decryption keys that is a huge saving in terms of computational overhead. Any malicious user having access of α cannot decrypt any CT. Thus secure write operation is ensured in our system.

6 Security Analysis

Our scheme is secured as a generic group model relying on Bethencourt's CP-ABE model[1] and is resistant to the following attacks:
Replay Attack: As τ maintained in CT will prevent replay attack. The cloud will verify τ with respect to the current time stamp of the existing CT. If τ has greater value than the existing τ in CT and CT is verified with Vk, then CSP accepts the new CT replacing the previous one. This prevents a reader trying to replace the existing CT with an old copy of CT signed by previous writer. The reader cannot attach new τ with CT and sign it as does not have the access of Sk. When a writer is revoked, Sk and Vk gets changed. Thus a revoked writer cannot use an obsolete copy of CT signed with previous Sk to replace the existing CT.

Prevent Unauthorized Users, CSP and Manager to perform Write Operation: For the lack of Sk and α unauthorized users (even reader) as well as the CSP cannot write data. Although manager has α, it cannot perform the write operation because the data owner always uploads CT into the cloud directly without giving it any access. Manager does not have the access of Sk and Vk and thus cannot download CT from the cloud.

Collusion Resistant: Our construction follow Bethencourt's CPABE scheme which is resistant to collusion attacks[1]. Thus two users each having insufficient attributes to decrypt CT cannot combine their attributes to decrypt CT.

CSP and Manager cannot Decrypt CT: Cloud is unable to decrypt CT as it does not have the access to DSK, s, α. Although manager has DSK it is not given the access of CT. Even when expensive operations are delegated to the CSP and manager, it is carefully designed so that no one party get enough information to decrypt CT.

Resistant against compromising GSK: If GSK is compromised, two situations can arise:

1. the data owner will select a new GSK, u_x and instruct CSP to re-encrypt CT using $R_{0\to x}$ and prevent revoked users to perform write operation.

2. An unauthorized user will get the access to α. In this case our scheme adopts a new α' that will only change α_2' without changing α_1. C, $\tilde{C_x}$, Sk and Vk are modified by the data owner.

7 Conclusion

This paper extended the single authority CP-ABE scheme[1] to support write operation. To this end, we full fill security requirements of the proposed system and explored the capacity of manager and cloud servers to reduce computation overhead on mobile users. In future we plan to reduce trust level on manager server and make the sign and verfication keys non-transferrable. Finally we will try to deploy the proposed scheme using existing CP-ABE[1] libraries in practical scenarios that involve interactions among cloud, server and mobile device users and evaluate the performance.

Bibliography

[1] J. Bethencourt, A. Sahai, and B. Waters. Ciphertext-policy attribute based encryption. In *Proc. of IEEE SP'2007*, 2007.

[2] D. Boneh and M. Franklin. Identity-based encryption from the weil pairing. In *Proc. Adv. Cryptol.(CRYPTO'01)*, volume 2139, pages 213–229, 2001.

[3] M. Chase. Multi-authority attribute based encryption. In *Proc. of 4th Conference on Theory of Cryptography(TCC'07)*, pages 515–534, 2007.

[4] M. Chase and S. S. M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proc. of 16th ACM Conf. Comp. Commun. Security*, pages 121–130, 2009.

[5] L. Cheung and C. Newport. Provably secure ciphertext policy abe. In *Proc. 14th ACM Conf. Computer and Comm. Security(CCS'07)*, pages 456–465, 2007.

[6] Xin Dong, Jiadi Yu, Yanmin Zhu, Yingying Chen, Yuan Luo, and Minglu Li. Seco: Secure and scalable data collaboration services in cloud computing. *Computers & Security*, 50:91–105, 2015.

[7] V. Goyal, O. Pandey, A. Sahai, and B. Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proc. of 13th ACM CCS'2006*, 2006.

[8] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. In *USENIX Security Symposium*, page 3, 2011.

[9] Jinguang Han, Willy Susilo, Yi Mu, and Jun Yan. Privacy-preserving decentralized key-policy attribute-based encryption. *IEEE Trans. Parallel and Distributed Systems*, 23(11):2150–2162, November 2012.

[10] Junbeom Hur and Kyungtae Kang. Secure data retrieval for decentralized disruption-tolerant military networks. *IEEE/ACM Trans. Netw.*, 22(1):16–26, February 2014.

[11] Allison Lewko and Brent Waters. Decentralizing attribute-based encryption. In *Proc. of 30th EUROCRYPT'2011)*, 2011.

[12] Xuejiao Liu, Yingjie Xia, Shasha Jiang, Fubiao Xia, and Yanbo Wang. Hierarchical attribute-based access control with authentication for outsourced data in cloud computing. In *Proc. of 12th IEEE TrustCom'2013*, 2013.

[13] Hemanta K Maji, Manoj Prabhakaran, and Mike Rosulek. Attribute-based signatures. In *Topics in Cryptology–CT-RSA 2011*, pages 376–392. Springer, 2011.

[14] Sushmita Ruj, Milos Stojmenovic, and Amiya Nayak. Decentralized access control with anonymous authentication of data stored in clouds. *IEEE Trans. Parallel and Distributed Systems*, 25(2):384–394, 2014.

[15] A. Sahai and B. Waters. Fuzzy identity-based encryption. In *Proc. 24th Annual Intl. on Theory and Applications of Cryptographic Techniques (EUROCRYPT'05)*, volume 3494, pages 457–473, 2005.

[16] Piotr K Tysowski and M Anwarul Hasan. Hybrid attribute-and re-encryption-based key management for secure and scalable mobile applications in clouds. *IEEE Trans. Cloud Computing*, 1(2):172–186, 2013.

[17] B. Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. *Public Key Cryptography (PKC'2011)*, 6571:53–70, 2011.

[18] Fangming Zhao, Takashi Nishide, and Kouichi Sakurai. Realizing fine-grained and flexible access control to outsourced data with attribute-based cryptosystems. In *Information Security Practice and Experience*, pages 83–97. Springer, 2011.