Who Is Touching My Cloud

Hua Deng[1,2,3], Qianhong Wu[2,5], Bo Qin[3], Jian Mao[2],
Xiao Liu[2], Lei Zhang[4], and Wenchang Shi[3]

[1] School of Computer, Wuhan University, Wuhan, China
denghua@whu.edu.cn
[2] School of Electronic and Information Engineering, Beihang University, Beijing, China
{qianhong.wu,maojian}@buaa.edu.cn
[3] School of Information, Renmin University of China, Beijing, China
{bo.qin,wenchang}@ruc.edu.cn
[4] Software Engineering Institute, East China Normal University, Shanghai, China
leizhang@sei.ecnu.edu.cn
[5] The Academy of Satellite Application, Beijing

Abstract. Advanced access controls have been proposed to secure sensitive data maintained by a third party. A subtle issue in such systems is that some access credentials may be leaked due to various reasons, which could severely damage data security. In this paper, we investigate leakage tracing enabled access control over outsourced data, so that one can revoke the suspected leaked credentials or prepare judicial evidences for legal procedure if necessary. Specifically, we propose a leaked access credential tracing (LACT) framework to secure data outsourced to clouds and formalize its security model. Following the framework, we construct a concrete LACT scheme that is provably secure. The proposed scheme offers fine-grained access control over outsourced data, by which the data owner can specify an access policy to ensure that the data is only accessible to the users meeting the policy. In case of suspectable illegal access to outsourced data with leaked credentials, a tracing procedure can be invoked to tracing in a black-box manner at least one of the users who leaked their access credentials. The tracing procedure can run without the cloud service provider being disturbed. Analysis shows that the introduction of tracing access credential leakage incurs little additional cost to either data outsourcing or access procedure.

Keywords: Data privacy, Access control, Cloud storage, Access credential leakage, Digital forensics.

1 Introduction

Cloud computing services provide an efficient and cost-effective mechanism for individuals and organizations to enforce highly scalable and technology-enabled management on their data. This new and exciting paradigm has generated significant interests in both industrial and academic world, resulting a number of notable theoretical and practical cloud computing models, such as Amazon EC2, Apple iCloud, Microsoft Azure and some more complex models designed for multi-cloud [21]. In the context of cloud storage [20, 23], users can outsource their data to a cloud storage server (maintained by

M. Kutyłowski and J. Vaidya (Eds.): ESORICS 2014, Part I, LNCS 8712, pp. 362–379, 2014.

a cloud service provider, CSP), so that themselves and other authorized users can access the outsorced data anytime and anywhere. In this way, users are able to share their data with others without worrying about their local hardware and software limitations.

Although cloud storage brings about many benefits, the concerns on data security are believed the major obstacles for the wide usage of cloud services. When users outsource their data to clouds, they may worry about unauthorized data access due to the loss of physical control of their data. Encryption is a standard approach to protect data security but traditional cryptosystems, including symmetric and asymmetric cryptosystems, can not support complicated access policy or suffer from complicated key management in securing outsourced data with flexible access policies. Nevertheless, in cloud storage scenario, users usually do not know who will request to access their data in the future, so a flexible access control over data is desired; besides, it is not practical to issue an access key for each authorized requestor. Attribute-based encryption (ABE, [9, 24]) is a recently proposed promising approach to enable flexible access control on the data outsourced to clouds. In an ABE system, data owners can specify access policies over attributes that the potential authorized users should possess. Then the authorized users with the attributes satisfying the specified access policy can access the outsourced data.

The attribute-based cryptosystem provides a reliable method to protect the data in clouds, while at the same time enabling fine-grained access control over the data. This is realized by assigning access credentials to authorized users so that the encrypted data are only accessible to them. In practice, these access credentials may be leaked due to various reasons, e.g., intentionally leaked by authorized users for their own benefits or compromised by hackers. For instance, a company employs a cloud storage system to store its data and assigns access credentials to its employees. It is possible that some employees unsatisfied with the company disclose their access credentials to the company's competitors who are interested in the sensitive data stored on the clouds. Once this happens, some countermeasures should be taken to find the leaked credentials in order to prevent illegal access in future.

There are some solutions for the purpose of tracing leaked access credentials. Boneh *et al.* [4, 5] provided a traitor tracing mechanism in broadcast encryption system, while only achieving a gross-grained access control over data. Based on the works [4, 5], the schemes [16–18] resolved the problem of tracing leaked access credentials in attribute-based encryption. Although these schemes achieve fine-grained access control, they either only possess a weak tracing capability or suffer from large-size ciphertexts. A desired solution in the cloud-based scenario is what on the one hand provides fine-grained access control over outsourced data, on the other hand fulfills a strong tracing mechanism to find leaked access credentials, and at the same time achieves short ciphertexts for outsourced data.

1.1 Our Contributions

In this paper, we investigate security-enhanced access control over the data stored in the cloud server, so that an access credential leakage tracing mechanism can be incorporated to find leaked access credentials used for illegal access. We propose a feasible solution to find leaked access credentials with strong tracing capability and achieve

short ciphertexts for outsourced data in a cloud-based scenario. Our contributions include the following aspects.

We present a leaked access credential tracing (LACT) framework to secure outsourced data in cloud storage. It allows a user to define an access policy for each file or its any content to be outsourced. Authorized cloud clients will be given a secret access credential for access to outsourced data. In case of illegal access with leaked access credentials, in a black-box way, a tracing procedure can find at least one of the leaked credentials, even if the illegal access credentials were produced with multiple leaked credentials in an unknown way. This implies that the trusted third party does not need to know how the illegal access credentials were forged, which captures powerful collusion attacks in practice.

Following the generic framework, we propose a concrete LACT scheme that is provably secure. When an access credential is assigned to an authorized user associated with his/her attributes, an unique fingerprint code is embedded into each user's access credential. During outsourcing a file, the file owner encrypts the file with a policy, so that only the users having access credentials of the matching attributes can decrypt the outsourced file, without fully trusting the cloud storage provider. When some access credentials are leaked and used to forge illegal credentials for unauthorized access, a trusted third party is employed to find at least one of the leaked credentials involved in the illegal access. Surprisingly, the tracing procedure does not disturb the CSP. The security properties are formally defined and proved by assuming the security of the underlying ABE scheme and the fingerprint codes scheme.

We analyze our LACT scheme and compare it with some up-to-date similar works. The analysis shows that the introduction of a black-box leakage tracing countermeasure does not incur any significant cost to the access control. The comparison demonstrates that the proposed scheme achieves a strong traceability with barely expanding the ciphertexts. Indeed, the most critical data outsourcing and access sub-protocols, which determine the practicality of the system, are almost as efficient as the underlying attribute-based access control model which does not provide any leakage tracing mechanism. Our scheme also supports any access structure admitting a linear secret sharing and enables fine-gained access control over outsourced data. These features make our scheme a practical solution to secure sensitive data outsourced to untrusted clouds.

1.2 Related Work

There is an increasing demand to secure data maintained by a third party in distributed computing systems. Asokan *et al.* [1] proposed a framework to safely share sensitive data among mobile devices, with the focus on the data privacy and user privacy. Huang *et al.* [10] exploited the attribute-based encryption (ABE) to protect user's privacy in mobile systems. Considering that in the ABE applied into mobile systems a single authority is too easy to be broken, Li *et al.* [13] proposed a multi-authority solution to reduce the power of single authority and alleviate overheads of mobile users.

In cloud computing environments, the protection of data security is pressing because of the scalability and easy accessibility requirements. Due to the fine-grained access control feature, ABE has been extensively employed in cloud computing to protect data security. Liu *et al.* proposed an elegant ABE scheme [15] with arbitrary attributes and

security against adaptively chosen ciphertext attacks in the standard. They achieved this goal with a novel application of Chameleon hash functions. Yu *et al.* [29] used the ABE to protect the security of data stored in clouds, then flexible access control over outsourced data is achieved. Lai *et al.* [11] presented an ABE scheme with partially hidden access structure to protect the access policies' privacy. To adapt for multi-authority scenario, where each authority may manage the attributes in its domain, Yang and Jia [27] proposed a multi-authority access control scheme for cloud storage to reduce the dependance on a single authority. Recently, Deng *et al.* [7] presented a novel ABE system allowing hierarchical key delegation and efficient data sharing among large organizations.

A challenging task in access control of sensitive data is to trace access credential leakage. Boneh and Naor presented a paradigm [3] to equip the public key cryptosystems with tracing mechanism by using fingerprint codes. To find the users who leaked their access credentials in broadcasting cryptosystems, Boneh *et al.* ([5, 4]) constructed two tracing schemes built from the composite-order bilinear groups, which are less efficient than in prime-order bilinear groups. Garg *et al.* [8] transmitted Boneh *et al.*'s tracing schemes to being constructed in prime-order bilinear groups to achieve better system performance in terms of encryption and decryption time. Wu and Deng [26] enhanced Boneh *et al.*'s schemes by considering the denial of tracing and farming attacks and proposed a countermeasure on the framing attack.

A few recent efforts have been made to trace access credential leakage since the employment of ABE for access control in clouds. Wang *et al.* [25] proposed the attribute-based traitor tracing system while the allowed access policy is not expressive. The systems in [28, 14, 12] support expressive policy, although the traceability is not collusion-resistant, that is, the attacker is not allowed to have more than one credential when building the illegal access devices. Liu *et al.* [16] proposed traceable ABE schemes allowing more than one access credentials used in forging an illegal access device, while the tracing capability is weak, i.e., white-box tracing. The white-box model can only capture weak attacks in which the dishonest user directly discloses his access credential. Liu *et al.* [17] suggested a method to construct the black-box traceable CP-ABE from the Boneh *et al.*'s schemes ([5, 4]). Their schemes require inefficient operations in composite-order bilinear groups and have ciphertexts sub-linear with the number of total users. Liu *et al.* [18] also proposed a black-box traceable CP-ABE with full security, although it still requires that the size of ciphertext grows sub-linearly with the number of users in the system. Recently, Deng *et al.* [6] achieved a very efficient trace-and-then-revoke mechanism of illegal access credentials distribution in cloud storage systems. The encryption procedure of their scheme requires to explicitly knows the identities of the ones who may later access the encrypted data.

Data sharing calls for efficient and reliable mechanisms to protect the data security. All the aforementioned works protect data security for different application scenarios and most of them achieve fine-grained access control due to the employment of ABE. In the face of access credentials leakage issue in ABE-based systems, the above countermeasures either allow less expressive access policy, or only withstand weak attacks, or incur heavy burdens. Our LACT scheme overcome these drawbacks in that it supports any access structure admitting a linear secret sharing and provides traceability in a

black-box manner which is a stronger security notion than white-box manner. Besides, it is built on prime-order bilinear groups and the computation operations thus are more efficient than in composite-order bilinear groups. Particularly, the LACT scheme incurs almost no extra costs for the most frequent procedures of data outsourcing and access. These advantages make the LACT scheme a very practical and secure solution to enforce a fine-grained access control over outsourced data and a trace mechanism to find out leaked access credentials in cloud storage systems.

1.3 Paper Organization

The rest of this paper is organized as follows. Section 2 presents the LACT framework and a threat model. We present the LACT scheme in Section 3. In Section 4, the security of the proposal is formally analyzed. We conduct detailed performance analysis of the LACT system in Section 5. Section 6 concludes the paper.

2 System Model and Security

2.1 System Architecture

We consider a LACT framework for cloud storage, as depicted in Fig.1. There are four types of entities in the system: the cloud service provider, the trusted authority (TA), the data owner and the data consumer. The cloud service provider (CSP) stores the outsourced data from the data owner and responds to the data access requests from the data consumer. The trusted authority is the key party trusted by other entities to generate system parameters and issue access credentials (i.e., decryption keys) for data consumers. Receiving a data owner's forensics request, the TA executes the digital forensics procedure and returns the forensic results to the data owner. The data owners define access policies and encrypt their data with the access policies before outsourcing them to the clouds. The data consumers are the cloud users who download the data owners' data from the cloud server and then decrypt them.

In our system, neither data owners nor data consumers are required to always keep online. The CSP and the TA are always online. We assume that the TA always correctly responds to the digital forensics requests and honestly returns the results.

2.2 Security Model

Unauthorized users and intruders may try to get the users' data that are outside their access privileges. We assume that the CSP is honest-but-curious in the sense that it is curious about the content of the encrypted data but still honestly execute the tasks assigned by data owners. To protect the security of the data stored in the clouds from unauthorized access, cloud users need to encrypt their data before outsourcing them to the clouds. Therefore, an encryption mechanism is preferable to make the stored data unreadable to any unauthorized users and curious CSP.

A single encryption mechanism is not sufficient to protect data privacy. In practice, there is an unavoidable problem that some access credentials may be leaked to unauthorized users, e.g., some access devices containing access credentials may be stolen

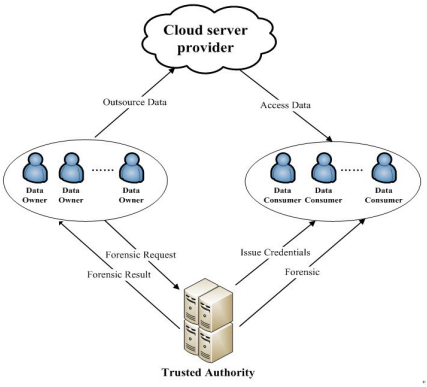

Fig. 1. System architecture

by the unauthorized users, or some users deliberately disclose their access credentials to others for some benefits. For instance, some employees of a company could sell their access rights to the sensitive data stored in clouds to the company's competitors due to economic interests. To avoid being traced, they could probably forge an illegal access credential and sell it in a black market. The misbehavior competitors then can buy the illegal access credential for unauthorized access to the company's sensitive data. In this case, we make a minimum assumption that the data owner (i.e., the sacrifice company) can find that its sensitive data were abnormally accessed, e.g., receiving alarms from the clouds that its data were accessed by some requestors with IP addresses out of the domain of the company's IP addresses, or the company find a decryption device able to access its stored data appearing at some public network market (e.g., eBay). To fulfill this assumption, we can enforce in the clouds an independent regulatory mechanism which monitors the access of the stored data and alarms the data owners in case of abnormal access.

To simplify the discussion about unauthorized access, we suppose that there exists a *pirate decoder* PD, which works in a black-box manner and enables unauthorized users to access to the stored data. The notion of black-box here means that one can access the stored data by using PD without knowing the internal construction of PD. This captures the realistic situation that the attacker may exploit technologies to conceal which access credentials are involved in creating the pirate decoder. To find out the users who leaked their access credentials, a tracing procedure is required. The tracing procedure should be allowed to access the PD and executed in a passive way, which means that it only needs to record and analyze the outputs of the PD on indistinguishable inputs. The formal definition for the security of LACT will be described in Section 4.

3 Our Solution

In this section, we propose our LACT scheme in cloud storage systems. Before presenting our scheme, we first review some basic concepts and technologies underlying our construction.

3.1 Preliminaries

Bilinear Maps. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p and g be a generator of $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map with the following properties:

i) Bilinearity: for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$;
ii) Non-degeneracy: $e(g, g) \neq 1$;
iii) Computability: there is an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

Fingerprint Codes. Following [3], we give the definition of collusion-resistant fingerprint codes.

- Let $\omega \in \{0,1\}^L$ denote a L-bit codeword. We write $\omega = \omega_1\omega_2 \cdots \omega_L$ and ω_i is the i-th bit of ω.
- Let $\mathbb{W} = \{\omega^{(1)}, \omega^{(2)}, ..., \omega^{(t)}\}$ be a set of codewords in $\{0,1\}^L$. We say that a codeword $\omega^* \in \{0,1\}^L$ is *feasible* for $\mathbb{W}$ if for all $i = 1, 2, ..., L$ there exists a $j \in \{1, 2, ..., t\}$ such that $\omega_i^* = \omega_i^{(j)}$.
- Let $F(\mathbb{W})$ be a feasible set of $\mathbb{W}$, if it includes all the codewords that are feasible for $\mathbb{W}$.

A t-collusion resistant fingerprint codes scheme is composed of generation algorithm $\mathbf{Gen}_{FC}$ and tracing algorithm $\mathbf{Tra}_{FC}$. The algorithm $\mathbf{Gen}_{FC}$ generates a set Γ of N L-bit codewords. Then each user will be assigned to a unique codeword. The t-collusion traceability guarantees that if the number of colluders is no greater than t, i.e., $|\mathbb{W}| \leq t$, the algorithm $\mathbf{Tra}_{FC}$ which takes in a feasible codeword $\omega^* \in F(\mathbb{W})$ can output at least one codeword in $\mathbb{W}$, provided that $\mathbb{W} \subseteq \Gamma$.

We will exploit the fingerprint codes scheme of [19] which is an improvement of the well-studied Tardos fingerprint codes [22]. To provide ϵ-security against t colluders in N users, the length L of this fingerprint codes is required to be no less than $-\frac{1}{\log T(t)}\left(\log \frac{N}{\epsilon} + \log \frac{c}{c-1} + \log\log \frac{c}{\epsilon}\right)$, where $T(t) < 1$ is parameterized by t, a fixed $c > 1$ and ϵ denotes the probability that one innocent has been accused.

Access Structure and LSSS [2]. In the following, we review the formal definitions for access structures and LSSS, which are extensively used in ABE schemes [9, 24, 16–18] and will still be adopted in our proposal.

Definition 1. *Let $\{P_1, P_2, \cdots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}}$ is monotone if for $\forall B, C$, we have that $C \in \mathbb{A}$ holds if $B \in \mathbb{A}$ and $B \subseteq C$. An access structure (respectively, monotone access structure) is a collection (respectively, monotone collection) $\mathbb{A}$ of non-empty subsets of $\{P_1, P_2, ..., P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1, P_2, \cdots, P_n\}} \backslash \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.*

In our LACT scheme, the role of the parties is played by the attributes. Then an access structure is a collection of sets of attributes. The monotone access states that, if a user's attribute set S satisfies an access structure $\mathbb{A}$, i.e., $S \in \mathbb{A}$, then another user

associated a larger attribute set $S' \supseteq S$ also satisfies the access structure, i.e., $S' \in \mathbb{A}$. Note that in most applications, the access policy has this monotone feature. Hence, we will only consider the monotone access structures in this paper.

Definition 2. *A secret-sharing scheme Π over a set of parties $\mathcal{P}$ is called linear (over $\mathbb{Z}_p$) if*

1. *The shares for each party form a vector over $\mathbb{Z}_p$.*
2. *There exists a matrix $\mathbf{A}$ called the share-generating matrix for Π, where $\mathbf{A}$ has l rows and n columns. For all $i = 1, \cdots, l$, the i-th row of $\mathbf{A}$ is labeled by a party $\rho(i)$, where ρ is a function from $\{1, \cdots, l\}$ to $\mathcal{P}$. We consider the column vector $\boldsymbol{s} = (s, s_2, \cdots, s_n)$, where $s \in \mathbb{Z}_p$ is the secret to be shared, and $s_2, \cdots, s_n \in \mathbb{Z}_p$ are randomly chosen. Then $\lambda_i = A_i \boldsymbol{s}$ is the share belonging to party $\rho(i)$, where A_i is the i-th row of $\mathbf{A}$.*

In practice, an LSSS scheme is employed to realize an access structure in an ABE scheme. For an access structure $\mathbb{A}$, it generates a share-generating matrix $\mathbf{A}$ with l rows and n columns and define a function that map each row of the matrix to an attribute involved in $\mathbb{A}$. Then for a secret s to be shared, the LSSS scheme forms an n-dimension vector with the first entry equal to s and rests randomly picked. It then computes the inner product of this vector and each row of the matrix, and takes the product as the share for the attribute associated with that row. The following linear reconstruction property guarantees that an LSSS scheme for an access structure $\mathbb{A}$ can recover the secret s if there exists a set S composed by some attribute associated with the rows of $\mathbf{A}$, satisfying that $S \in \mathbb{A}$.

Linear Reconstruction. It has been shown in [2] that every LSSS Π enjoys the linear reconstruction property. Suppose Π is the LSSS for access structure $\mathbb{A}$ and S is an authorized set in $\mathbb{A}$, i.e., $\mathbb{A}$ contains S. There exist constants $\{w_i \in \mathbb{Z}_p\}$ which can be found in time polynomial in the size of the share-generating matrix $\mathbf{A}$ such that if $\{\lambda_i\}$ are valid shares of s, then $\sum_{i \in I} w_i \lambda_i = s$, where $I = \{i : \rho(i) \in S\} \subseteq \{1, \cdots, l\}$.

3.2 The LACT Scheme

We first provide a high-level view of our LACT construction. To fulfill fine-grained access control over the data stored in cloud, we apply the Ciphertext-Policy Attribute-based Encryption (CP-ABE) in [24]. The data owners encrypt their files with some access policies they specified and upload the encrypted files to the CSP. Authorized data consumers will obtain access credentials, serving as decryption keys, issued by the trusted authority. A data consumer can access a file stored in clouds only if his/her associated attribute set satisfies the access policy specified in the file. In practice, some users' access credentials may be stolen or leaked. These leaked credentials might be used to forge an illegal functional access credential. To address this problem, by exploiting the tracing technology of [3], we label each user with a distinct fingerprint such that each functional access credential corresponds to a feasible codeword. The tracing procedure first finds the feasible codeword associated with the illegal access credential used by PD and then takes this feasible codeword as the input of the tracing algorithm

of the underlying fingerprint codes scheme. Then the tracing algorithm will output at least one of the codewords of the access credentials used in forging the illegal functional access credential.

The LACT scheme works as follows. First, the system is set up by the TA to generate and publish public parameters. For any user qualified to join the system, the TA generates a user credential (i.e., decryption key) with the set of attributes describing the user and embeds a fingerprint codeword into the credential. Before outsourcing data to the clouds, the data owner encrypts the data with an access structure so that only the users with attribute sets meeting this access structure can access the data. At some point, some user credentials are leaked and used to create a pirate decoder PD. This PD then can be sold on network market such as eBay for anyone interested in the sensitive data that can be decrypted by PD. Once this pirate decoder is found and reported to the data owner, the TA can be called to execute the digital forensics procedure to find out at least one of the user credentials in creating PD. Specifically, our LACT scheme consists of the following procedures.

System Setup: In this procedure, the TA setups the system. It runs the following algorithm to generate system public parameter PP, a set of fingerprint codewords Γ and system master secret key MSK. It keeps MSK secret while publishes PP for other entities.

$(PP, MSK) \leftarrow \mathrm{Setup}$: This algorithm selects a bilinear group $\mathbb{G}$ of prime order p. It chooses a random generator $g \in \mathbb{G}$ and two random elements $\alpha, \gamma \in \mathbb{Z}_p$. It chooses a hash function $H : \{0,1\}^* \rightarrow \mathbb{G}$ that will be modeled as a random oracle in the security proof. By calling the generation algorithm $\mathbf{Gen}_{FC}$ on inputs N and L, this algorithm also generates a set of codewords $\Gamma = \{\omega^{(1)}, ..., \omega^{(N)}\}$, where N denotes the maximum number of cloud users in this system and L denotes the length of each codeword. The public parameter and master secret key are set as

$$PP = (g,\ g^{\gamma},\ e(g,g)^{\alpha}, H)\,,\quad MSK = g^{\alpha}.$$

User Admission: In this procedure, a user requests to join the system. The TA checks whether the requestor is qualified, if so, it works as follows. First, the TA randomly selects a codeword $\omega \in \Gamma$ and specifies an attribute set S which describes the requestor. Then the TA generates a user credential UC, serving as a decryption key, for the requesting user by calling the following credential generation algorithm.

$UC \leftarrow \mathrm{CreGen}(MSK, S, \omega)$: This algorithm takes as inputs MSK, an attribute set S and a codeword ω. Recall that $\omega = \omega_1 \cdots \omega_L$. First, for each attribute $x \in S$, this algorithm uses the hash function H to compute $H(x||j||\omega_j)$, where $j = 1, 2, ..., L$ and the symbol "$||$" represents the operation of concatenation. Next, the algorithm picks a random exponent $r \in \mathbb{Z}_p$ and computes:

$$K_0 = g^{\alpha} g^{\gamma \cdot r}, \quad K_1 = g^{r},$$

$$\{D_{x,j} = H(x||j||\omega_j)^{r}\}_{\forall x \in S, j=1,...,L}.$$

The access credential of this user associated with S and ω is set as (including S)

$$UC = (K_0, K_1, \{D_{x,j}\}_{\forall x \in S, j=1,...,L})\,.$$

Here, the codeword ω is embedded in the user credential and distinctly associated with the user. Then in the tracing procedure, tracing a user is identical to tracing the user's codeword.

File Creation: Before outsourcing data to the CSP, the data owner encrypts his/her data as follows. First, the data owner encrypts the data using a symmetric session key $M \overset{R}{\leftarrow} \mathbb{G}_T$ of a symmetric encryption, e.g., AES. The encrypted data under the symmetric encryption forms the body of the file stored in clouds. Second, the data owner specifies an access structure $\mathbb{A}$ over a group of attributes which indicate what attributes the potential decryptors should possess. Then the data owner encapsulates the key M with the $\mathbb{A}$ (that is represented by an LSSS $(\mathbf{A}, \rho)$) by calling the following algorithm. The ciphertexts Hdr output by this algorithm is the header of the file stored in clouds.

$Hdr \leftarrow \mathrm{Encapsulate}(PP, M, (\mathbf{A}, \rho))$: This algorithm takes as inputs PP, an LSSS $(\mathbf{A}, \rho)$ and an element $M \in \mathbb{G}_T$, where M is the symmetric session key used in the symmetric encryption. To enable traceability in the system, the algorithm picks a random $j \in \{1, 2, ..., L\}$ and runs the following algorithm twice on input $b = 0$ and $b = 1$, respectively.

$Hdr_{j,b} \leftarrow \mathrm{Enc}'(PP, M, (\mathbf{A}, \rho), (j, b))$: In the LSSS $(\mathbf{A}, \rho)$, ρ is a function mapping each row of the matrix $\mathbf{A}$ to an attribute. In this construction, we limit ρ to be an injection function, that is, an attribute is associated with at most one row of $\mathbf{A}$ (note that this requirement can be relaxed to allow multiple use of one attribute by taking multiple copies of each attribute in the system, similar with the work [24]). Let $\mathbf{A}$ be an $l \times n$ matrix. This algorithm first chooses a random vector

$$\boldsymbol{v} = (s, v_2, ..., v_n) \in \mathbb{Z}_p^n$$

where s is the secret exponent needed to be shared by all involved attributes. For each i from 1 to l, the algorithm computes the hash value $H(\rho(i)||j||b)$ for attribute $\rho(i)$ and calculates the share $\lambda_i = A_i \cdot \boldsymbol{v}$, where A_i is the vector corresponding to the i-th row of $\mathbf{A}$. It then computes

$$C = Me(g, g)^{\alpha s}, \quad C_0 = g^s$$

and

$$C_i = g^{\gamma \cdot \lambda_i} H(\rho(i)||j||b)^{-s}$$

for each $i = 1, ..., l$. The algorithm outputs

$$Hdr_{j,b} = \left(C, C_0, \{C_i\}_{i=1}^{l}\right).$$

After running twice Enc' respectively on input $(j, 0)$ and $(j, 1)$, the algorithm $\mathrm{Encapsulate}$ obtains $Hdr_{j,0}$ and $Hdr_{j,1}$. It finally outputs

$$Hdr = (j, Hdr_{j,0}, Hdr_{j,1}).$$

File Access: In this procedure, a data consumer requests a file stored in CSP. CSP gives the requested file to the data consumer. Then, the data consumer decapsulates the file's header to recover the symmetric session key by calling the following algorithm and then uses this key to decrypt the file's body.

$M/\perp \leftarrow \text{Decapsulate}(Hdr, UC)$: This algorithm takes as inputs the file's header $Hdr = (j, Hdr_{j,0}, Hdr_{j,1})$ associated with LSSS $(\mathbf{A}, \rho)$ and the data consumer's credential UC associated with attribute set S. If S does not satisfy the access structure, this algorithm returns a false symbol $\perp$. If S satisfies the access structure, i.e., $S \in \mathbb{A}$, due to the the linear reconstruction property of LSSS, the algorithm can find constants $w_i \in \mathbb{Z}_p$ such that

$$\sum_{i \in I} w_i \lambda_i = s,$$

where $I = \{i : \rho(i) \in S\} \subseteq \{1, 2, ..., l\}$. This algorithm only needs to know $\mathbf{A}$ and I to determine these constants.

Recall that ω is associated with the credential UC. If the j-bit $\omega_j = 0$, the algorithm picks $Hdr_{j,0}$ (otherwise, chooses $Hdr_{j,1}$) and computes

$$\begin{aligned} M' &= \frac{e(C_0, K_0)}{\prod_{\rho(i) \in S} \Big(e(C_i, K_1) \cdot e(C_0, D_{\rho(i),j}) \Big)^{w_i}} \\ &= \frac{e(g^s, g^\alpha) e(g^s, g^{\gamma r})}{e(g^\gamma, g^r)^{\sum_{\rho(i) \in S} w_i \lambda_i}} = e(g, g)^{\alpha s}. \end{aligned}$$

It recovers M as $M = C/M'$.

In the above decapsulation algorithm, if the user's codeword has the value 0 at the j-th position, then he can only decapsulate the header $Hdr_{j,0}$ since his credential only has the component $H(\rho(i)||j||0)$ which is required to cancel out the same blind factor in $Hdr_{j,0}$; otherwise, he can only decapsulate $Hdr_{j,1}$ in the same way. This is the key point in the execution of the tracing procedure.

Digital Forensics: In this procedure, when a data owner finds that his/her file stored in clouds can be accessed by a pirate decoder PD, he can request the TA to find out the misbehavior users who have leaked their access credentials in forging the PD. Recall that in the file creation procedure all files stored in clouds are encrypted by symmetric keys which were encapsulated with specific access policies, thus the data owner can identify the access policy associated with his file that was illegally accessed. Note that a PD could possess accessability to different stored files, which means it holds the attribute sets that satisfy different access policies of these files. However, as for this data owner, he/she may only care about the access policy he/she specified for the file accessed by PD. Given this access policy, denoted by $\mathbb{A}_{PD}$, as well as PD, the data owner then can ask the TA to proceed a forensics procedure.

When we consider the pirate decoder PD, it is possible that it correctly decrypts a file with a probability less than 1. This issue has been extensively studied in [3]. In this paper, to simplify the discussion about the tracing procedure, we assume that PD can correctly decrypt a file with probability 1.

Upon a request for digital forensics, the TA responds by calling the following algorithm.

$\mathbb{C} \leftarrow \text{Find}(PP, PD, \mathbb{A}_{PD})$: This algorithm takes as inputs PP, the pirate decoder PD and the access structure $\mathbb{A}_{PD}$ satisfied by an attribute set involved in PD. It generates an LSSS $(\mathbf{A}, \rho)_{PD}$ for $\mathbb{A}_{PD}$ and works in two steps to find out the users who leaked their access credentials in forging PD.

The first step is to find the feasible codeword ω^* associated with the illegal access credential used by PD to access the data owner's file. This algorithm chooses each j from 1 to L and conducts the following experiment:

1. Choose two distinct random symmetric keys $M_j \neq M'_j \in \mathbb{G}_T$.
2. Compute the header

$$Hdr_{j,0} \leftarrow \mathrm{Enc}'(PP, M_j, (\mathbf{A}, \rho)_{PD}, (j, 0)),$$

$$Hdr'_{j,1} \leftarrow \mathrm{Enc}'(PP, M'_j, (\mathbf{A}, \rho)_{PD}, (j, 1)).$$

3. Take the header $Hdr' = (j, Hdr_{j,0}, Hdr'_{j,1})$ as the input of PD and define PD's output as M^*_j. If $M^*_j = M_j$, set $\omega^*_j = 0$; otherwise, set $\omega^*_j = 1$.

Finally, the algorithm defines $\omega^* = \omega^*_1 \omega^*_2 \cdots \omega^*_L$.

The second step is to find out the users involved in leaking their access credentials. This algorithm runs the tracing algorithm $\mathbf{Gen}_{FC}$ of the underlying fingerprint codes scheme on input ω^* to obtain a set $\mathbb{C} \subseteq \{1, ..., N\}$.

The TA returns the set $\mathbb{C}$, as the set of the users who are accused of leaking their access credentials, to the data owner.

4 Security Analysis

In this section, we formally analyze the security of our LACT scheme. At a high level, we show that the LACT scheme is secure against any number of unauthorized accesses colluding with CSP. We also demonstrates that when some users leaked their access credentials to forge an illegal access credential, which was then used to access the files stored in clouds, our LACT scheme can find out at least one of these users with a high probability. Formally, the security of the LACT scheme is defined by the following *SS-Game* and *T-Game*.

the security of LACT is composed of semantical security and traceability. The semantic security states that without the access credential, no one can get any useful information about the file outsourced by the data owner. The traceability demonstrates that if one uses an unauthorized access credential to access the file stored in clouds, TA can find out at least one of the access credentials involved in forging the unauthorized one.

SS-Game: To capture the unauthorized access to a file, we define an adversary which can query for any access credential except the authorized one that is able to access that file. We also define a challenger responsible for simulating the system procedures to interact with the adversary. In this game, the adversary is able to choose an access structure $\mathbb{A}^*$ to be challenged and ask for any user's credential for a set S of attributes on the condition that S does not satisfy $\mathbb{A}^*$. This game is formally defined as follows.

Init: The adversary $\mathcal{A}$ outputs the access structure $\mathbb{A}^*$ to be challenged.

Setup: The challenger runs the setup algorithm and gives the system public parameter, PP to the adversary $\mathcal{A}$.

Phase 1: The adversary $\mathcal{A}$ queries the challenger for user credentials corresponding to attribute sets $S_1, S_2, ..., S_{q_1}$,

Challenge: The adversary $\mathcal{A}$ outputs two equal-length messages M_0 and M_1 and an access structure $\mathbb{A}^*$. The restriction is that $\mathbb{A}^*$ can not be satisfied by any of the queried attribute sets in phase 1. The challenger flips a coin $\beta \in \{0, 1\}$, and encapsulates M_β with $\mathbb{A}^*$, producing header Hdr^*. It then returns Hdr^* to $\mathcal{A}$.

Phase 2: The adversary $\mathcal{A}$ queries the challenger for user credentials corresponding to attribute sets $S_{q_1+1}, ..., S_q$, with the added restriction that none of these sets satisfies $\mathbb{A}^*$.

Guess: The attacker outputs a guess $\beta' \in \{0, 1\}$.

The advantage of the adversary $\mathcal{A}$ in this game is defined as $Adv_{\mathcal{A}}^{SS} = |\Pr[\beta = \beta'] - 1/2|$.

Definition 3. *Our LACT scheme is semantically secure if all polynomial-time adversaries have at most negligible advantages in the above game.*

In this semantic security, the adversary is able to query for users' access credentials, which means it can collude with any user, as well as the CSP. When it is challenged, there is a requirement that it cannot trivially win the challenge. The semantic security states that given any user's credential, there is no polynomial time adversary which can distinguish the encapsulations of two symmetric keys, provided that it does not have the access credential able to decapsulate any of these encapsulations.

T-Game: In this game, we define an adversary which can collude with users by querying their access credentials. The adversary can use some or all of the queried access credentials to forge a pirate decoder PD. The adversary outputs the PD as a challenge and terminates the credential queries. This game is formally defined as follows.

Setup: The challenger runs the setup algorithm and gives the system public parameter PP to the adversary $\mathcal{A}$.

Query: The adversary adaptively makes credential queries for attribute sets. In response, the challenger runs the credential generation algorithm and gives the queried credentials to $\mathcal{A}$.

Challenge: The adversary $\mathcal{A}$ stops the credential queries and gives the challenger a pirate decoder PD able to access the file associated with access structure $\mathbb{A}_{PD}$.

Trace: The challenger runs the tracing algorithm on inputs PD and $\mathbb{A}_{PD}$ to obtain the set $\mathbb{C} \subseteq \{1, ..., N\}$. Let $\mathbb{S}$ denote the set of users whose access credentials have been queried by $\mathcal{A}$. We say that the adversary $\mathcal{A}$ wins in this game if:

1. The set $\mathbb{C}$ is empty, or not a subset of $\mathbb{S}$.
2. The pirate decoder can decapsulate any valid header with probability 1.
3. There are at most t credential queries for the attribute sets which can satisfy the access structure $\mathbb{A}_{PD}$.

We briefly explain the correctness of the three conditions above. The first condition is straightforward and the second condition is required by the assumption about

PD discussed in the digital forensics procedure. The third condition is implied by the underlying fingerprint codes scheme. The pirate decoder was created by various credentials for attribute sets, some of which satisfy the access structure $\mathbb{A}_{PD}$. The underlying fingerprint codes scheme that is secure against t-collusion attack restricts that there are at most t queried credentials that can be used to directly decapsulate the header associated with $\mathbb{A}_{PD}$. Then in the traceability definition above, it is also required that at most t credentials associated with the attribute sets satisfying $\mathbb{A}_{PD}$ can be queried by the adversary. Specially, when $t = N$, our scheme is fully resistant.

We define the advantage of the adversary $\mathcal{A}$ in winning in this game as $Adv^T_{\mathcal{A}} = |\Pr[\mathcal{A} \text{ wins}]|$.

Definition 4. *A LACT scheme is t-collusion resistant if all polynomial-time adversaries have at most negligible advantages in both* SS-Game *and* T-Game*.*

The following theorem claims the semantic security and traceability of the LACT scheme. The proof of this theorem is given in the full version.

Theorem 1. *Our LACT scheme is semantically secure if the underlying CP-ABE scheme [24] is secure. It is also t-collusion resistant with the additional condition that the underlying fingerprint codes scheme [19] is t-collusion resistant, where t is the maximum number of colluders. In particular, let $\mathcal{M}$ denote the message space, L denote the length of fingerprint codes and ϵ denote the probability that one innocent has been accused, then any polynomial-time adversary breaks the LACT system with the advantage at most*

$$Adv^T_{\mathcal{A}} \leq L \cdot Adv^{SS}_{\mathcal{A}} + \epsilon + \frac{L}{|\mathcal{M}|}.$$

The LACT scheme is semantically secure if the underlying CP-ABE scheme is secure. Hence, the advantage $Adv^{SS}_{\mathcal{A}}$ is negligible. In the t-collusion resistant fingerprint codes scheme, the error probability ϵ is negligible too. Moreover, since the size of message space $\mathcal{M}$ is much larger than the codes length L, then $L/|\mathcal{M}|$ is very close to 0. Hence, the advantage $Adv^T_{\mathcal{A}}$ of the adversary in breaking the traceability of our LACT scheme is negligible, which means that the LACT scheme is t-collusion resistant.

5 Performance Analysis

In this section, we analyze the computation cost of each procedure of the LACT scheme. We view the underlying fingerprint codes scheme as a black-box. We use $O(\textbf{Gen})_{FC}$ and $O(\textbf{Tra})_{FC}$ to denote the computation complexity of the generation algorithm and tracing algorithm of the fingerprint codes respectively. Tardos proposed a fingerprint codes scheme which is a milestone in this area and has been well studied, while the codes length is a bit long. Nuida et.al's fingerprint codes [19] achieves a shorter length, about 1/20 of Tardos codes for the same security level. Thus we suggest Nuida et.al's codes to be used. Their fingerprint codes scheme has the length

$$L \geq -\frac{1}{\log T(t)} \left(\log \frac{N}{\epsilon} + \log \frac{c}{c-1} + \log \log \frac{c}{\epsilon} \right) \tag{1}$$

where $T(t)$ is a function of t and valued in $(0, 1)$, c an auxiliary variable larger than 1, N the number of users and ϵ is the error probability of tracing an innocent user.

Our LACT scheme is built on the bilinear groups $\mathbb{G}$ and $\mathbb{G}_T$. We evaluate the time consumed by the basic groups operations, i.e., exponentiation and bilinear pairing map. Although the multiplication operation is also involved, its cost time is negligible compared to the former two operations. We use τ_e and τ_p to denote the time consumed by exponentiation and bilinear pairing map, respectively, without discriminating exponentiation operations in $\mathbb{G}$ and $\mathbb{G}_T$.

Table 1 gives the computation cost of each procedure in our LACT scheme. In this table, L denotes the length of the fingerprint codes, $|S|$ the number of attributes of the set associated with a credential, l the number of attributes involved in the access structure associate with a file and $|S^*|$ the number of attributes of the set S^* which satisfies the access structure. From Table 1, we can see that only the user admission procedure and the digital forensics procedure are affected by the introduction of tracing access credential leakage. Specially, adding the traceability functionality does not affect the file creation and file access procedures. This is a desirable property in practice because the most frequent operations in cloud environments are uploading and downloading files while the user admission operation is only carried out once by the TA.

Table 2 compares our LACT scheme with other similar works in terms of public and secret key size, ciphertext size, the number of pairings required in decryption, fine-grained access control (supported or not), tracing capability (black-box tracing or not) and the type of based bilinear groups (composite order or prime order). In this table, the schemes of [4] and [5] are proposed for the broadcast encryption systems, where the number of total users is denoted by N. The two schemes are built from composite order groups, thus the efficiency of cryptographic operations is much lower (about one-order) than in prime order groups. The scheme of [6] is devised for the identity-based broadcast encryption which assumes the maximal size of the set with each member as a decryptor to be m. This scheme achieves a constant-size ciphertext at the expense of secret key size linear with the length L of codes. The schemes [4–6] all provides black-box tracing, but due to the property of broadcast encryption, they do not support the fine-grained access control. The rest schemes in the table are proposed in the attribute-based encryption systems and support fine-grained access control. The schemes of [16–18], as well as ours, require that the number of pairings in a decryption is linear with the cardinality of the set S^* satisfying the access policy of a ciphertext, which is inevitable in almost all ABE schemes. In the scheme of [16], the size of public key is linear with the size of attribute universe $\mathcal{U}$ and the size of secret key is linear with the size of the attribute set S, while this scheme only provides the weak white-box tracing. The schemes in [17] and [18] achieve black-box tracing, while the size of public key grows linearly with the attribute universe size $|\mathcal{U}|$ and the quadratic root of the number of total users. Besides, the ciphertexts of these two schemes are sub-linear with the number of the users, which results larger amount of data to be uploaded/stored and more bandwidth consumption for the communication between the cloud server and cloud clients.

Table 2 also reveals the better practicality for the LACT scheme to be employed in cloud storage systems. Compared to the schemes [4–6], our scheme achieve the fine-grained access control but still retains the black-box tracing capability. Compared to

Table 1. Computation

Operation	Computation Cost
System Setup	$1\tau_p + 2\tau_e + O(\mathbf{Gen}_{FC})$
User Admission	$(L \cdot \|S\| + 2)\tau_e$
File Creation	$4(l+1)\tau_e$
File Access	$(2\|S^*\| + 1)\tau_p + \|S^*\|\tau_e$
Digital Forensics	$4L(l+1)\tau_e + O(\mathbf{Tra}_{FC})$

Table 2. Comparison with related works

	Public key size	Private key size	Ciphertext size	$e(\cdot,\cdot)$ in decryption	Fine-grained control	Black-box tracing	Prime-order groups
[4]	$3+4\sqrt{N}$	1	$6\sqrt{N}$	3	$\times$	$\surd$	$\times$
[5]	$5+9\sqrt{N}$	$1+\sqrt{N}$	$7\sqrt{N}$	4	$\times$	$\surd$	$\times$
[6]	$3+m$	L	6	2	$\times$	$\surd$	$\surd$
[16]	$\|\mathcal{U}\|+4$	$\|S\|+4$	$2l+3$	$2\|S^*\|+1$	$\surd$	$\times$	$\times$
[17]	$\|\mathcal{U}\|+7+8\sqrt{N}$	$\|S\|+3$	$2l+8\sqrt{N}$	$2\|S^*\|+5$	$\surd$	$\surd$	$\times$
[18]	$\|\mathcal{U}\|+3+4\sqrt{N}$	$\|S\|+4$	$2l+9\sqrt{N}$	$2\|S^*\|+6$	$\surd$	$\surd$	$\times$
Ours	3	$\|S\|L+2$	$2l+4$	$2\|S^*\|+1$	$\surd$	$\surd$	$\surd$

the similar works [17, 18], the ciphertext of the LACT scheme is independent of the number of total users in the system. In addition, the LACT scheme achieves constant-size public key. Moreover, the LACT scheme is based on prime order groups, thus the cryptographic operations are much more efficient than those of schemes [17, 18] that are built from composite order bilinear groups. Although the secret keys are linear with the product of the size $|S|$ of the attribute set and the length L of codes, this will not severely impact the system practicality since the key generation operation is run by the TA in offline phase. All these advantages render our scheme as an efficient solution to enable leaked access credentials finding mechanism in cloud storage systems.

6 Conclusion

In this paper, we investigated the access credentials leakage problem in cloud storage systems. The proposed LACT scheme not only offers fine-grained access control over outsourced data, but also provides a tracing mechanism to find the leaked access credentials. Formal proofs show the security and traceability of the LACT scheme. We also conducted detailed performance analysis on the LACT scheme and compared it with similar works. The analysis and comparisons show that our LACT scheme has enjoyable performance and provides an efficient solution to find leaked access credentials in data outsourced environments.

Acknowledgments and Disclaimer. We appreciate the anonymous reviewers for their valuable suggestions. Dr. Bo Qin is the corresponding author. This paper was supported by the National Key Basic Research Program (973 program) under project 2012CB315905, the Natural Science Foundation of China through projects 61370190, 61173154, 61003214, 60970116, 61272501, 61321064 and 61202465, the Beijing Natural Science Foundation under projects 4132056 and 4122041, the Shanghai NSF under Grant No. 12ZR1443500, the Shanghai Chen Guang Program (12CG24), the Science and Technology Commission of Shanghai Municipality under grant 13JC1403500, the Fundamental Research Funds for the Central Universities, and the Research Funds(No. 14XNLF02) of Renmin University of China, the Open Research Fund of The Academy of Satellite Application and the Open Research Fund of Beijing Key Laboratory of Trusted Computing.

References

1. Asokan, N., Dmitrienko, A., Nagy, M., Reshetova, E., Sadeghi, A.-R., Schneider, T., Stelle, S.: CrowdShare: Secure mobile resource sharing. In: Jacobson, M., Locasto, M., Mohassel, P., Safavi-Naini, R. (eds.) ACNS 2013. LNCS, vol. 7954, pp. 432–440. Springer, Heidelberg (2013)
2. Beimel, A.: Secure schemes for secret sharing and key distribution. PhD thesis, Israel Institute of Technology, Technion, Haifa, Israel (1996)
3. Boneh, D., Naor, M.: Traitor tracing with constant size ciphertext. In: ACM CCS 2008, pp. 501–510. ACM Press, New York (2008)
4. Boneh, D., Sahai, A., Waters, B.: Fully collusion resistant traitor tracing with short ciphertexts and private keys. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 573–592. Springer, Heidelberg (2006)
5. Boneh, D., Waters, B.: A fully collusion resistant broadcast, trace, and revoke system. In: ACM CCS 2006, pp. 211–220. ACM Press, New York (2006)
6. Deng, H., Wu, Q., Qin, B., Chow, S.S.M., Domingo-Ferrer, J., Shi, W.: Tracing and revoking leaked credentials: accountability in leaking sensitive outsourced data. In: ASIACCS 2014, pp. 425–434. ACM Press, New York (2014)
7. Deng, H., Wu, Q., Qin, B., Domingo-Ferrer, J., Zhang, L., Liu, J., Shi, W.: Ciphertext-policy hierarchical attribute-based encryption with short ciphertexts. Information Sciences 275, 370–384 (2014)
8. Garg, S., Kumarasubramanian, A., Sahai, A., Waters, B.: Building efficient fully collusion-resilient traitor tracing and revocation schemes. In: ACM CCS 2010, pp. 121–130. ACM Press, New York (2010)
9. Goyal, V., Pandey, O., Sahai, A., Waters, B.: Attribute-based encryption for fine-grained access control of encrypted Data. In: ACM CCS 2006, pp. 89–98. ACM Press, New York (2006)
10. Huang, D., Zhou, Z., Xu, L., Xing, T., Zhong, Y.: Secure data processing framework for mobile cloud computing. In: IEEE Conferenc on Computer Communications Workshops, pp. 614–618. IEEE (2011)
11. Lai, J., Deng, R.H., Li, Y.: Expressive cp-abe with partially hidden access structures. In: ASIACCS 2012, pp. 18–19. ACM Press, New York (2012)
12. Li, J., Huang, Q., Chen, X., Chow, S.S.M., Wong, D.S., Xie, D.: Multi-authority ciphertext-policy attribute-based encryption with accountability. In: ASIACCS 2011, pp. 386–390. ACM Press, New York (2011)

13. Li, F., Rahulamathavan, Y., Rajarajan, M., Phan, R.C.W.: Low complexity multi-authority attribute based encryption scheme for mobile cloud computing. In: IEEE 7th International Symposium on Service Oriented System Engineering, pp. 573–577. IEEE (2013)
14. Li, J., Ren, K., Kim, K.: A2BE: accountable attribute-based encryption for abuse free access control. IACR Cryptology ePrint Archive, Report 2009/118 (2009), http://eprint.iacr.org/
15. Liu, W., Liu, J., Wu, Q., Qin, B., Zhou, Y.: Practical direct chosen ciphertext secure key-policy attribute-based encryption with public ciphertext test. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8713, pp. 91–108. Springer, Heidelberg (2014)
16. Liu, Z., Cao, Z.F., Wong, D.S.: White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures. IEEE Transaction on Informaction Forensics and Security 8(1), 76–88 (2013)
17. Liu, Z., Cao, Z.F., Wong, D.S.: Expressive black-box traceable ciphertext-policy attribute-based encryption. IACR Cryptology ePrint Archive, Report 2012/669 (2012), http://eprint.iacr.org/
18. Liu, Z., Cao, Z.F., Wong, D.S.: Blackbox traceable cp-abe: how to catch people leaking their keys by selling decryption devices on eBay. In: ACM CCS 2013, pp. 475–486. ACM Press, New York (2013)
19. Nuida, K., Fujitsu, S., Hagiwara, M., Kitagawa, T., Watanabe, H., Ogawa, K., Imai, H.: An improvement of discrete tardos fingerprinting codes. Designs, Codes and Cryptography 52(3), 339–362 (2009)
20. Qin, B., Wang, H., Wu, Q., Liu, J., Domingo-Ferrer, D.: Simultaneous authentication and secrecy in identity-based data upload to cloud. Cluster Computing 16(4), 845–859 (2013)
21. Singhal, M., Chandrasekhar, S., Ge, T., Sandhu, R., Krishnan, R., Ahn, G.J., Bertino, E.: Collaboration in multicloud computing environments: framework and security issues. IEEE Computer 46(2), 76–84 (2013)
22. Tardos, G.: Optimal Probabilistic Fingerprint Codes. In: STOC 2003, pp. 116–125. ACM Press, New York (2003)
23. Wang, Y., Wu, Q., Wong, D.S., Qin, B., Chow, S.S.M., Liu, Z., Tan, X.: Securely outsourcing exponentiations with single untrusted program for cloud storage. In: Kutylowski, M., Vaidya, J. (eds.) ESORICS 2014. LNCS, vol. 8712, pp. 323–340. Springer, Heidelberg (2014)
24. Waters, B.: Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization. In: Catalano, D., Fazio, N., Gennaro, R., Nicolosi, A. (eds.) PKC 2011. LNCS, vol. 6571, pp. 53–70. Springer, Heidelberg (2011)
25. Wang, Y.T., Chen, K.F., Chen, J.H.: Attribute-based traitor tracing. J. Inf. Sci. Eng. 27(1), 181–195 (2011)
26. Wu, Y., Deng, R.H.: On the security of fully collusion resistant taitor tracing schemes. IACR Cryptology ePrint Archive, Report 2008/450 (2008), http://eprint.iacr.org/
27. Yang, Y., Jia, X.: Attributed-based access control for multi-authority systems in cloud storage. In: IEEE 32nd International Conference on Distributed Computing Systems, pp. 536–545. IEEE (2012)
28. Yu, S., Ren, K., Lou, W., Li, J.: Defending against key abuse attacks in KP-ABE enabled broadcast systems. In: Chen, Y., Dimitriou, T.D., Zhou, J. (eds.) SecureComm 2009. LNICST, vol. 19, pp. 311–329. Springer, Heidelberg (2009)
29. Yu, S., Wang, C., Ren, K., Lou, W.: Achieving secure, scalable, and fine-grained data access control in cloud computing. In: 2010 Proceedings of IEEE INFOCOM, pp. 1–9. IEEE (2010)