Control Cloud Data Access Privilege and Anonymity With Fully Anonymous Attribute Based Encryption

Taeho Jung[1], Xiang-Yang Li[12], Zhiguo Wan[34], Meng Wan[5]
[1]Department of Computer Science, Illinois Institute of Technology, Chicago, IL
[2]Department of Computer Science and Technology, TNLIST, Tsinghua University, Beijing
[3]School of Software, TNLIST, Tsinghua University, Beijing
[4]State Key Laboratory of Information Security, Chinese Academy of Sciences, Beijing
[5]Center for Science and Technology Development, Ministry of Education, Beijing

***Abstract*—Cloud computing is a revolutionary computing paradigm which enables flexible, on-demand and low-cost usage of computing resources, but the data is outsourced to some cloud servers, and various privacy concerns emerge from it. Various schemes based on the Attribute-Based Encryption have been proposed to secure the cloud storage. However, most work focuses on the data contents privacy and the access control, while less attention is paid to the privilege control and the identity privacy. In this paper, we present a semi-anonymous privilege control scheme *AnonyControl* to address not only the data privacy but also the user identity privacy in existing access control schemes. *AnonyControl* decentralizes the central authority to limit the identity leakage and thus achieves semi-anonymity. Besides, it also generalizes the file access control to the privilege control, by which privileges of all operations on the cloud data can be managed in a fine-grained manner. Subsequently, we present the *AnonyControl-F* which fully prevents the identity leakage and achieve the full anonymity. Our security analysis shows that both *AnonyControl* and *AnonyControl-F* are secure under the DBDH assumption, and our performance evaluation exhibits the feasibility of our schemes.**

I. Introduction

Cloud computing is a revolutionary computing technique, by which computing resources are provided dynamically via Internet and the data storage and computation are outsourced to someone or some party in a 'cloud'. It greatly attracts attention and interest from both academia and industry due to the profitability, but it also has at least three challenges that must be handled before coming to our real life to the best of our knowledge. First of all, data confidentiality should be guaranteed. The data privacy is not only about the data contents. Since the most attractive part of the cloud computing is the computation outsourcing, it is far beyond enough to just conduct an access control. More likely, users want to control the privileges of data manipulation over other users or cloud servers. This is because when sensitive information or computation is outsourced to the cloud servers or another user, which is out of users' control in most cases, privacy risks would rise dramatically because the servers might illegally inspect users' data and access sensitive information, or other users might be able to infer sensitive information from the outsourced computation. Therefore, not only the access but also the operation should be controlled. Secondly, personal information (defined by each user's attributes set) is at risk because one's identity is authenticated based on his information for the purpose of access control (or privilege control in this paper). As people are becoming more concerned about their identity privacy these days, the identity privacy also needs to be protected before the cloud enters our life. Preferably, any authority or server alone should not know any client's personal information. Last but not least, the cloud computing system should be resilient in the case of security breach in which some part of the system is compromised by attackers.

Various techniques have been proposed to protect the data contents privacy via access control. Identity-based encryption (IBE) was first introduced by Shamir [1], in which the sender of a message can specify an identity such that only a receiver with matching identity can decrypt it. Few years later, Fuzzy Identity-Based Encryption [2] is proposed, which is also known as Attribute-Based Encryption (ABE). In such encryption scheme, an identity is viewed as a set of descriptive attributes, and decryption is possible if a decrypter's identity has some overlaps with the one specified in the ciphertext. Soon after, more general tree-based ABE schemes, Key-Policy Attribute-Based Encryption (KP-ABE) [3] and Ciphertext-Policy Attribute-Based Encryption (CP-ABE) [4], are presented to express more general condition than simple 'overlap'. They are counterparts to each other in the sense that the decision of encryption policy (who can or cannot decrypt the message) is made by different parties.

In the KP-ABE [3], a ciphertext is associated with a set of attributes, and a private key is associated with a monotonic access structure like a tree, which describes this user's identity (e.g. IIT AND (Ph.D OR Master)). A user can decrypt the ciphertext if and only if the access tree in his private key is satisfied by the attributes in the ciphertext. However, the encryption policy is described in the keys, so the encrypter does

*Dr. Xiang-Yang Li (xli@cs.iit.edu) is the contact author of this paper.

[1]The research of Li is partially supported by NSF CNS-1035894, NSF ECCS-1247944, NSF ECCS-1343306, NSF CMMI 1436786, National Natural Science Foundation of China under Grant No. 61170216, No. 61228202. Any opinions, findings, conclusions, or recommendations expressed in this paper are those of author(s) and do not necessarily reflect the views of the funding agencies (NSF, and NSFC).

not have entire control over the encryption policy. He has to trust that the key generators issue keys with correct structures to correct users. Furthermore, when a re-encryption occurs, all of the users in the same system must have their private keys re-issued so as to gain access to the re-encrypted files, and this process causes considerable problems in implementation. On the other hand, those problems and overhead are all solved in the CP-ABE [4]. In the CP-ABE, ciphertexts are created with an access structure, which specifies the encryption policy, and private keys are generated according to users' attributes. A user can decrypt the ciphertext if and only if his attributes in the private key satisfy the access tree specified in the ciphertext. By doing so, the encrypter holds the ultimate authority about the encryption policy. Also, the already issued private keys will never be modified unless the whole system reboots.

Unlike the data confidentiality, less effort is paid to protect users' identity privacy during those interactive protocols. Users' identities, which are described with their attributes, are generally disclosed to key issuers, and the issuers issue private keys according to their attributes. But it seems natural that users are willing to keep their identities secret while they still get their private keys. Therefore, we propose *AnonyControl* and *AnonyControl-F* (Fig. 1) to allow cloud servers to control users' access privileges without knowing their identity information. Their main merits are:

1) The proposed schemes are able to protect user's privacy against each single authority. Partial information is disclosed in *AnonyControl* and no information is disclosed in *AnonyControl-F*.

2) The proposed schemes are tolerant against authority compromise, and compromising of up to $(N-2)$ authorities does not bring the whole system down.

3) We provide detailed analysis on security and performance to show feasibility of the scheme *AnonyControl* and *AnonyControl-F*.

4) We firstly implement the real toolkit of a multi-authority based encryption scheme *AnonyControl* and *AnonyControl-F*.

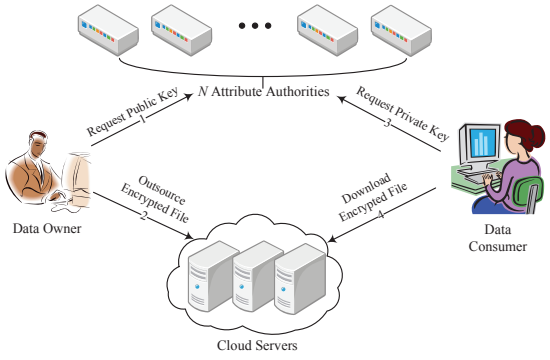


Fig. 1. General flow of our scheme

The rest of the paper is organized as follows. Section II describes related works and Section III introduces preliminary cryptographic backgrounds. Then, Section IV formally defines our problem for the construction of *AnonyControl* in Section V and *AnonyControl-F* in Section VI. Finally, we evaluate the security in Section VII and performance in Section VIII, and conclude in Section IX.

II. Related Work

In [5] [6], a multi-authority system is presented in which each user has an ID and they can interact with each key generator (authority) using different pseudonyms. One user's different pseudonyms are tied to his private key, but key generators never know about the private keys, and thus they are not able to link multiple pseudonyms belonging to the same user. Also, the whole attributes set is divided into N disjoint sets and managed by N attributes authorities. In this setting, each authority knows only a part of any user's attributes, which are not enough to figure out the user's identity. However, the scheme proposed by Chase *et al.* [6] considered the basic threshold-based KP-ABE, which lacks generality in the encryption policy expression. Many attribute based encryption schemes having multiple authorities have been proposed afterwards [7]–[10], but they either also employ a threshold-based ABE [7], or have a semi-honest central authority [8]–[10], or cannot tolerate arbitrarily many users' collusion attack [7].

The work by Lewko *et al.* [11] and Muller *et al.* [12] are the most similar ones to ours in that they also tried to decentralize the central authority in the CP-ABE into multiple ones. Lewko *et al.* use a LSSS matrix as an access structure, but their scheme only converts the AND, OR gates to the LSSS matrix, which limits their encryption policy to boolean formula, while we inherit the flexibility of the access tree having threshold gates. Muller *et al.* also supports only Disjunctive Normal Form (DNF) in their encryption policy. Besides the fact that we can express arbitrarily general encryption policy, our system also tolerates the compromise attack towards attributes authorities, which is not covered in many existing works.

Recently, there also appeared traceable multi-authority ABE [13] [14], which are on the opposite direction of ours. Those schemes introduce accountability such that malicious users' keys can be traced. On the other hand, similar direction as ours can be found in [15]–[17], who try to hide encryption policy in the ciphertexts, but their solutions do not prevent the attribute disclosure in the key generation phase. To some extent, these three works and ours complement each other in the sense that the combination of these two types protection will lead to a completely anonymous ABE.

III. Preliminaries

Let $\mathbb{G}_0$ be a multiplicative cyclic group of prime order p and g be its generator. The bilinear map e ([18] [19]) is defined as follows: $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$, where $\mathbb{G}_T$ is the codomain of e. The bilinear map e has the following properties: $\forall u, v \in \mathbb{G}_0$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$ (bilinearity); for all $u, v \in \mathbb{G}_0$, $e(u, v) = e(v, u)$ (symmetry); and $e(g, g) \neq 1$ (non-degeneracy).

Definition 1. *The Decisional Bilinear Diffie-Hellman (DBDH) problem in group $\mathbb{G}_0$ of prime order p with generator g is defined as follows: on input $g, g^a, g^b, g^c \in \mathbb{G}_0$ and $e(g, g)^z \in \mathbb{G}_T$, where $a, b, c \in \mathbb{Z}_p$, decide whether $e(g, g)^z = e(g, g)^{abc}$.*

The security of many ABE schemes ([4], [20]–[23]) and ours rely on the assumption that no probabilistic polynomial-time algorithms can solve the DDH or DBDH problem with

non-negligible advantage (DDH assumption and DBDH assumption). This assumption is reasonable since discrete logarithm problems in large number field are widely considered to be intractable [24]–[28], and the groups we chose are cyclic multiplicative groups of prime order, in which DBDH problems are believed to be hard.

We introduce the Lagrange coefficient $\triangle_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$: $\triangle_{i,S}(x) := \prod_{j \in S, j \neq i} \frac{x-j}{i-j}$, which will be used in the polynomial interpolation in the decryption algorithm. Additionally, a one-way hash function $H : \{0,1\}^* \rightarrow \mathbb{G}_0$ is defined as a random oracle, which maps any attribute value to a random element in $\mathbb{Z}_p$.

A. Privilege Trees T_p

In our work, encryption policy is described with a tree called access tree. Each non-leaf node of the tree is a threshold gate, and each leaf node is described by an attribute. One access tree is required in every data file to define the encryption policy. In this paper, we extend existing schemes by generalizing the access tree to a privilege tree. The privilege in our scheme is defined as similar to the privileges managed in ordinary operating systems. A data file has several operations executable on itself, and each of them is allowed only to authorized users with different level of qualifications. For example, {*Read_mine, Read_all, Delete, Modify, Create*} is a privileges set of students' grades. Then, reading Alice's grades is allowed to her and her professors, but all other privileges should be authorized only to the professors, so we need to grant the "*Read_mine*" to Alice and all other to the professors.

Every operation is associated with one privilege p, which is described by a privilege tree T_p. If a user's attributes satisfy T_p, he is granted the privilege p. By doing so, we not only control the file access but also control other executable operations, which makes the file controlling fine-grained and thus suitable for cloud storage service.

In our scheme, several trees are required in every data file to verify users' identity and to grant him a privilege accordingly. There are supposed to be r these kind of structures, which means there are r different privileges defined for the corresponding data file. The privilege 0 is defined as the privilege to read the file, and other privileges may be defined arbitrarily (the m-th privilege does not necessarily have more powerful privilege than the n-th one when $m > n$). The tree is similar to the one defined in [4]. Given a tree, if num_x is the number of the node x's children node and k_x is its threshold value $0 < k_x \leq num_x$, then node x is assigned a true value if at least k_x children nodes have been assigned true value. Specially, the node becomes an OR gate when $k_x = 1$ and an AND gate when $k_x = num_x$.

B. Satisfying the Privilege Tree

If a user's attributes set S satisfies the privilege tree T_p or the node x, we define it as $T_p(S) = 1$ or $x(S) = 1$ respectively. $T_p(S)$ is calculated recursively as follows. If x is a leaf node, $x(S) = 1$ if and only if $att(x) \in S$. If x is a non-leaf node, $x(S) = 1$ only when at least k_x child nodes return 1. For the root node R_p of T_p, $T_p(S) = 1$ only if $R_p(S) = 1$.

IV. Problem Formulation

A. System Model

In our system, there are four types of entities: N *Attribute Authorities* (denoted as $\mathcal{A}$), *Cloud Server*, *Data Owners* and *Data Consumers*. A user can be a Data Owner and a Data Consumer simultaneously.

Authorities are assumed to have powerful computation abilities, and they are supervised by government offices because some attributes partially contain users' personally identifiable information. The whole attribute set is divided into N disjoint sets and controlled by each authority, therefore each authority is aware of only part of attributes.

A Data Owner is the entity who wishes to outsource encrypted data file to the Cloud Servers. The Cloud Server, who is assumed to have adequate storage capacity, does nothing but store them.

Newly joined Data Consumers request private keys from all of the authorities, and they do not know which attributes are controlled by which authorities. When the Data Consumers request their private keys from the authorities, authorities jointly create corresponding private key and send it to them.

All Data Consumers are able to download any of the encrypted data files, but only those whose private keys satisfy the privilege tree T_p can execute the operation associated with privilege p. The server is delegated to execute an operation p if and only if the user's credentials are verified through the privilege tree T_p.

B. Threats Model

We assume the Cloud Servers are semi-honest, who behave properly in most of time but may collude with malicious Data Consumers or Data Owners to harvest others' file contents to gain illegal profits. But they are also assumed to gain legal benefit when users' requests are correctly processed, which means they will follow the protocol in general.

N authorities are assumed to be untrusted. That is, they will follow our proposed protocol in general, but try to find out as much information as possible individually. More specifically, we assume they are interested in users' attributes to achieve the identities, but they will not collude with users or other authorities. This assumption is similar to many previous researches on security issue in cloud computing (e.g. [20], [29]–[31]), and it is also reasonable since these authorities will be audited by government offices. However, we will further relax this assumption and allow the collusion between the authorities in Section VI.

Data Consumers are untrusted since they are random users including attackers. They may collude with other Data Consumers to illegally access what they are not allowed to.

Besides, we do not consider the identity leakage from the underlying network since this can be trivially prevented by employing anonymized network protocols (e.g., [32], [33]).

C. Security Model

To formally define the security of our *AnonyControl*, we first give the following definitions.

Setup $\rightarrow \mathbf{PK}, \mathbf{MK}_k$:

This algorithm takes nothing as input except implicit inputs such as security parameters. Attributes authorities execute this algorithm to jointly compute a system-wide public parameter $\mathbf{PK}$ as well as an authority-wide public parameter y_k, and to individually compute a master key $\mathbf{MK}_k$.

KeyGenerate($\mathbf{PK}$, $\mathbf{MK_k}$, $\mathbb{A}^u$) $\rightarrow \mathbf{SK}_u$:

This algorithm enables a user to interact with every attribute authority, and obtains a private key $\mathbf{SK}_u$ corresponding to the input attribute set $\mathbb{A}^u$.

Encrypt($\mathbf{PK}$, M, $\{T_p\}_{p\in\{0,\cdots,r-1\}}$) $\rightarrow (\mathbf{CT}, \mathbf{VR})$:

This algorithm takes as input the public key $\mathbf{PK}$, a message M, and a set of privilege trees $\{T_p\}_{p\in\{0,\cdots,r-1\}}$, where r is determined by the encrypter. It will encrypt the message M and returns a ciphertext $\mathbf{CT}$ and a verification set $\mathbf{VR}$ so that a user can execute specific operation on the ciphertext if and only if his attributes satisfy the corresponding privilege tree T_p. As we defined, T_0 stands for the privilege to read the file.

Decrypt($\mathbf{PK}$, $\mathbf{SK}_u$, $\mathbf{CT}$) $\rightarrow M$ or verification parameter:

This algorithm will be used at file controlling (e.g. reading, modification, deletion). It takes as input the public key $\mathbf{PK}$, a ciphertext $\mathbf{CT}$, and a private key $\mathbf{SK}_u$, which has a set of attributes $\mathbb{A}^u$ and corresponds to its holder's $\mathbf{GID}_u$. If the set $\mathbb{A}^u$ satisfies any tree in the set $\{T_p\}_{p\in\{0,\cdots,r-1\}}$, the algorithm returns a message M or a verification parameter. If the verification parameter is successfully verified by Cloud Servers, who use $\mathbf{VR}$ to verify it, the operation request will be processed.

Next, we define the security of our *AnonyControl* with the following game.

Init The adversary $\mathcal{A}$ declares the set of compromised authorities $\{\mathcal{A}_k\} \subset \mathbf{A}$ (where at least two authorities in $\mathbf{A}$ are not controlled by $\mathcal{A}$) that are under his control (remaining authorities $\mathbf{A}/\{\mathcal{A}_k\}$ are controlled by the challenger). Then, he declares T_0 that he wants to be challenged, in which some attributes are being in charged by the challenger's authorities.

Setup* The challenger and the adversary jointly run the Setup algorithm to receive the valid outputs.

Phase 1 The adversary launches KeyGenerate algorithms to query for as many private keys as he wants, which correspond to attribute sets $\mathbb{A}_1, \cdots, \mathbb{A}_q$ being disjointly in charged by all authorities $\{\mathcal{A}_k\}$, but none of these keys satisfy T_0. Besides, he also conducts arbitrarily many computations using the public and secret keys that he has (belonging to compromised authorities).

Challenge The adversary submits two messages M_0 and M_1 of equal size to the challenger. The challenger flips a random binary coin b and encrypts M_b with T_0. The ciphertext $\mathbf{CT}$ is given to the adversary.

Phase 2 Phase 1 is repeated adaptively, but none of the queried keys satisfy T_0.

Guess The adversary outputs a guess b' of b.

The advantage of an adversary $\mathcal{A}$ in this game is defined as $\Pr[b' = b] - \frac{1}{2}$.

Definition 2. *Our scheme is secure and indistinguishable against chosen-attribute attack (IND-CAA) if all probabilistic polynomial-time adversaries (PPTA) have at most a negligible advantage in the above game.*

Note that the IND-CAA defined above implies IND-CCA since the adversary can conduct encryptions and decryptions using the public keys and secret keys it owns in Phase 1 and Phase 2 (but he cannot decrypt the target ciphertext since none of its secret keys satisfy T_0).

D. Design Goals

Our goal is to achieve a multi-authority CP-ABE which: achieves the security defined above; guarantees the confidentiality of Data Consumers' identity information; and tolerates compromise attacks on the authorities or the collusion attacks by the authorities.

For the visual comfort, we frequently use the following notations hereafter. $\mathcal{A}_k$ denotes the k-th attribute authority; $\mathbb{A}^u$ denotes the attributes set of user u; $\mathbb{A}^u_k$ denotes the subset of $\mathbb{A}^u$ controlled by $\mathcal{A}_k$; and $\mathbb{A}^{T_p}$ denotes the attributes set included in tree T_p.

V. *AnonyControl* Construction

A. Setup

At the system initialization phase, any one of the authorities chooses a bilinear group $\mathbb{G}_0$ of prime order p with generator g and publishes it. Then, all authorities independently and randomly picks $v_k \in \mathbb{Z}_p$ and send $Y_k = e(g,g)^{v_k}$ to all other authorities who individually compute $Y := \prod_{k\in\mathcal{A}} Y_k = e(g,g)^{\sum_{k\in\mathcal{A}} v_k}$.

Then, every authority $\mathcal{A}_k$ randomly picks $N-1$ integers $s_{kj} \in \mathbb{Z}_p (j \in \{1,\cdots,N\}\backslash\{k\})$ and computes $g^{s_{kj}}$. Each $g^{s_{kj}}$ is shared with each other authority $\mathcal{A}_j$ An authority $\mathcal{A}_k$, after receiving $N-1$ pieces of $g^{s_{jk}}$ generated by $\mathcal{A}_j$, computes its secret parameter $x_k \in \mathbb{Z}_p$ as follows:

$$\begin{aligned} x_k &= \Big(\prod_{j\in\{1,\cdots,N\}\backslash\{k\}} g^{s_{kj}}\Big) \Big/ \Big(\prod_{j\in\{1,\cdots,N\}\backslash\{k\}} g^{s_{jk}}\Big) \\ &= g^{\left(\sum_{j\in\{1,\cdots,N\}\backslash\{k\}} s_{kj} - \sum_{j\in\{1,\cdots,N\}\backslash\{k\}} s_{jk}\right)} \end{aligned}$$

It is easy to see that these randomly produced integers satisfy $\prod_{k\in\mathcal{A}} x_k = 1 \mod p$. This is an important property which achieves compromise attack tolerance for our scheme, which will be discussed in the next section.

Then, the master key for the authority $\mathcal{A}_k$ is $\mathbf{MK}_k = \{v_k, x_k\}$, and public key of the whole system is published as $\mathbf{PK} = \{\mathbb{G}_0, g, Y = e(g,g)^{\sum v_k}\}$.

Note that the time complexity of the setup computation is $O(N^2)$ since every authority computes $N-1$ pieces of $g^{s_{kj}}$. However, this can be further reduced to $O(N)$ by applying the following simple trick. We first cluster the authorities into C clusters, and exchanges the parameters within the cluster only. Then, the time complexity is reduced to $O(CN) = O(N)$ since C is a constant.

B. KeyGenerate($\mathbf{PK}$, $\mathbf{MK_k}$, $\mathbb{A}^u$)

When a new user u with $\mathbf{GID}_u$ wants to join the system, he requests the private key from all of the authorities by following this process which is composed of two phases.

Attribute Key Generation: For any attribute $i \in \mathbb{A}^u$, every $\mathcal{A}_k$ randomly picks $r_i \in \mathbb{Z}_p$ to individually compute the partial private keys $H(att(i))^{r_i}, D'_i = g^{r_i}$,which are privately sent to the user u.

Then, each authority $\mathcal{A}_k$ randomly picks $d_k \in \mathbb{Z}_p$, computes $x_k \cdot g^{v_k} \cdot g^{d_k}$ and privately shares it with other authorities (*i.e.* kept secret to the user u). Then, he privately sends $x_k \cdot g^{d_k}$ to the user u (*i.e.* kept secret to other authorities).

Any one of N authorities computes and sends the following term to the user u:

$$D = \prod x_k g^{v_k} g^{d_k} = g^{\sum v_k + \sum d_k}$$

where $g^{\sum v_k}$ acts as a system-wide master key used to generate a valid secret key, but no single authority is able to infer its value. A valid D with a valid $g^{\sum v_k}$ can be achieved only when all the authorities correctly follow the protocol and conduct a joint computation.

Then, the user computes the following term which is the attribute key for the attribute i ($att(i)$ refers to the element in $\mathbb{G}_0$ corresponding to i):

$$D_i = H(att(i))^{r_i} \cdot \prod (x_k \cdot g^{d_k}) = H(att(i))^{r_i} \cdot g^{(\sum d_k)}$$

Note that D_i is computed securely without disclosing individual g^{d_k}'s to the user or disclosing $g^{\sum d_k}$ to any attribute authority. This is very important in the tolerance to the compromise attack, which will be discussed later.

Key Aggregation: User u, after receiving D, D_i's and D'_i's, aggregates the components as his private key:

$$\mathbf{SK}_u = \{D, \forall i \in \mathbb{A}^u : D_i = g^{(\sum d_k)} \cdot H(att(i))^{r_i}, D'_i = g^{r_i}\}$$

C. Encrypt($\mathbf{PK}$, M, $\{T_p\}_{p\in\{0,\cdots,r-1\}}$)

The Data Owner encrypts the data with any existing symmetric encryption scheme, and generates the decryption key K_e. Then, he determines a set of privilege trees $\{T_p\}_{p\in\{0,\cdots,r-1\}}$ and executes $Encrypt(\mathbf{PK}, K_e, \{T_p\})$.

Remember that the privilege tree in our scheme is based on the threshold gates. Here, Shamir's secret sharing technique [34] is directly used to implement the threshold gate. Shamir's t-out of-n secret share scheme allows one to divide a secret to n shares, and the original secret can be recovered with t of them. So, in our tree, the node value of the gate is recovered if and only if at least k_x values of children nodes are recovered in recursive manner. The random number, which is used to mask the decryption key K_e, is stored at the root of the privilege tree and is secret-shared to its children nodes, and the secret shares in the children nodes are secret-shared to their children nodes, so and so forth until the recursive secret sharing reaches the leaf nodes.

This is implemented in the following way. For each T_p, the algorithm first chooses a polynomial q_x for each node x in it. For each node x, sets the degree d_x of the polynomial q_x as one less than the threshold value k_x. Starting from the root node R_p, the algorithm randomly picks $s_p \in \mathbb{Z}_p$ and sets $q_{R_p}(0) := s_p$ and randomly chooses other coefficients for q_{R_p}. Then, for any other node x, the coefficients are chosen randomly and the constant term is set as $q_{parent(x)}(index(x))$ such that $q_x(0) = q_{parent(x)}(index(x))$ ($index(x)$ is the index of the x's child nodes, and $parent(x)$ is node x's parent node). Finally, he picks a random element $h \in \mathbb{Z}_p$ such that $h^{-1} \bmod p$ exists, and calculates $g^{h \cdot s_p}, D^{h^{-1}}$, and the ciphertext $\mathbf{CT}$ is created as

$$\mathbf{CT} = \langle \{T_p\}_{p\in\{0,\cdots,r-1\}}, E_0 = K_e \cdot Y^{s_0}, C = g^{hs_p}, \hat{C} = D^{h^{-1}} \\ \{C_i = g^{q_i(0)}, C'_i = H(att(i))^{q_i(0)}\}_{i\in\mathbb{A}^{T_p}, \forall p\in\{0,\cdots,r-1\}} \rangle$$

Note that $D^{h^{-1}}$ is introduced to prevent key combination attack, which is similar to the idea appeared in [4], but in different ways: they introduced such a inverse in the power in key generation algorithm while we does so in the encryption in order to achieve the de-centralization.

Then, $\mathbf{VR}$, which is disclosed only to the Cloud Server, is created for the purpose of privilege verification.

$$\mathbf{VR} = \langle \{E_p = Y^{s_p}\}_{p\in\{1,\cdots,r-1\}} \rangle$$

Finally, Data Owner sends $\mathbf{CT}$, $\mathbf{VR}$ and the encrypted file to the Cloud Server to share them with other Data Consumers.

D. Decrypt($\mathbf{PK}$, $\mathbf{SK}_u$, $\mathbf{CT}$)

Every user within the system can download the ciphertext from the Cloud Server, but he is able to execute operations on encrypted data only after he successfully decrypts it. Firstly, we define a recursive algorithm $DecryptNode(\mathbf{CT}, \mathbf{SK}_u, x)$, where x stands for a node in the privilege tree T_p. If the node x is a leaf node, we let i be the attribute of the node x and define as follows. If $i \in \mathbb{A}^u$,

$$DecryptNode(\mathbf{CT}, \mathbf{SK}_u, x) = \frac{e(D_i, C_x)}{e(D'_i, C'_x)} \\ = \frac{e(g^{\sum d_k} \cdot H(att(i))^{r_i}, g^{q_x(0)})}{e(g^{r_i}, H(att(i))^{q_x(0)})} = e(g,g)^{(\sum d_k)\cdot q_x(0)}$$

If not, we define $DecryptNode(\mathbf{CT}, \mathbf{SK}_u, x) := \perp$. If x is not a leaf node, the algorithm proceeds as follows: For all nodes z that are children of x, it calls $DecryptNode(\mathbf{CT}, \mathbf{SK}_u, z)$ and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z such that $F_z \neq \varnothing$. If no such set exists then the node was not satisfied and the algorithm returns $\perp$. Otherwise, compute

$$F_x = \prod_{z\in S_z} F_z^{\triangle_{d,s'_x}(0)}, \text{ where } \begin{cases} d = index(z) \\ S'_x = index(z) : z \in S_x \end{cases} \\ = \prod_{z\in S_z} (e(g,g)^{(\sum d_k)\cdot q_z(0)})^{\triangle_{d,S'_x}(0)} \\ = \prod_{z\in S_z} (e(g,g)^{(\sum d_k)\cdot q_{parent(z)}(d)})^{\triangle_{d,S'_x}(0)} \\ = \prod_{z\in S_z} (e(g,g)^{(\sum d_k)\cdot q_x(d)})^{\triangle_{d,S'_x}(0)} \\ = e(g,g)^{(\sum d_k)\cdot q_x(0)} \text{ (using polynomial interpolation)}$$

The interpolation above recovers the parent node's value by calculating coefficients of the polynomial and evaluating the $p(0)$. We direct the readers to [34] for complete calculation. A user recursively calls this algorithm, starting from the root node R_p of the tree T_p, after downloading the file. If the tree is satisfied, which means he is granted the privilege p, then

$$DecryptNode(\mathbf{CT}, \mathbf{SK}_u, R_p) = e(g,g)^{s_p \sum d_k}$$

Finally, if the user is trying to read the file, the decryption key K_e can be recovered by:

$$\frac{E_0}{\frac{e(C,\hat{C})}{e(g,g)^{s_0 \sum d_k}}} = \frac{K_e \cdot Y^{s_0}}{\frac{e(g,g)^{s_0(\sum d_k + \sum v_k)}}{e(g,g)^{s_0 \sum d_k}}} = K_e$$

Then, the data file can be decrypted by using it. Otherwise, if he wants to execute some operation on the data, he should be verified as an authorized user for the execution first. If the execution requires the j-th privilege, the user recursively calls $Decrypt(\mathbf{CT}, \mathbf{SK}_u, x)$ starting from the root node R_j of the tree T_j to get $e(g,g)^{s_j \sum d_k}$ and further achieve Y^{s_j} with the same equation as above. The user sends it to the Cloud Server as well as the operation request. The Cloud Server checks whether $Y^{s_j} = E_j$, and proceeds if they do equal each other. In fact, Y^{s_j} should be encrypted to avoid replay attack. This can be simply implemented by introducing any public key encryption protocol.

VI. Achieving Full Anonymity

We have assumed semi-honest authorities in *AnonyControl* and we assumed that they will not collude with each other. This is a necessary assumption in *AnonyControl* because each authority is in charge of a subset of the whole attributes set, and for the attributes that it is in charge of, it knows the exact information of the key requester. If the information from all authorities is gathered altogether, the complete attribute set of the key requester is recovered and thus his identity is disclosed to the authorities. In this sense, *AnonyControl* is semi-anonymous since partial identity information (represented as some attributes) is disclosed to each authority, but we can achieve a full-anonymity and also allow the collusion of the authorities.

The key point of the identity information leakage we had in our previous scheme as well as every existing attribute based encryption schemes is that key generator (or attribute authorities in our scheme) issues attribute key based on the reported attribute, and the generator has to know the user's attribute to do so. We need to introduce a new technique to let key generators issue the correct attribute key without knowing what attributes the users have. A naive solution is to give all the attribute keys of all the attributes to the key requester and let him pick whatever he wants. In this way, the key generator does not know which attribute keys the key requester picked, but we have to fully trust the key requester that he will not pick any attribute key not allowed to him. To solve this, we leverage the following Oblivious Transfer (OT).

A. 1-out-of-n Oblivious Transfer

In an 1-out-of-n OT, the sender Bob has n messages $M_1, \cdots, M_n$, and the receiver Alice wants to pick one M_i from those $M_1, \cdots, M_n$. Alice successfully achieves M_i without knowing any useful information about other messages, and Bob does not know which M_i is picked by Alice. We employ [35] as a building block out of many implementations [35]–[37], in our fully anonymous multi-authority CP-ABE in the next section.

Algorithm 1 1-out-of-2 Oblivious Transfer

1: Bob randomly picks a secret s and publishes g^s to Alice.
2: Alice creates an encryption/decryption key pair:$\{g^r, r\}$
3: Alice chooses i and calculates $EK_i = g^r$, $EK_{i-1} = \frac{g^s}{g^r}$ and sends EK_0 to Bob.
4: Bob calculates $EK_1 = \frac{g^s}{EK_0}$ and encrypts M_0 using EK_0 and M_1 using EK_1 and sends two cipher texts $E_{EK_0}(M_0), E_{EK_1}(M_1)$ to Alice.
5: Alice can use r to decrypt the desired cipher text $E_{EK_i}(M_i)$, but she cannot decrypt the other one. Meanwhile, Bob does not know which cipher text is decrypted.

We use the 1-out-of-2 OT (Algorithm 1), in which Alice picks M_i from Bob's M_0, M_1, to introduce the 1-out-of-n OT described in Algorithm 2.

Algorithm 2 1-out-of-n Oblivious Transfer

1: Bob randomly picks n secrets $s_1, \cdots, s_n$ and calculates t_i as follows:

$$\forall i \in \{1, \cdots, n\} : t_i = s_1 \oplus \cdots \oplus s_{i-1} \oplus M_i$$

2: For each $i \in \{1, \cdots, n\}$, Bob and Alice are engaged in a 1-out-of-2 OT where Bob's first message is t_i and the second message is s_i. Alice picks t_i to receive if she wants M_i and s_i otherwise.
3: After Alice receives n components, she has $t_i = s_1 \oplus \cdots \oplus s_{i-1} \oplus M_i$ for the i she wants and s_k for $k \neq i$, she can recover the M_i by

$$M_i = t_i \oplus s_{i-1} \oplus s_{i-2} \oplus \cdots \oplus s_1$$

In Algorithm 2, Alice can achieve M_i if and only if she picks t_i for the i she wants the message and s_k for any $k \neq i$. If she picks several t_k's, some s_k's are missing and she is not able to recover any message.

B. Fully Anonymous Multi-Authority CP-ABE

In this section, we present how to achieve the full anonymity in *AnonyControl* to designs the fully anonymous privilege control scheme *AnonyControl-F*.

The KeyGenerate algorithm is the only part which leaks identity information to each attribute authority. Upon receiving the attribute key request with the attribute value, the attribute authority will generate $H(att(i))^{r_i}$ and sends it to the requester where $att(i)$ is the attribute value and r_i is a random

number for that attribute. The attribute value is disclosed to the authority in this step.

We can introduce the above 1-out-of-n OT to prevent this leakage. We let each authority be in charge of all attributes belonging to the same category. For each attribute category c (e.g., University), suppose there are k possible attribute values (e.g., IIT, NYU, CMU ...), then one requester has at most one attribute value in one category. Upon the key request, the attribute authority can pick a random number r_u for the requester and generates $H(att(i))^{r_u}$ for all $i \in \{1, \cdots, k\}$. After the attribute keys are ready, the attribute authority and the key requester are engaged in a 1-out-of-k OT where the key requester wants to receive one attribute key among k.

By introducing the 1-out-of-k OT in our KeyGenerate algorithm, the key requester achieves the correct attribute key that he wants, but the attribute authority does not have any useful information about what attribute is achieved by the requester. Then, the key requester achieves the full anonymity in our scheme and no matter how many attribute authorities collude, his identity information is kept secret.

C. Discussions

Trustfulness of Users: Our *AnonyControl-F* also needs to trust the requester that he picks correct attribute keys corresponding to his identity, but the requester can pick only one attribute key in one category, which is much better than the naive idea above, and it is not this paper's scope to guarantee the truthful reporting of the attributes. To the best of our knowledge, it is assumed that some other authentication (e.g., government check) is in place to verify the reported attributes in most of ABE-related works.

Performance: The extra computation introduced in *AnonyControl-F* is just several exponent calculations, which are negligible. However, extra communication overhead is a problematic issue in *AnonyControl-F*. For each attribute category, the user is involved in a 1-out-of-n OT which needs $O(n)$ rounds of communication. Therefore, the communication overhead grows from $O(1)$ in *AnonyControl* to $O(I)$ where I is the size of the entire attribute set. This is the main drawback of our fully anonymous scheme, which should be solved in our future work.

VII. Security Analysis

A. Tolerance Against Authorities' Collusion or Compromise Attack

In the proposed scheme, an authority $\mathcal{A}_k$ generates a set of random secret parameters $\{s_{kj}\}$ and shares $g^{s_{kj}}$ it with other authorities via secure channel, and x_k is computed based on this parameters. It is believed that DDH problem is intractable in the group $\mathbb{G}_0$ of prime order p, therefore $g^{s_{kj}}$ does not leak any statistical information about s_{kj}. This implies even if an adversary is able to compromise up to $(N-2)$ authorities, there are still two parameters s_{kj} kept unknown to the adversary. So, the adversary is not able to guess the valid $g^{\sum v_k}$, and he fails to construct a valid secret key. Hence, the scheme achieves compromise tolerance to up to $(N-2)$ authorities compromise.

But, if we reduce the time complexity of the setup phase by dividing authorities into several clusters having C authorities in each, attackers can compromise $C-1$ authorities in a cluster to create valid master keys of that cluster. Therefore, there is a trade-off between tolerance and complexity. However, since the number of authorities is typically not very huge, and the setup is one-time operation at the very beginning of the system setup, we recommend using the original setup algorithm whose complexity is $O(N^2)$.

Note that the compromised authorities are able to issue valid attribute keys for which they are in charge of, so the ciphertexts whose privilege trees have only those attributes might be illegally decrypted if the attacker issue all possible attribute keys to himself. But, since the authorities are well protected servers, it is hard to compromise even one authority, and the probability of compromising enough authorities to illegally decrypt some ciphertext is very low.

B. Tolerance Against Users' Collusion Attack

In order to access a plaintext, attackers must recover $Y^{s_0} = e(g,g)^{s_0 \sum v_k}$, which can be recovered only if the attackers have enough attributes to satisfy the tree T_0. When two different keys' components are combined, the combined key cannot go through the polynomial interpolation in the decryption algorithm due to the different randomizers in each key. Therefore, at least one key should be valid to satisfy a privilege tree.

C. Formal Proof

With aforementioned properties (indistinguishability of s_{kj}'s and inability of interpolation using different users' keys), we are ready to formally prove that *AnonyControl* and *AnonyControl-F* are both secure. To be granted the file access privilege ($T_p = T_0$), one needs to recover Y^{s_0} from $E_0 = K_e \cdot Y^{s_0}$, while one needs to recover Y^{s_p} if he needs other privileges. They are basically the same parameters with different values, therefore it is enough to prove that no polynomial time adversaries have significant advantage in our security game (Section IV, defined only for the file access privilege) to show the security of our schemes instead of proving it for all privileges.

Theorem VII.1. *If an adversary has a non-negligible advantage in our security game (Section IV), there exists at least one probabilistic polynomial-time algorithm who can solve the DBDH problem (Section III) with a non-negligible advantage.*

Proof: Suppose a probabilistic polynomial-time adversary's advantage in our security game is ϵ. We prove that the following DBDH game can be solved with an advantage $\frac{\epsilon}{2}$.

Let $e : \mathbb{G}_0 \times \mathbb{G}_0 \to \mathbb{G}_T$ be a bilinear map, where $\mathbb{G}_0$ is a multiplicative cyclic group of prime order p and g is its generator. First the DBDH challenger flips a binary coin μ, and he sets $(g, A, B, C, Z) := (g, g^a, g^b, g^c, e(g,g)^{abc})$ if $\mu = 0$; otherwise he sets $(g, A, B, C, Z) := (g, g^a, g^b, g^c, e(g,g)^z)$, where $a, b, c, z \in \mathbb{Z}_p$ are randomly picked. The challenger then gives the simulator $\langle g, A, B, C, Z\rangle = \langle g, g^a, g^b, g^c, Z\rangle$. The simulator sim then plays the role of a challenger in the following DBDH game.

Init The adversary $\mathcal{A}$ controls the set of compromised authorities $\{\mathcal{A}_k\} \subset \mathbf{A}$ (where at least two authorities in $\mathbf{A}$ are not controlled by $\mathcal{A}$), and remaining authorities $\mathbf{A}/\{\mathcal{A}_k\}$ are controlled by sim. Then, he declares a T_0 which he wants to be challenged, in which some attributes are being in charged by the simulator's authorities $\mathbf{A}/\{\mathcal{A}_k\}$ (*i.e.,* non-compromised authorities).

Setup sim sets $a = \sum d_k, b = \frac{\sum v_k}{\sum d_k}, c = s_0$, where $d_1, \cdots, d_n, v_1, \cdots, v_n, s_0 \in \mathbb{Z}_p$ are all randomly chosen. Meanwhile, he sets the parameter $Y := e(A, B) = e(g, g)^{ab}$ and gives this public parameter to $\mathcal{A}$.

Phase 1 $\mathcal{A}$ queries for as many private keys as he wants, which correspond to the attributes sets $\mathbb{A}_1, \cdots, \mathbb{A}_q$ being disjointly in charged by all authorities $\{\mathcal{A}_k\}$, but none of them satisfy the T_0. sim, after receiving the key queries, computes the components in private keys to respond the $\mathcal{A}$'s requests. For all attributes $i \in \mathbb{A}^u$, he randomly picks $r_i \in \mathbb{Z}_p$, and computes $D_i := A \cdot H(att(i))^{r_i}, D_i' := g^{r_i}$. Then, sim returns the created private keys to $\mathcal{A}$.

Challenge The adversary $\mathcal{A}$ submits two challenge messages m_0 and m_1 to the challenger. The challenger flips a binary coin γ, and returns the following ciphertext to $\mathcal{A}$.

$$\mathbf{CT}^* = \langle T_0, E_0 = m_\gamma \cdot Z, \{C_i = g^{q_i(0)}, C_i' = H(att(i))^{q_i(0)}\}_{i \in \mathbb{A}^{T_0}} \rangle$$

If $\mu = 0$, $Z = e(g, g)^{abc}$. Note that $a = \sum d_k, ab = \sum v_k$ and $c = s_0$, and we have $Z = e(g, g)^{abc} = (e(g, g)^{ab})^c = Y^{s_0}$ and $D_i = g^{\sum d_k} H(att(i))^{r_i}$. Therefore, $\mathbf{CT}^*$ is a valid ciphertext of the message m_γ, and D_i is a valid component of the private key. Otherwise, if $\mu = 1$, $Z = e(g, g)^z$. Then, we have $E_0 = m_\gamma \cdot e(g, g)^z$. Since $z \in \mathbb{Z}_p$ is a random element, E_0 is a random element in $\mathbb{G}_T$ from $\mathcal{A}$'s perspective (if DBDH is hard in the prime order group $\mathbb{G}_T$), therefore $\mathbf{CT}^*$ contains no information about m_γ.

Phase 2 Repeat Phase 1 adaptively.

Guess $\mathcal{A}$ submits a guess γ' of γ. If $\gamma' = \gamma$, sim outputs $\mu' = 0$, indicating that it was given a valid DBDH-tuple (g, A, B, C, Y^{s_0}), otherwise it outputs $\mu' = 1$, indicating that he was given a random 5-element tuple (g, A, B, C, Z).

As shown in the construction of the game, the simulator sim computes the public parameter and the private key in the same way as our scheme. When $\mu = 1$, the adversary $\mathcal{A}$ learns no information about γ, so we have $\Pr[\gamma \neq \gamma' | \mu = 1]$ =$\Pr[\gamma = \gamma' | \mu = 1] = \frac{1}{2}$. Since the simulator sim outputs his guess $\mu' = 1$ when $\gamma \neq \gamma'$, we have $\Pr[\mu' = \mu | \mu = 1]$ =$\Pr[\gamma \neq \gamma' | \mu = 1] = \frac{1}{2}$. If $\mu = 0$, the adversary $\mathcal{A}$ gets a valid ciphertext of m_γ. $\mathcal{A}$'s advantage in this situation is ϵ by definition, so we have $\Pr[\gamma = \gamma' | \mu = 0] = \frac{1}{2} + \epsilon$. Since the simulator gives his guess $\mu' = 0$ when $\gamma = \gamma'$, we have $\Pr[\mu' = \mu | \mu = 0]$ =$\Pr[\gamma = \gamma' | \mu = 0] = \frac{1}{2} + \epsilon$. The overall advantage in this DBDH game is:

$$\Pr[\mu' = \mu | \mu = 0]\Pr[\mu = 0] + \Pr[\mu' = \mu | \mu = 1]\Pr[\mu = 1] - \frac{1}{2} = \frac{1}{2} \cdot (\frac{1}{2} + \epsilon) + \frac{1}{2} \cdot \frac{1}{2} - \frac{1}{2} = \frac{\epsilon}{2}$$

To conclude, the advantage for a PPTA in the DBDH game is $\frac{\epsilon}{2}$ if the advantage for a polynomial-time adversary in our security game is ϵ. Therefore, if an adversary has a non-negligible advantage ϵ in our security game, he has a non-negligible advantage to solve the DBDH problem. ■

Based on the assumption that no PPTA can solve the DBDH problem with non-negligible advantage, it can be deduced that no adversary has significant advantage in our security game. Therefore, our *AnonyControl* is secure according to the definition in Section IV.

Security of AnonyControl-F: The only difference between *AnonyControl* and *AnonyControl-F* is the newly introduced 1-out-of-n OT during the KeyGenerate algorithm. Therefore, as long as the introduced OT does not leak information about the attributes that are transferred via it, *AnonyControl-F* leaks as much as information as *AnonyControl* does. Since the 1-out-of-n OT used in our work is proved to be secure [35], *AnonyControl-F* is as secure as *AnonyControl*.

VIII. Performance Evaluation

In this section, we present the performance evaluation based on our measurement on the implemented prototype system of *AnonyControl-F*. To the best of our knowledge, this is the first implementation of a multi-authority attribute based encryption scheme. Our prototype system provides five command line tools.

anonycontrol-setup : Jointly generates a public key and N master keys.

anonycontrol-keygen : Generates a part of private key for the attribute set it is responsible for.

anonycontrol-enc : Encrypts a file under r privilege trees.

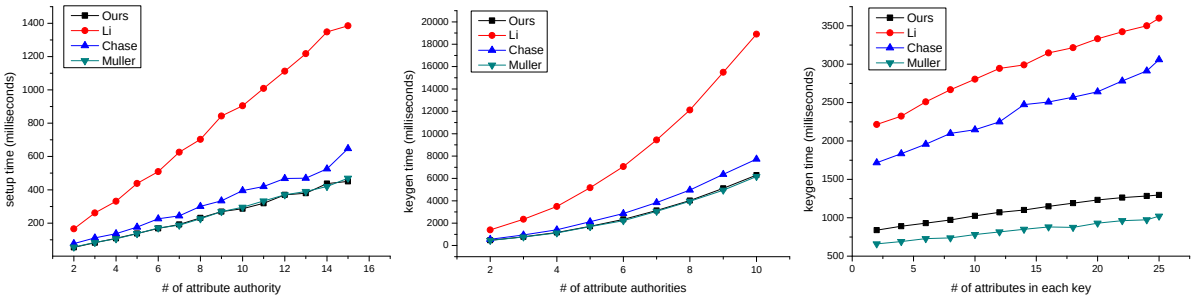
anonycontrol-dec : Decrypts a file if possible.

anonycontrol-rec : Decrypts a file and re-encrypts it under different privilege trees.

This toolkit is based on the CP-ABE toolkit [4] which is available online [38], and the whole system is implemented on a linux system with Intel i7 2^{nd} Gen @ 2.7GHz and 2GB RAM.

Fig. 2 shows the computation overhead incurred in the core algorithms Setup, KeyGenerate, Encrypt, and Decrypt under various conditions. We additionally implemented three similar works (Li [13], Chase [13], and Müller [12]) under the same condition (same security level and same environment) for the comparison purpose.

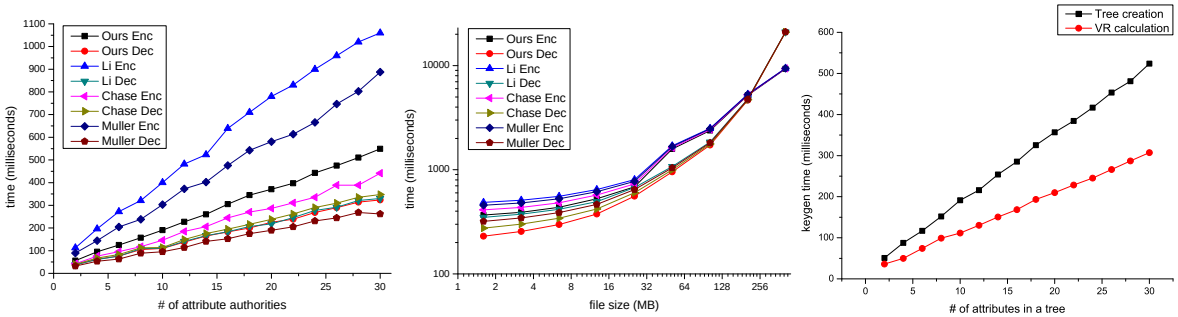
Particularly, in Fig. 2(e), we set only one privilege for the file access, and we measured the time to create one privilege tree and calculate its verification parameter in Fig. 2(f). In general, the computation overhead of Li [13] is much higher than others because their scheme involves many more exponentiations and bilinear mappings due to the accountability. The encryption/decryption under different file sizes did not show big differences when file sizes are large ($\geq$ 20MB), because the run times are dominated by the symmetric encryption (AES-256). Finally, only our run times are plotted in Fig. 2(f) because the privilege creation is the unique process in our scheme.



(a) Setup time

(b) Keygen time with different authorities #. 20 attributes per key.

(c) Keygen time with different attributes #. 4 authorities.



(d) Encryption and decryption time with different attributes number. File size is 100KB

(e) Encryption and decryption time with different file size. 20 attributes in T_0

(f) Time to create a privilege tree and decrypt a verification parameter from it

Fig. 2. Experiment result on our implemented prototype system

IX. Conclusion and Possible Extensions

This paper proposes a semi-anonymous attribute-based privilege control scheme *AnonyControl* and a fully-anonymous attribute-based privilege control scheme *AnonyControl-F* to address the user privacy problem in a cloud storage server. Using multiple authorities in the cloud computing system, our proposed schemes achieve not only fine-grained privilege control but also identity anonymity while conducting privilege control based on users' identity information. More importantly, our system can tolerate up to $N-2$ authority compromise, which is highly preferable especially in Internet-based cloud computing environment. We also conducted detailed security and performance analysis which shows that *AnonyControl* both secure and efficient for cloud storage system. The *AnonyControl-F* directly inherits the security of the *AnonyControl* and thus is equivalently secure as it, but extra communication overhead is incurred during the 1-out-of-n oblivious transfer.

One of the promising future works is to introduce the efficient user revocation mechanism on top of our anonymous ABE. Supporting user revocation is an important issue in the real application, and this is a great challenge in the application of ABE schemes. Making our schemes compatible with existing ABE schemes [39]–[41] who support efficient user revocation is one of our future works.

Acknowledgements

We offer our gratitude to our colleague Jingshan Yin, who implemented the 1-out-of-n protocol, and Shih-ming Huang from NCTU, who proofread our paper to correct minor errors.

References

[1] A. Shamir, "Identity-based cryptosystems and signature schemes," in *CRYPTO*. Springer, 1985, pp. 47–53.

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

[3] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *CCS*. ACM, 2006, pp. 89–98.

[4] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *S&P*. IEEE, 2007, pp. 321–334.

[5] M. Chase, "Multi-authority attribute based encryption," in *TCC*. Springer, 2007, pp. 515–534.

[6] M. Chase and S. S. Chow, "Improving privacy and security in multi-authority attribute-based encryption," in *CCS*. ACM, 2009, pp. 121–130.

[7] H. Lin, Z. Cao, X. Liang, and J. Shao, "Secure threshold multi authority attribute based encryption without a central authority," *Information Sciences*, vol. 180, no. 13, pp. 2618–2632, 2010.

[8] V. Božović, D. Socek, R. Steinwandt, and V. I. Villányi, "Multi-authority attribute-based encryption with honest-but-curious central authority," *IJCM*, vol. 89, no. 3, pp. 268–283, 2012.

[9] F. Li, Y. Rahulamathavan, M. Rajarajan, and R.-W. Phan, "Low complexity multi-authority attribute based encryption scheme for mobile cloud computing," in *SOSE*. IEEE, 2013, pp. 573–577.

[10] K. Yang, X. Jia, K. Ren, and B. Zhang, "Dac-macs: Effective data access control for multi-authority cloud storage systems," in *INFOCOM*. IEEE, 2013, pp. 2895–2903.

[11] A. Lewko and B. Waters, "Decentralizing attribute-based encryption," in *EUROCRYPT*. Springer, 2011, pp. 568–588.

[12] S. Müller, S. Katzenbeisser, and C. Eckert, "On multi-authority ciphertext-policy attribute-based encryption," *Bulletin of the Korean Mathematical Society*, vol. 46, no. 4, pp. 803–819, 2009.

[13] J. Li, Q. Huang, X. Chen, S. S. Chow, D. S. Wong, and D. Xie, "Multi-authority ciphertext-policy attribute-based encryption with accountability," in *ASIACCS*. ACM, 2011, pp. 386–390.

[14] H. Ma, G. Zeng, Z. Wang, and J. Xu, "Fully secure multi-authority attribute-based traitor tracing," *JCIS*, vol. 9, no. 7, pp. 2793–2800, 2013.

[15] S. Hohenberger and B. Waters, "Attribute-based encryption with fast decryption," in *PKC*. Springer, 2013, pp. 162–179.

[16] J. Hur, "Attribute-based secure data sharing with hidden policies in smart grid," *TPDS*, vol. 24, no. 11, pp. 2171–2180, 2013.

[17] Y. Zhang, X. Chen, J. Li, D. S. Wong, and H. Li, "Anonymous attribute-based encryption supporting efficient decryption test," in *ASIACCS*. ACM, 2013, pp. 511–516.

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

[19] A. Sahai and B. Waters, "Fuzzy identity-based encryption," *EUROCRYPT*, 2005.

[20] Z. Wan, M. Gu *et al.*, "Hierarchical attribute-set based encryption for scalable, flexible and fine-grained access control in cloud computing," in *ISPEC*. Springer, 2011, pp. 98–107.

[21] A. Kapadia, P. Tsang, and S. Smith, "Attribute-based publishing with hidden credentials and hidden policies," *NDSS*, 2007.

[22] S. Yu, K. Ren, and W. Lou, "Attribute-based content distribution with hidden policy," in *Workshop on Secure Network Protocols*. IEEE, 2008.

[23] Z. Wan, J. Liu, and R. H. Deng, "Hasbe: A hierarchical attribute-based solution for flexible and scalable access control in cloud computing," *Information Forensics and Security, IEEE Transactions on*, vol. 7, no. 2, pp. 743–754, 2012.

[24] T. Jung, X. Mao, X.-Y. Li, S.-J. Tang, W. Gong, and L. Zhang, "Privacy-preserving data aggregation without secure channel: Multivariate polynomial evaluation," in *INFOCOM*. IEEE, 2013, pp. 2634–2642.

[25] T. Jung and X.-Y. Li, "Collusion-tolerable privacy-preserving sum and product calculation without secure channel," *TDSC*, 2014.

[26] X.-Y. Li and T. Jung, "Search me if you can: privacy-preserving location query service," in *INFOCOM*. IEEE, 2013, pp. 2760–2768.

[27] L. Zhang, X.-Y. Li, Y. Liu, and T. Jung, "Verifiable private multi-party computation: ranging and ranking," in *INFOCOM*. IEEE, 2013, pp. 605–609.

[28] L. Zhang, X.-Y. Li, and Y. Liu, "Message in a sealed bottle: Privacy preserving friending in social networks," in *ICDCS*. IEEE, 2013, pp. 327–336.

[29] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *INFOCOM*. IEEE, 2010.

[30] C. Wang, K. Ren, and J. Wang, "Secure and practical outsourcing of linear programming in cloud computing," in *INFOCOM*. IEEE, 2011.

[31] C. Wang, N. Cao, J. Li, K. Ren, and W. Lou, "Secure ranked keyword search over encrypted cloud data," in *ICDCS*. IEEE, 2010.

[32] Y. Liu, J. Han, and J. Wang, "Rumor riding: anonymizing unstructured peer-to-peer systems," *TPDS*, vol. 22, no. 3, pp. 464–475, 2011.

[33] [Online]. Available: https://www.torproject.org/

[34] A. Shamir, "How to share a secret," *CACM*, vol. 22, no. 11, pp. 612–613, 1979.

[35] M. Naor and B. Pinkas, "Oblivious transfer and polynomial evaluation," in *STOC*. ACM, 1999, pp. 245–254.

[36] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," *CACM*, vol. 28, no. 6, pp. 637–647, 1985.

[37] W.-G. Tzeng, "Efficient 1-out-of-n oblivious transfer schemes with universally usable parameters," *TC*, vol. 53, no. 2, pp. 232–240, 2004.

[38] [Online]. Available: http://acsc.csl.sri.com/cpabe/

[39] W. Ren, K. Ren, W. Lou, and Y. Zhang, "Efficient user revocation for privacy-aware pki," in *ICST*, 2008, p. 11.

[40] 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," *TPDS*, vol. 24, no. 1, pp. 131–143, 2013.

[41] S. Yu, C. Wang, K. Ren, and W. Lou, "Attribute based data sharing with attribute revocation," in *ASIACCS*. ACM, 2010, pp. 261–270.

Taeho Jung received the B.E degree in Computer Software from Tsinghua University, Beijing, in 2007, and he is working toward the Ph.D degree in Computer Science at Illinois Institute of Technology while supervised by Dr. Xiang-Yang Li. His research area, in general, includes privacy & security issues in mobile network and social network analysis. Specifically, he is currently working on the privacy-preserving computation in various applications and scenarios.

Dr. Xiang-Yang Li is a professor at the Illinois Institute of Technology. He holds EMC -Endowed Visiting Chair Professorship at Tsinghua University. He currently is a distinguished visiting professor at XiAn JiaoTong University, and University of Science and Technology of China. He is a recipient of China NSF Outstanding Overseas Young Researcher (B). Dr. Li received MS (2000) and PhD (2001) degree at Department of Computer Science from University of Illinois at Urbana-Champaign, a Bachelor degree at Department of Computer Science and a Bachelor degree at Department of Business Management from Tsinghua University, P.R. China, both in 1995. His research interests include wireless networking, mobile computing, security and privacy, cyber physical systems, and algorithms. He and his students won three best paper awards (ACM MobiCom 2014, COCOON 2001, IEEE HICSS 2001), one best demo award (ACM MobiCom 2012)and was selected as best paper candidates twice (ACM MobiCom 2008, ACM MobiCom 2005). He published a monograph "Wireless Ad Hoc and Sensor Networks: Theory and Applications". He co-edited several books, including, "Encyclopedia of Algorithms". Dr. Li is an editor of several journals, including IEEE Transaction on Mobile Computing. He has served many international conferences in various capacities, including ACM MobiCom, ACM MobiHoc, IEEE MASS. He is a senior member of IEEE and a member of ACM.

Zhiguo Wan is a lecturer in School of Software, Tsinghua University, and he is also with State Key Laboratory of Information Security (Institute of Information Engineering, Chinese Academy of Sciences. His main research interests include security protocols, privacy enhancing techniques, and system security. He received his B.S. degree in computer science from Tsinghua University in 2002, and the Ph.D. degree from School of Computing, National University of Singapore in 2007.

Meng Wan is a professor at Center for Science and Technology DevelopmentMinistry of Education. He received his Ph. D. from Wuhan University in 2008, and his M.S. from Central University of Finance and Economics in 2000. His research interests include computer network architecture, network and systems management, science and technology management, system engineering etc. He is current served as the division director of Department of Network and Information, Center for Science and Technology DevelopmentMinistry of Education and the associate editor of Sciencepaper Online. He is a member of the IEEE,ACM.