Controlling Cloud Data Access Privilege: Cryptanalysis and Security Enhancement

Yaser Baseri[1], Abdelhakim Hafid[1], Mohammed Amine Togou[1], and Soumaya Cherkaoui[2]

[1]Department of Computer Science and Operational Research, University of Montreal
[2]INTERLAB Research Laboratory, Université de Sherbrooke
[1]yaser.baseri@umontreal.ca, {togoumoh, ahafid}@iro.umontreal.ca
[2]soumaya.cherkaoui@usherbrooke.ca

***Abstract*—Recently, Jung et al. [1] proposed a data access privilege scheme and claimed that their scheme addresses data and identity privacy as well as multi-authority, and provides data access privilege for attribute-based encryption. In this paper, we show that this scheme, and also its former and latest versions (i.e. [2] and [3] respectively) suffer from a number of weaknesses in terms of fine-grained access control, users and authorities collusion attack, user authorization, and user anonymity protection. We then propose our new scheme that overcomes these shortcomings. We also prove the security of our scheme against user collusion attacks, authority collusion attacks and chosen plaintext attacks. Lastly, we show that the efficiency of our scheme is comparable with existing related schemes.**

Keywords: **Attribute-Based Encryption, User Anonymity, Data Privacy, Outsourcing Decryption.**

I. Introduction

Cloud computing has emerged as a promising technology to make use of highly scalable servers and reduce the operational costs of individual users and enterprises. It enhances collaboration, agility and scale, and provides a global computing model over the Internet infrastructure. However, because of the challenges associated with security and privacy, there is a widespread concern in using this type of technology. Secure data protection and authorized access provision is one of these challenges whose violation can cause unauthorized use of resources and services. *Attribute-Based Encryption* (*ABE*) is a cryptographic approach that makes access decision based on attributes and policies in an encrypted way [4], [5]. It allows data owners to define access policies and encrypt data based on that policy. It is a one-to-many public-key cryptography used to enforce fine-grained access policies. In particular, *Ciphertext Policy ABE* (*CP-ABE*) provides access such that encrypted data can be decrypted only by a user possessing a set of attributes. Thus, based on access policy embedded in ciphertext, different users are able to access different pieces of information based on the attributes they are assigned. Since *ABE* encrypts data without exact knowledge of receivers, it is suitable for large-scale applications.

978-1-5386-3531-5/17/$31.00

Providing fine-grained access control for *ABE* requires issuing different attributes for each user. These attributes can be issued either by a single authority or multiple authorities. In single authority *ABE*, the authority should be fully trusted to verify users' legitimacy and issue their whole secret keys. While, in multi-authory *ABE* [6], [7], each authority is responsible for verifying the legitimacy of users for a bunch of attributes and issuing the corresponding part of secret keys.

Security and privacy do not only focus on the data content stored in the cloud. Privacy protection of users and preservation of their anonymity against malicious authorities and providers is the other aspect which is necessary for critical and sensitive information. In some applications, such as e-heath, there might be some attributes containing sensitive and private information like type of diseases; this information should be secret and malicious authorities and providers should not be allowed to have access.

In this paper, we discuss how to protect user privacy in multi-authority attribute-based encryption. We review three existing contributions [1]–[3], and show their weaknesses in fine-grained access control, users and authorities collusion, and user authorization. Then, we propose a *Fine-grained Access Control Scheme (FACS)* which supports multi-authority, anonymizes users' identity, and is immune against user collusion attacks, authority collusion attacks and chosen plaintext attacks. The security analysis and performance evaluation show that *FACS* is suitable to provide anonymous attribute-based encryption.

The rest of this paper is organized as follows. Section II presents some preliminaries. Section III analyzes the security of [1]–[3]. Section IV discusses the system and security models of the proposed scheme. Section V describes the proposed *FACS*. Section VI evaluates the the security and performance of *FACS*. Finally, Section VIII concludes the paper and presents future work.

II. Preliminaries

In this section, we briefly introduce concepts related to tree access structure which will be used in the coming

sections.

A. Tree Access Structure

To enforce fine-grained access control and describe encryption policy, we adapt tree access structure, definded in [8], to our scheme. Let $\mathbb{A}_u$ be a user's set of attributes, $\mathcal{T}$ be a tree with root R representing an access structure, $\mathcal{T}_x$ be a sub-tree of $\mathcal{T}$ rooted at node x, $att(x)$ denotes the attribute associated with the leaf node x, num_x be the number of child nodes of non-leaf node x, k_x be a threshold value $0 \leq k_x \leq num_x$. Then, node x is assigned a true value if at least k_x child nodes of x have been assigned true value. Particularly, the node becomes an OR gate when $k_x = 1$ and an AND gate when $k_x = num_x$.

Definition 1 (Satisfying a Tree Access Structure)**.** If user's attribute set $\mathbb{A}_u$ satisfies the tree access structure $\mathcal{T}$ or the node x, we denote it as $\mathcal{T}(\mathbb{A}_u) = 1$ or $\mathcal{T}_x(\mathbb{A}_u) = 1$ respectively. $\mathcal{T}_x(\mathbb{A}_u)$ can be calculated recursively as follows: If x is a leaf node, then $\mathcal{T}_x(\mathbb{A}_u)$ is equal to 1 if and only if $att(x) \in \mathbb{A}_u$. If x is a non-leaf node, then $\mathcal{T}_x(\mathbb{A}_u)$ is equal to 1 when at least k_x child nodes of x returns 1. $\mathcal{T}(\mathbb{A}_u) = 1$ if and only if $\mathcal{T}_R(\mathbb{A}_u) = 1$.

III. Security Attacks

Jung et al. [1] proposed a multi-authority access control scheme (an extension to the contribution in [2]) and claimed that their work support data privacy, user anonymity and data access privilege. They introduce *1-Out-of-n Oblivious Transfer* technique to protect user's identity and achieve full anonymity. However, the scheme suffers from the following shortcomings:

User Authorization Attack: Assume that a malicious user *U* has been authorized and received $D = g^{\Sigma v_k + d_k}$ and $x_k.g^{d_k}$ to generate his own secret key $SK_u = \{D = g^{\Sigma v_k + d_k}, \forall i \in \mathbb{A}_u : D_i = H(att(i))^{r_i}.g^{\Sigma d_k}, D'_i = g^{r_i}\}$ in key generation phase. He can calculate $\hat{D} = \prod x_k.g^{d_k}$, select a random number $r_{i'}$ and forge secret key for any attribute $(att(i')$ (i.e. $D_{i'} = H(att(i))^{r_{i'}}.g^{\Sigma d_k}, D'_i = g^{r_{i'}}$) and thus be authorized to access any file he wants.

Apart from this attack, Jung et al. [1] proposed an approach to achieve full anonymity. They assume k possible attribute values for each attribute requested by user *U*. Then, using *1-Out-of-n Oblivious Transfer* technique, *U* asks for just one attribute value he wants to be issued by the authority. A malicious user may request to have any other attribute value from the same set, which may not necessarily belong to him. In this way, a non-eligible requester should be authorized by an authority responsible for issuing attributes anonymously.

Coarse-Grained Access Control: In key generation phase, attribute authorities collaborate to issue a secret parameter D as part of secret key for each user. In encryption phase, D is used to compute $\hat{C} = D^{h^{-1}}$. This means that data owner should use different Ds for different users to encrypt files restricting the "fine-grained" property of the scheme.

Jung et al. [3] extended their contribution in [2] and made it immune against the leakage of master secret key [9]. However, their new scheme suffers from the following problems:

User and Authority Collusion Attack: To generate secret key for each user, attribute authority AA_k, for each attribute $att(i) \in \mathbb{A}_u$ it is responsible for, selects random number $r_i \in_r \mathbb{Z}_q^*$, computes $\hat{D}_i = H(att(i))^{r_i}.x_k.g^{d_k}$ and $D'_i = g^{r_i}$, and sends them to that user. Let us assume that attribute $att(j)$ is issued by malicious authority AA_k. Moreover, assume that malicious user *U* is authorized to receive secret key $SK_u = \{D = g^{\Sigma v_k + d_k}, \forall i \in \mathbb{A}_u : D_i = H(att(i))^{r_i}.g^{\Sigma d_k}, D'_i = g^{r_i}\}$. In a collusion with malicious user *U* and malicious authority AA_k, instead of $\hat{D}_j = H(att(j))^{r_j}.x_k.g^{d_k}$, AA_k can send $x_k.g^{d_k}$ to *U*. In this case, *U* is able to compute $\prod_{i=1}^{\mathcal{K}} x_i.g^{d_i} = g^{\Sigma d_i}$, and forge any attribute he wants.

Coarse-Grained Access Control: Similarly to [1] and [2], the scheme in [3] suffers from coarse-grained access control.

IV. System and Security Models

In this section, we present the system model, and the framework of *FACS*. We also define the security model used to prove the security of *FACS* against chosen plaintext attacks.

A. System Model

In the architecture considered for *FACS*, there are four entities: $\mathcal{K}$ *Attribute Authorities* (AA_i), *User* (*U*), *Cloud Service Provider* (*CSP*), and *Data Owner* (*DO*). Attribute authorities $AA_{i(1 \leq i \leq \mathcal{K})}$ issue a set of public parameters and generate a set of secret keys for users according to their attributes. *U*, who wants to access a file, sends his query including his attributes, issued by authorities, to *CSP*. *DO* defines an access policy, encrypts data based on that policy and stores it into the cloud. Based on attributes embedded in access request and access policy associated with ciphertext, *CSP* partially decrypts data and sends it back to *U*. If authorized, *U* will be able to decrypt the received data.

It is assumed that attribute authorities are untrusted in the sense that they will follow the scheme, but, they may try to collude with users or other authorities and forge a new key without presence of some authorities. Users might be malicious, collude together or with authorities and forge their attributes to escalate their rights and get access to services and information they are not eligible for. In our architecture, we assume that *CSP*s are honest but curious in practice. That means that *CSP*s will faithfully follow the proposed scheme, but can launch passive attacks to get as much secret information as possible. Hence, the data stored in the

cloud should remain encrypted all the time and any required transformation should not reveal the plaintext in the process.

B. Framework

The framework of the proposed scheme is composed of the following six algorithms:

Setup: The setup algorithm, which is run by attribute authorities, takes as input the attribute universe $\mathbb{U}$ and an implicit security parameter Λ. It outputs public parameters PK as well as master secret key MK_{a_i} for each attribute authority AA_i.

KeyGeneration: The key generation algorithm, which is run by attributes authorities, takes as input public parameters PK, master secret keys $MK_{a_i(1\leq i\leq \mathcal{K})}$ and user's set of attributes $\mathbb{A}_u$. It generates secret key SK_U, corresponding to the input attribute set $\mathbb{A}_u$, which is sent to U.

KeyGenOut: This algorithm, which is run by U, takes as input public parameters PK, user's secret key SK_U and a secret retrieving key b known only to U. It returns as output a blinded secret key $SK_U^{1/b}$, which is sent to *CSP*.

Encryption: The encryption algorithm, which is run by *DO*, takes as input public parameters PK, a tree access structure $\mathcal{T}$, that determines access policy, and a message M; it generates a ciphertext CT as output.

DecryptionOut: The algorithm, which is run by *CSP*, takes as input the public parameters PK, a blinded secret key $SK_U^{1/b}$ and a ciphertext CT. If the attributes $\mathbb{A}_u$, associated with user, satisfies the policy associated with CT, it computes and returns a transformed ciphertext CT'.

Decryption: The decryption algorithm, which is run by user, takes as input the public parameters PK, a transformed ciphertext CT' and a secret retrieving key b, and returns a message M as output.

C. Security Model

The selective security and indistinguishability against chosen plaintext attacks (*IND-CPA*) for *FACS* is defined by the following game between challenger $\mathcal{C}$ and adversary $\mathcal{A}$:

Initialization. $\mathcal{A}$ declares the set of at most $\mathcal{K}-1$ compromised authorities that are under his control. The remaining authorities are controlled by $\mathcal{C}$. $\mathcal{A}$ also commits a tree access structure $\mathcal{T}$ for the game.

Setup. $\mathcal{C}$ and $\mathcal{A}$ jointly run *Setup* algorithm to obtain the valid parameters.

Learning 1. $\mathcal{A}$ is able to query for an arbitrary number (i.e. p) of secret keys corresponding to chosen attribute sets $\mathbb{A}_{u_1},\ldots,\mathbb{A}_{u_p}$ for a selected number of users $\{U_1,\ldots,U_p\}$. It is also allowed to query for an arbitrary number (i.e. $q-p$) of blinded secret keys corresponding to attribute sets $\mathbb{A}_{u_{p+1}},\ldots,\mathbb{A}_{u_q}$ for selected number of users $\{U_{p+1},\ldots,U_q\}$. These attribute sets are disjointly issued by all authorities, but none of them satisfies the tree access structure $\mathcal{T}$. Furthermore, $\mathcal{A}$ can conduct an arbitrary number of computations, using its own (or compromised) secret (or blinded secret) keys, and all public parameters.

Challenge. $\mathcal{A}$ sends two distinct chosen plaintexts M_0 and M_1 to $\mathcal{C}$. $\mathcal{C}$ randomly selects bit $\varpi\in\{0,1\}$, encrypts M_ϖ with respect to tree access structure $\mathcal{T}$ and returns the result to $\mathcal{A}$.

Learning 2. $\mathcal{A}$ continues to repeat learning 1 adaptively.

Response. $\mathcal{A}$ outputs guess ϖ' of ϖ.

Definition 2. *FACS* is selective secure and indistinguishable against chosen plaintext attacks (*IND-CPA*), if all probabilistic polynomial time adversaries have only a negligible advantage in the above game. An adversary is said to have the advantage ε, if it wins the game with probability $Pr[\varpi'=\varpi]=\left(\frac{1}{2}+\varepsilon\right)$

V. Construction of Our New Algorithm

In this section, we describe the construction of the proposed scheme. *FACS* contains six algorithms which are performed in four phases: setup, key generation, encryption, and decryption.

Setup Phase In the setup phase, attribute authorities $AA_{i(1\leq i\leq\mathcal{K})}$, perform $Setup$ algorithm to fix public parameters as well as master secret keys. In this phase, each attribute authority AA_k

- Aggregates on prime number p and generator g for $\mathbb{Z}_p^*$ with other authorities. This can be processed by one authority and shared with other ones.
- Chooses random secret parameter $v_k\in\mathbb{Z}_p^*$ and computes $e(g,g)^{v_k}$ and shares them with all other authorities.
- Selects random integer $s_{kj}\in\mathbb{Z}_p^*(j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\})$, computes $g^{s_{kj}}$ and shares it with other authorities $AA_j(j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\})$.
- Receives $\mathcal{K}-1$ pieces of $g^{s_{jk}}$ and $e(g,g)^{v_j}$ generated by other authorities $AA_j(j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\})$. It selects random exponent $\tau_k\in\mathbb{Z}_p^*$, computes $\vartheta_k=g^{\tau_k}$ and $\zeta=\prod_{i\in\{1,\ldots,\mathcal{K}\}}e(g,g)^{v_i}=e(g,g)^{\sum_{i=1}^{\mathcal{K}}v_i}$ and publishes ϑ_k and ζ as authorities' public parameters. Aggregating authorities' public parameters constructs the public parameters $PK=(\mathbb{G},g,\zeta=e(g,g)^{\sum_{i=1}^{\mathcal{K}}v_i},\vartheta_i=g^{\tau_i}(1\leq i\leq\mathcal{K}),H(.))$.
- Computes authority secret key $x_k\in\mathbb{Z}_p^*$ as follows:

$$x_k=\Big(\prod_{j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\}}g^{s_{kj}}\Big)/\Big(\prod_{j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\}}g^{s_{jk}}\Big)$$
$$=g^{\left(\sum_{j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\}}g^{s_{kj}}-\sum_{j\in\{1,\ldots,\mathcal{K}\}\setminus\{k\}}g^{s_{jk}}\right)}$$

It can be observed that these randomly produced integers satisfy $\prod_{k\in\{1,\ldots,\mathcal{K}\}}x_k=1\ mod\ p$. Aggregating global master key along with authority's secret parameters v_k, τ_k and x_k constructs master

secret key for attribute authority AA_k (i.e. $MK_{a_k} = (v_k, \tau_k, x_k)$).

Key Generation Phase In the key generation phase, attribute authorities $AA_{i(1\leq i\leq \mathcal{K})}$ collaborate to perform $KeyGeneration$ algorithm and issue secret key for each user. Then, a typical user U performs $KeyGenOut$ to anonymize his secret key and sends his access request to *CSP*. To perform $KeyGeneration$, each attribute authority AA_k

- Selects a random number $d_k \in \mathbb{Z}_p^*$, computes $x_k.g^{d_k}$ and $x_k.g^{(v_k+d_k)}$, and shares them with all other authorities $AA_i(i \in \{1,\ldots,\mathcal{K}\} \setminus \{k\})$.
- Receives $\mathcal{K}-1$ pieces of $x_k.g^{d_k}$ and $x_i.g^{(v_i+d_i)}$ generated by $AA_i(i \in \{1,\ldots,\mathcal{K}\} \setminus \{k\})$.
- Computes $D_{\Sigma d_i} = \prod_{\mathcal{K}}^{i=1} x_i.g^{d_i} = g^{\sum_{i=1}^{\mathcal{K}} d_i}$ and $D = \prod_{i=1}^{\mathcal{K}} x_i.g^{(v_i+d_i)} = g^{\sum_{i=1}^{\mathcal{K}}(v_i+d_i)}$, and sends D to user U.
- Chooses a random number $r_j \in_r \mathbb{Z}_q^*$ for each attribute $att(j) \in \mathbb{A}_u$, computes $D_j = D_{\Sigma d_i}^{(1/\tau_k)}(H(att(j)))^{r_j} = g^{(\sum_{i=1}^{\mathcal{K}} d_i/\tau_k)}(H(att(j)))^{r_j}$ and $D'_j = g^{r_j}$ and sends them to user U.

To perform $KeyGenOut$, user U

- Chooses random number $b \in \mathbb{Z}_p^*$ as secret retrieving key RK_U and transforms secret key $SK_U = (D, \forall j \in \mathbb{A}_u : (D_j,\ D'_j))$ to its blinded version. This is performed by raising its components to the power $1/b$ (i.e. $SK_U^{1/b} = (D^{1/b}, \forall j \in \mathbb{A}_u : (D_j^{1/b}, D_j'^{1/b}))$.

Encryption Phase In the encryption phase, data owner *DO* performs $Encryption$ algorithm to define access policy and encrypt data based on that policy. In this process, data owner *DO*

- Defines access control policy for all attributes.
- Selects random number s in $\mathbb{Z}_p^*$ and computes $E = M.e(g,g)^{\sum_{i=1}^{\mathcal{K}} v_i s}$.
- Shares secret s in tree access structure $\mathcal{T}$ with root R. In a top-down approach, it chooses a polynomial q_x for each node x in the tree access structure $\mathcal{T}$ as follows: For each node x, the algorithm sets degree d_x of polynomial q_x to $k_x - 1$ where k_x is the threshold value of node x. Starting from root node R, the algorithm sets $q_R(0) = s$ and randomly chooses d_R other coefficients to define q_R completely. For any other node x, it sets $q_x(0) = q_{parent(x)}(index(x))$[1] and chooses d_x other coefficients randomly to completely define q_x. Note that each leaf node y is associated with atomic attribute $att(y)$in the set of attributes.
- Uploads the ciphertext $CT = \Big(E = M.e(g,g)^{\sum_{i=1}^{\mathcal{K}} v_i s}, C = g^s, \forall y \in \mathbb{A}_{\mathcal{T}} : (C_y = \vartheta_{iss(y)}^{q_y(0)} = g^{(\tau_{iss(y)}\ q_y(0))}$ [2], $C'_y = H(att(y))^{q_y(0)})\Big)$ to *CSP*.

[1] $parent(x)$ is the node x's parent node and $index(x)$ is a number associated with each node x ranging from 1 to $num_{parent(x)}$.

[2] $iss(y)$ is the authority responsible for issuing $att(y)$.

Note that, to adopt the key encapsulation mechanism and reduce the size of ciphertext without scarifying security [10], [11], instead of computing $M.e(g,g)^{\sum_{i=1}^{\mathcal{K}} v_i s}$ for plaintext M, *DO* selects a random content key CK and computes $CK.e(g,g)^{\sum_{i=1}^{\mathcal{K}} v_i s}$. It uses the content key CK to encrypt the target file (i.e., the file we want to encrypt and for which we define access control) with symmetric encryption $ENC_{CK}(.)$.

Decryption Phase When user U sends his access request to a ciphertext, *CSP* performs $DecryptionOut$ algorithm to check his eligibility to access, outsource the computation cost and partially decrypt the ciphertext. Then, U performs $Decryption$ algorithm to to have access to the corresponding plaintext. To perform $DecryptionOut$, *CSP*

- Computes $DecNode(y) = \frac{e(D_j^{1/b}, C_y)}{e(D_j'^{1/b}, C'_y)} = e(g,g)^{(\sum_{i=1}^{\mathcal{K}} d_i.q_y(0))/b}$ for each leaf node y in tree access structure $\mathcal{T}$ and corresponding attribute $att(j) \in \mathbb{A}_u$. Then, it recursively computes A as $DecNode(R)$ for root node R of $\mathcal{T}$ as follows: For each non-leaf x, the algorithm calls $DecNode(z)$ for all child nodes z of x and stores the output as F_z. Let S_x be an arbitrary k_x-sized set of child nodes z; we can compute $DecNode(x)$ as follows:

$$
\begin{aligned}
F_x &= \Pi_{z\in S_x} F_z^{\Delta_{i,S'_x}(0)},\ where\ \begin{cases} i &= index(z) \\ S'_x &= \{index(z) : z \in S_x\} \end{cases} \\
&= \Pi_{z\in S_x}\ \big(e(g,g)^{(\sum_{i=1}^{\mathcal{K}} d_i.q_z(0))/b}\big)^{\Delta_{i,S'_x}(0)} \\
&= \Pi_{z\in S_x}\ \big(e(g,g)^{(\sum_{i=1}^{\mathcal{K}} d_i.q_{parent(z)}(index(z)))/b}\big)^{\Delta_{i,S'_x}(0)} \\
&= \Pi_{z\in S_x}\ e(g,g)^{\sum_{i=1}^{\mathcal{K}} d_i.(q_x(i)\Delta_{i,S'_x}(0))/b} \\
&= e(g,\omega)^{\sum_{i=1}^{\mathcal{K}} d_i.q_x(0)/b}
\end{aligned}
$$

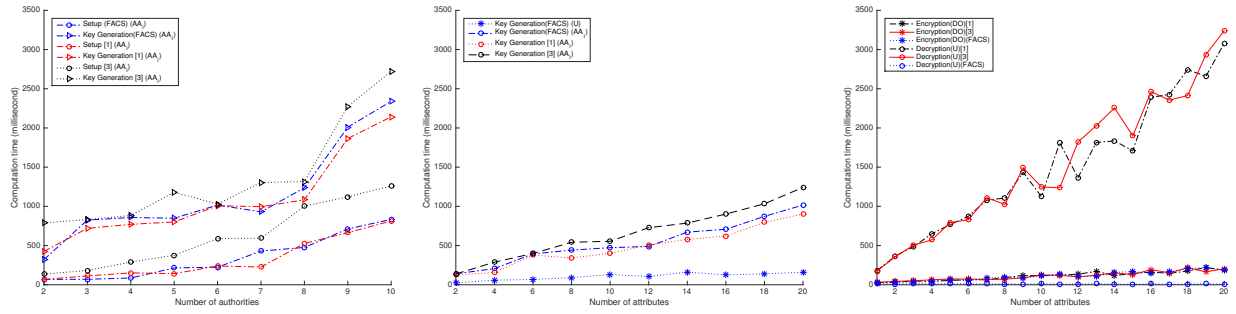
  where $\Delta_{i,S'_x}(0)$ is the Lagrange coefficient which is defined as $\Delta_{i,S'_x}(x) = \Pi_{j\in S, j\neq i} \frac{x-j}{i-j}$ [8]. The algorithm begins by simply calling the function for root node R of the tree access structure $\mathcal{T}$. If the tree access structure is satisfied by $\mathbb{A}_u$, then A will be equal to $e(g,g)^{\sum_{i=1}^{\mathcal{K}} d_i.q_R(0)/b} = e(g,g)^{\sum_{i=1}^{\mathcal{K}} d_i.s/b}$.
- Computes $M^{1/b} = \frac{\hat{e}(C, D^{1/b})}{A} = \frac{e(g,g)^{\sum_{i=1}^{\mathcal{K}}(v_i+d_i)s/b}}{e(g,g)^{\sum_{i=1}^{\mathcal{K}} d_i.s/b}} = e(g,g)^{\sum_{i=1}^{\mathcal{K}} v_i s/b}$ and sends it to the user.

To perform $Decryption$, user U

- Computes plaintext M by raising $M^{1/b}$ to the power b selected in key generation phase.

VI. Security and Performance Evaluation

The security analysis includes considering immunity of *FACS* against authorities collusion attacks, user collusion attacks and chosen plaintext attacks. It also discusses how *FACS* anonymizes users' identity. Due to space constraints, the analysis is not included in the paper; it will be made available for interested readers.



(a) (b) (c)

Figure 1: Computation time analysis: (a) Setup and key generation (5 attributes per authority), (b) Key generation ($\mathcal{K} = 5$), (c) encryption and decryption times ($\mathcal{K} = 5$)

To evaluate the performance of the proposed schemes, we compare computation time of *FACS* against state of the art, including [1], [3]. To simulate the results, we make use of *MNT159* curves type D with the embedding degree $k = 6$. Our evaluation is based on *Java* realization for *CP-ABE* toolkit and uses *Java Pairing-Based Cryptography* (*jPBC*) library [12]. The analysis is conducted based on computation time evaluation on a platform consisting of (a) a large instance Amazon EC2 with the configuration of 7.5 GB RAM, 4 EC2 compute units, 850 GB instance storage and 64-bit platform as *CSP*, and (b) an Apple iMac with Intel Core 2 Duo at 2.66 GHz, 4 GB RAM as data owner, user and attribute authorities. We evaluate the time overhead caused by authorities and users interactions in key generation phase using *OMNET++* and *INET* framework.

Figure 1 shows the computation time of different phases as a function of number of authorities and number of attributes. When varying either the number of authorities (see Figure 1a) or the number of attributes (see Figure 1b) *FACS* incurs slighty larger computation times for aauthorites in setup and key generation phases compared to those incurred by [1]. When varying either the number of authorities (see Figure 1a) or the number of attributes (see Figure 1b) *FACS* incurs slighty larger computation times for aauthorites in setup and key generation phases compared to those incurred by [1]. In addition, *FACS* incurs a small computation overhead at the user side since it makes use of *KeyGenOut* algorithm (see Section V). Yet, *FACS* incurs the lowest computation time of decryption, almost constant for user irrespective of the number of attributes (see Figure 1c).

VII. Conclusion

In this paper, we did discuss how to provide user privacy in multi-authority attribute-based encryption. After reviewing existing contributions [1]–[3], and showing their weaknesses, we proposed *FACS* as a fine-grained access control scheme which supports multi-authority, anonymizes users' identity, and is immune against user collusion attacks, authority collusion attacks and chosen plaintext attacks. The security analysis and performance evaluation showed that *FACS* is suitable to provide anonymous attribute-based encryption.

References

[1] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Control cloud data access privilege and anonymity with fully anonymous attribute-based encryption," *Information Forensics and Security, IEEE Transactions on*, vol. 10, no. 1, pp. 190–199, Jan 2015.

[2] T. Jung, X. Y. Li, Z. Wan, and M. Wan, "Privacy preserving cloud data access with multi-authorities," pp. 2625–2633, 2013.

[3] T. Jung, X.-Y. Li, Z. Wan, and M. Wan, "Rebuttal to "comments on 'control cloud data access privilege and anonymity with fully anonymous attribute-based encryption'"," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 868–868, April 2016.

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

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

[6] R. Jiang, X. Wu, and B. Bhargava, "Sdss-mac: Secure data sharing scheme in multi-authority cloud storage systems," *Computers & Security*, vol. 62, pp. 193–212, 2016.

[7] W. Li, K. Xue, Y. Xue, and J. Hong, "Tmacs: A robust and verifiable threshold multi-authority access control system in public cloud storage," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1484–1496, May 2016.

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

[9] H. Ma, R. Zhang, and W. Yuan, "Comments on "control cloud data access privilege and anonymity with fully anonymous attribute-based encryption"," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 4, pp. 866–867, 2016.

[10] J. Lai, R. H. Deng, C. Guan, and J. Weng, "Attribute-based encryption with verifiable outsourced decryption," *Information Forensics and Security, IEEE Transactions on*, vol. 8, no. 8, pp. 1343–1354, 2013.

[11] M. Green, A. Akinyele, and M. Rushanan, "Libfenc: The functional encryption library," *Avaiable from http://code. google. com/p/libfenc. Baodong Qin received the B. Sc. degree*, 2004.

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