Tracing and Revoking Leaked Credentials: Accountability in Leaking Sensitive Outsourced Data

Hua Deng
School of Computer
Wuhan University
Wuhan, P. R. China
denghua@whu.edu.cn

Qianhong Wu
Academy of Satellite
Application, Electronic &
Information Engineering
Beihang University, China
qianhong.wu@buaa.edu.cn

Bo Qin
School of Information
Renmin University of China
Beijing, P. R. China
bo.qin@ruc.edu.cn

Sherman S. M. Chow
Department of
Information Engineering
Chinese University of
Hong Kong, Hong Kong
smchow@ie.cuhk.edu.hk

Josep Domingo-Ferrer
Department of Computer
Engineering and Mathematics
Universitat
Rovira i Virgili, Catalonia
josep.domingo@urv.cat

Wenchang Shi
School of Information
Renmin University of China
Beijing, P. R. China
wenchang@ruc.edu.cn

ABSTRACT

There have been increasing security concerns on the data stored in clouds. Most existing proposals mainly aim at guaranteeing that the outsourced data are only accessible to authorized requestors who have the access credentials. This paper investigates a trace-and-then-revoke mechanism of illegal access credential distribution, which serves as an *a posteriori* approach to complement existing *a priori* solutions for securing outsourced data. Our system for tracing and revoking leaked access credentials, TRLAC, has the following desirable properties: (1) the tracing procedure can trace at least one dishonest user who illegally distributed an access credential for decryption; (2) the tracing procedure is run in a black-box manner; and (3) the tracing procedure does not heavily rely on cooperation from the cloud service provider. Once the dishonest users have been found, a revocation mechanism can be called to deprive them of access rights. We formally prove the security of TRLAC and conduct theoretical and experimental analyses. The results show that the system is secure and efficient to be employed in cloud storage, and, surprisingly, the introduction of the tracing feature incurs little outsourcing and access costs.

Categories and Subject Descriptors

E.4 [**Data**]: Public Key Cryptosystems

Keywords

Cloud computing; Data security; Access control; Credential leakage tracing

ASIA CCS'14, June 4–6, 2014, Kyoto, Japan.

http://dx.doi.org/10.1145/2590296.2590342 .

1. INTRODUCTION

In recent years, cloud computing has emerged as one of the most promising computing paradigms and attracted significant interest in both academia and industry. For example, it has already been used in many applications such as live media [33], distributed cooperative work [34], neural network learning [38], mobile health monitoring [19], etc. In the context of data storage, a user can outsource a huge amount of data to a cloud server maintained by a cloud service provider (CSP). Both the user and any authorized peers can access (and process) the outsourced data from anywhere anytime. In this way, users can obtain storage and computing resources on demand without local hardware and software maintenance burden.

Security and privacy are widely recognized to be one of the main barriers for the public to accept cloud computing ([31, 21]). Numerous existing approaches (to be reviewed in Section II) have been proposed, but most of them are *a priori* or only provide *outsider security* in the sense that only those who hold some appropriate access credentials can retrieve the data in plaintext. This is insufficient since the insiders who hold the access credentials may be hacked or may simply intentionally leak the credentials for other benefits. Unfortunately, few efforts have been made to facilitate digital forensics analysis of access credential leakage, although it is crucial to obtain electronic evidence to bring the case in a court of law [15]. Countermeasures should be adopted to trace and then revoke the compromised secret keys to minimize the exposure risks to future data. This observation motivates us to investigate an *a posteriori* approach to enhance the access control over outsourced data.

Our contribution includes the following aspects. We propose a generic framework for tracing and revoking leaked access credentials (TRLAC) to enhance access control over data outsourced to clouds, which provides a mechanism to find which credentials have been leaked. We instantiate our framework with a concrete TRLAC scheme by extending an existing identity-based broadcast encryption [11] and leveraging optimal fingerprint codes [24] to encode each authorized user's access credential (which is the decryption key for

the encryption system). In our TRLAC system, users are organized in different groups and given access credentials associated with their groups' identities and fingerprint codes. Before outsourcing data, the data owners can encrypt them by specifying a set of groups in which the members can access their data. When credential leakage has occurred, even if multiple credentials are embedded in a pirate decoder box, a tracing procedure can be run in a black-box manner (without knowing how the illegal access credential was produced) to trace at least one compromised key among those used to create the decoder box. The tracing procedure does not require the help of the CSP, who may not even know that a tracing procedure has been invoked. When compromised access credentials are found out by the tracing procedure, the owners can revoke the privileges associated to those credentials for accessing the outsourced data.

We conduct a thorough analysis of the proposed TRLAC scheme. Its security is formally reduced to the security of the collusion resistance of the underlying fingerprint codes and the semantic security (against chosen-plaintext attacks) of the identity-based broadcast encryption scheme. We also perform a comprehensive performance evaluation. Both the theoretical analysis and the experimental results confirm that our scheme can trace illegal access credentials without introducing almost any extra cost to file creation and access procedures in cloud storage. Even the cost of the user admission procedure, which is much affected by the introduction of tracing, is pretty acceptable to a powerful key distribution center. The revocation mechanism only requires few computations and we will show empirically that it is very cost-effective. These advantageous features make our scheme an outstanding solution to secure sensitive data that are being outsourced to untrusted clouds.

The rest of this paper is organized as follows. Section 2 reviews the related work in the literature. Section 3 presents our TRLAC framework and a threat model. We present our TRLAC construction in Section 4. In Section 5, the security of TRLAC is formally analyzed. We evaluate the performance of our TRLAC system both theoretically and experimentally in Section 6. Section 7 concludes the paper.

2. RELATED WORK

It is essential for the owners to validate that their data are kept intact to guarantee availability. Sebé and Domingo-Ferrer *et al.* [27] identified the requirements to remotely check integrity of the data stored in a third party. Ateniese *et al.* [1] proposed a model of provable data possession (PDP) and realized the first PDP scheme. In PDP, privacy concerns have also been studied. The auditing scheme in [32] enables a third party auditor (TPA) to simultaneously perform multiple auditing tasks, without letting the TPA know the possibility of inferring the data being audited. For enterprise uses, anonymity of the uploader also matters, and that is achieved with the help of a security-mediator in [31], without letting the mediator link the upload to the identity of the user, or learn the data being uploaded.

One way to allow only authorized users to access the service is authentication. Chow *et al.* [9] proposed SPICE, a simple privacy-preserving identity-management for cloud environment, which provided various features such as delegatable verification from one CSP to another CSP, yet not covered by previous privacy-preserving authentication schemes. Qin *et al.* [26] proposed an identity-based admission scheme which simultaneously enables access right proofs and privacy-preserving data uploads. Even though cryptographic solutions are employed, it requires trusting the server to act according to the authentication results.

The standard way to achieve data privacy is to encrypt the data before they are outsourced. The users allowed to access the digital content are given access credentials which are usually secret keys enabling decryption of the encrypted data. Some early works (e.g., [28]) adopt traditional public key encryption, such as RSA, to ensure data storage security in cloud computing, which can be cumbersome regarding key management and validation of the public keys of the receivers before encryption. To circumvent these problems, Boneh and Franklin proposed the first practical identity-based encryption (IBE) [3], in which the users' public keys are their identities, such as email addresses or telephone numbers. Waters [35] proposed the first fully-secure and efficient IBE scheme in the standard model. In the one-to-many encryption scenario, Delerablée [11] presented an identity-based broadcast encryption (IBBE) to enable a sender to encrypt his data once by specifying a set of identities so that the receivers with the specified identities can decrypt the same ciphertext. Recent works ([21, 22, 36]) have employed attribute-based encryption to provide flexible and fine-grained access to encrypted data, although this is achieved by sacrificing to some extent the efficiency of encryption and decryption.

The encryption of outsourced files should allow users to retrieve selected items without downloading and decrypting the entire files and then searching locally. This challenging requirement is usually satisfied with searchable encryption schemes. For some of the latest works with special focus on the outsourced cloud scenario, Yu *et al.* [37] presented a searchable encryption scheme to provide sufficiently accurate search; Li *et al.* [20] proposed ranked keyword searchable encryption to enhance the relevance of the returned search results; and Cao et al. [7] proposed privacy-preserving queries over encrypted graph-structured data.

All the aforementioned works only provide *a priori* approaches to secure outsourced data. While it is challenging to enforce digital forensics analysis in cloud computing due to the separation of the data ownership and its physical access control, some efforts have been made to trace leaked decryption keys in a multi-receiver scenario. To find the traitors who leaked decryption keys in broadcast encryption environments, Boneh *et al.* ([5, 6]) constructed two schemes built from composite-order bilinear groups to achieve full collusion resistance. In view of the low efficiency of composite-order bilinear groups, Garg *et al.*'s construction [12] is built in prime-order bilinear groups to achieve more efficient encryption and decryption. Liu *et al.* [16] proposed in attribute-based encryption environments a traitor tracing mechanism executed in white-box manner, which can only capture weak attacks where dishonest users directly disclose their access credentials. Subsequently, Liu *et al.* constructed black-box traceable attribute-based encryption schemes ([17, 18]) from the Boneh *et al.*'s constructions ([5, 6]). Thus their schemes also suffer from low efficiency due to the composite-order bilinear groups, and the ciphertexts are sub-linear with the number of total users in system. Boneh and Naor [4] proposed a paradigm to equip public key cryptosystems with the traitor tracing functionality by exploiting fingerprint codes. This paradigm supports shorter cipher-

texts and thus it is very suitable for cloud storage systems when it comes to saving storage space. By applying this paradigm, Guo *et al.* [14] presented a tracing mechanism with short ciphertexts and decryption keys in identity-based encryption environments.

The issue of leakage of sensitive data in cloud computing has already attracted some attention. Chow *et al.* [8] proposed a dynamic secure provenance scheme which can record the data ownership as an evidence if there is a dispute later. Tan *et al.* [30] presented an auditing methodology that enables tracking data transferred out of clouds, and provides a countermeasure in case of data leakage. Pappas *et al.* [25] proposed a CloudFence framework to provide data owners with tracking capabilities to independently audit the treatment of their data. Zhang *et al.* [39] investigated the threat of leaking data in clouds and presented rule-based data provenance tracing algorithms to detect actual operations that have been performed on files, especially those which can harm data confidentiality.

When decryption keys are compromised and then revoked, countermeasures should be taken to prevent access to the encrypted data that were previously decryptable by the revoked keys. Proxy re-encryption (PRE), first formulated by [2], enables a proxy to re-encrypt a ciphertext under Alice's public key to a ciphertext under Bob's public key, without revealing the underlying message. Green and Ateniese [13] and Chu and Tzeng [10] then extended PRE to identity-based proxy re-encryption (IB-PRE), which allows an arbitrary string to be used as the public key of a user, so that a certificate authority is not required. While providing such useful properties, PRE and IB-PRE require a proxy that cannot be fully trusted to enforce re-encryption; if the proxy is compromised, users risk their data to be exposed. The group signature scheme proposed in [23] also allows membership revocation, but it only focuses on revoking the anonymity of the signed message rather than revoking the private keys.

In contrast to the previous researches, we study tracing leakage of access credentials (equivalent to decryption keys) in cloud storage and we propose a generic framework for this purpose. We leverage optimal fingerprint codes to encode the decryption keys (assigned to the authorized users) in identity-based broadcast encryption, so that the leaked keys can be traced with the unique fingerprints. We also conduct formal security proofs of the proposed scheme and perform overhead analysis theoretically and experimentally. We aim to provide a thorough study of the effect of introducing tracing of leaked keys on storing confidential files in clouds and accessing them remotely. Our results reveal that, with the gains of enhanced security protection of the outsourced data, our *a posteriori* security mechanism incurs a surprisingly low extra cost, especially for file creation and access.

3. DEFINITIONS AND MODELS

3.1 System Model

As depicted in Fig. 1, our TRLAC framework consists of four parties which can be identified as follows:

- Cloud Storage Server (CSS): a party which provides storage service for cloud users, i.e., data owners and data consumers in clouds.

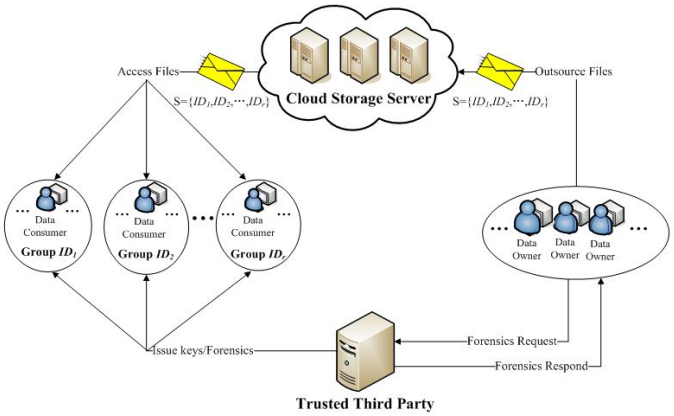

Figure 1: System Model

- Data Owners: the users who encrypt and outsource their data to CSS for sharing with data consumers.
- Data Consumers: the users who download the shared data from CSS and then decrypt them.
- Trusted Third Party (TTP): an authority who is trusted to publish the system parameters, and distribute private keys for data consumers after they pass the identity verification. Upon a data owner's request, TTP responds by executing the digital forensics procedure and returning the result to the data owner.

All cloud users are organized in distinct groups associated with different identities. One user in multiple groups will be assigned to multiple keys corresponding to different groups (identities). The data owners interact with CSS via CSP to store their data. They can specify an access policy, i.e., a set $S = \{ID_1, ID_2, \cdots, ID_r\}$, so that the data consumers in these groups can access their data. The CSS is thought to have sufficient storage space and computation resources. TTP (e.g., the IT center of a company that employs the cloud storage systems) is assumed to be unbiased in the sense that it will correctly respond to the users' digital forensics requests.

3.2 Security Model and Assumption

The cloud service provider (CSP) is assumed not to be fully trustable. It may be curious about the users' uploaded files or collude with some cloud users for accessing the files without authorization. Unauthorized users and intruders may also try to access the users' files. Therefore, a user needs to encrypt his files before outsourcing them to the CSS so that only authorized ones can decrypt with assigned secret access credentials. The problem is that some users' access credentials may be leaked, e.g., if their devices storing their access credentials were hacked. Some authorized users may also intentionally leak their credentials. For instance, an employee who is unsatisfied with his company could leak his credentials for revenge.

We make the minimum assumption that the data owners can be informed on successful illegal access to their data. In fact, similar to [18] for simplifying data leakage in the cloud, we assume that there exists a *pirate decryptor* ($\mathcal{PD}$), that is, a procedure created by the leaked secret credentials equivalently treated as the decryption keys, that allows

unauthorized parties to access the user's files. The output of $\mathcal{PD}$ is visible to the TTP. We require that a tracing procedure can find at least one of the leaked credentials used to produce $\mathcal{PD}$. The tracing procedure should be able to fulfill its task without knowledge about how $\mathcal{PD}$ is created since the attacker may exploit technologies to avoid being traced. The tracing procedure could be executed in a passive way in the sense that the TTP only needs to observe and analyze the outputs of $\mathcal{PD}$, which is given indistinguishable inputs.

4. OUR TRLAC SCHEME

We first review some basic techniques used in constructing the TRLAC scheme. We then give our construction.

4.1 Preliminaries

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

1. Bilinearity: for all $u, h \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$, it holds that $e(u^a, h^b) = e(u, h)^{ab}$;
2. Non-degeneracy: $e(g, g) \neq 1$. We say that $\mathbb{G}$ is a bilinear group if the group operation in $\mathbb{G}$ and the bilinear map e are both efficiently computable.

Fingerprint codes. We are mainly interested in binary fingerprint codes and follow their definition in [4].

- A set $\Gamma = \{\omega^{(1)}, \cdots, \omega^{(n)}\} \subseteq \{0,1\}^l$ will be called an (l, n)-code. The codeword $\omega^{(i)}$ will be assigned to the i-th user, for $1 \leq i \leq n$, where $\omega^{(i)} = \omega_1^{(i)} \cdots \omega_l^{(i)}$ and n is the number of users.
- Let set $\mathbb{W} = \{\omega^{(1)}, \cdots, \omega^{(t)}\} \subseteq \{0,1\}^l$, where $t \leq n$. A codeword $\omega^* \in \{0,1\}^l$ is called *feasible* for $\mathbb{W}$ if for all $i = 1, \cdots, l$ there is a $j \in \{1, \cdots, t\}$ such that $\omega_i^* = \omega_i^{(j)}$.
- For a set $\mathbb{W} \subseteq \{0,1\}^l$ we say that the feasible set of $\mathbb{W}$, denoted by $F(\mathbb{W})$, is the set of all codewords that are feasible for $\mathbb{W}$.

A fingerprint code scheme consists of a generation algorithm $\mathbf{Gen}_{FC}$ and a tracing algorithm $\mathbf{Tra}_{FC}$. The algorithm $\mathbf{Gen}_{FC}$ can efficiently generate a fingerprint code Γ of n l-bit codewords, with each codeword being assigned to one user. Given a feasible codeword $\omega^* \in F(\mathbb{W})$, the algorithm $\mathbf{Tra}_{FC}$ can efficiently find at least one codeword $\omega \in \mathbb{W}$, where $\mathbb{W} \subseteq \Gamma$ and $|\mathbb{W}| \leq t$. Our scheme exploits the Nuida *et al.*'s fingerprint codes [24] which is an improvement of the well-known Tardos's scheme [29]. Nuida *et al.*'s fingerprint codes achieve a shorter length, about 1/20 of Tardos's fingerprint codes, for the same security level. To provide traceability against at most t colluders among n users, Nuida *et al.*'s fingerprint codes require the codeword length to be

$$l \geq -\frac{1}{\log T(t)} \left(\log \frac{n}{\epsilon} + \log \frac{c}{c-1} + \log \log \frac{c}{\epsilon} \right), \quad (1)$$

where $T(t) < 1$ is parameterized by t, $c > 1$ is a constant and ϵ denotes the probability that an innocent user is accused.

4.2 Our Proposal

4.2.1 Basic idea

We first introduce some basic ideas of our TRLAC scheme. We construct the TRLAC scheme by using the fingerprint codes [24] and following the Boneh and Naor's paradigm. This paradigm requires generating a pair of public and secret keys for each bit position of the fingerprint codes when the system is set up, which causes the system public keys to grow linearly with the length of codes. We overcome this problem by embedding the information about fingerprint codes not in the system setup phase but only in the key generation procedure. More specifically, in generating a private key for a user, we concatenate the bit value of each position of the user's codeword onto the end of the identity of the group which this user belongs to. Then we independently hash the concatenation of each bit of the codeword and the identity and finally we produce a private key on the hash results. The user is unable to modify the fingerprint code information involved in his private key since he cannot change the concatenation from his key due to the one-way property of the hash function. We exploit the IBBE scheme in [11] to ensure the security of outsourced data. In this way, data owners can specify sets of group identities that they want to share data with and then encrypt their data with the specified sets before outsourcing them to clouds. If data consumers are in a group with identity included in the specified set, they can access the outsourced data using their access credentials, serving as decryption keys, issued by the TTP.

In case that access credentials are leaked and used to create an illegal access credential for unauthorized access, the tracing procedure of TRLAC finds out the users compromising their access credentials in two steps. First, due to the adopted Boneh and Naor's paradigm, the tracing procedure finds the feasible codeword associated with the illegal access credential used by $\mathcal{PD}$. Second, given this feasible codeword, the tracing procedure calls the tracing algorithm of the underlying fingerprint codes to output a set of codewords associated with the users who leaked their access credentials. To fulfill revocation, we make the group identities to be blinded as exponents in the ciphertexts by the TTP. Once the dishonest users are traced, the TTP, which has the blinding factor, can be called by data owners to revoke access rights from the groups involving the dishonest users.

4.2.2 Construction

We are now ready to describe our TRLAC scheme as consisting of the following procedures.

System Setup: The TTP calls the Setup algorithm to create master public key parameters MPK, a master secret key MSK and a set of codewords for each group. MPK is public to other parties and MSK must be kept secret.

$(MPK, MSK) \leftarrow$ Setup: For each group, this algorithm calls the code generation algorithm $\mathbf{Gen}_{FC}$ to generate a set of codewords

$$\Gamma = \{\omega^{(1)}, \cdots, \omega^{(n)}\}$$

where n is the maximum number of cloud users in each group. This algorithm selects a bilinear group $\mathbb{G}$ of prime order p with generator g. Next, it randomly chooses $h \in \mathbb{G}, \gamma \in \mathbb{Z}_p^*$ and computes

$$u = g^{\gamma}, h^{\gamma}, h^{\gamma^2}, \cdots, h^{\gamma^m},$$

where m is the maximal size of a set of groups with each member as decryptor. The algorithm also chooses a hash function $H : \{0,1\}^* \to \mathbb{Z}_p^*$ which will be modeled as a random oracle in the security proof. It sets the system master public key and secret key, respectively, as

$$MPK = \left(u, h, h^{\gamma}, \cdots, h^{\gamma^m}, e(g,h), H\right), \quad MSK = (g, \gamma).$$

User Admission: When a new user, denoted by (ID, i), $1 \le i \le n$, in group ID wants to join the system, TTP will first check whether this user is valid or not. If yes, TTP first picks a codeword, denoted by $\omega^{(i)} \in \Gamma$, that has not been used before and then calls the KeyGen algorithm to generate a private key for this user.

$SK_{ID,i} \leftarrow \text{KeyGen}(MSK, (ID, i))$: This algorithm uses MSK, the group identity ID and the index i to produce a private key for user (ID, i). For each j from 1 to l, it computes

$$ID_j = ID|j|\omega_j^{(i)}$$

and outputs the private key as

$$SK_{ID,i} = \{K_j\}_{j=1}^{l} = \{g^{\frac{1}{\gamma + H(ID_j)}}\}_{j=1}^{l},$$

We note that the unique codeword $\omega^{(i)}$ is embedded in the private key; thus, this key is solely associated with the user. Hence, to trace a leaked key, we just need to trace the associated fingerprint code $\omega^{(i)}$.

File Creation: A data owner creates a file to be stored in CSS by the following steps. First, the owner encrypts the file with a random symmetric session key $DEK \xleftarrow{R} \mathbb{G}_T$ of some symmetric cryptosystem such as AES. Second, the owner encrypts the symmetric session key DEK with the identities of the groups the owner wants to share his file with. The procedure runs as follows.

$CT \leftarrow \text{Encrypt}(MPK, S, M)$: This algorithm takes as inputs MPK, a set of group identities $S = \{ID_z\}_{z=1}^{r}$ and message $M \in \mathbb{G}_T$, where $r \le m$ and M is the symmetric key DEK to be encrypted. To equip the system with digital forensics, it randomly picks $j \in \{1, 2, \cdots, l\}$ and calls the following algorithm twice by respectively taking in as inputs

$$\{ID_{z,j} = ID_z|j|0\}_{z=1}^{r} \text{ and } \{ID_{z,j} = ID_z|j|1\}_{z=1}^{r}.$$

$Enc \leftarrow \text{Encrypt}'(MPK, \{ID_{z,j}\}_{z=1}^{r}, M)$: This algorithm runs as follows.

- Choose a random $s \in \mathbb{Z}_p^*$ and compute:

$$C_0 = Me(g,h)^s, C_1 = u^{-s}, C_2 = h^{s\prod_{z=1}^{r}(\gamma + H(ID_{z,j}))}.$$

- Output $Enc = (C_0, C_1, C_2)$.

The algorithm sets CT_0 and CT_1 as the output of the above algorithm on input $ID_z|j|0$ and $ID_z|j|1$, respectively. It finally outputs

$$CT = (j, CT_0, CT_1).$$

This CT, together with the ciphertext of data under the symmetric key, forms the encrypted file to be stored in the cloud.

File Access: When a user requests a file from CSS, the server returns the corresponding encrypted file. The user first recovers the symmetric key by decrypting CT and then uses this symmetric key to decrypt the body of the file. The Decrypt algorithm outputs the symmetric key as follows.

$M/\perp \leftarrow \text{Decrypt}(MPK, CT, S, SK_{ID_k,i})$: It takes as inputs MPK, ciphertext

$$CT = (j, CT_0, CT_1)$$

of an encrypted file in the cloud, private key $SK_{ID_k,i}$ of user (ID_k, i) and the set

$$S = \{ID_z\}_{z=1}^{r}$$

of group identities. If S does not include ID_k, this algorithm returns a false symbol $\perp$. Otherwise, it decrypts the ciphertext as follows.

The algorithm chooses K_j from $SK_{ID_k,i}$. If $\omega_j^{(i)} = 0$, which means $ID_{k,j} = ID_k|j|0$, it also chooses CT_0 (otherwise, chooses CT_1) and computes:

$$M' = \left(e\left(C_1, h^{\Delta_{k,j}}\right) \cdot e\left(K_j, C_2\right)\right)^{\frac{1}{\prod_{z=1, z\neq k}^{r} H(ID_{z,j})}}$$

with

$$\Delta_{k,j} = \frac{1}{\gamma}\left(\prod_{z=1, z\neq k}^{r}(\gamma + H(ID_{z,j})) - \prod_{z=1, z\neq k}^{r} H(ID_{z,j})\right).$$

Then it recovers M as $M = C_0/M'$.

Correctness. Suppose CT is well formed for S, then

$$\begin{aligned} M' &= \left(e\left(g^{-\gamma s}, h^{\Delta_{k,j}}\right) \cdot \right. \\ &\quad \left. e\left(g^{\frac{1}{\gamma + H(ID_{k,j})}}, h^{s\prod_{z=1}^{r}(\gamma + H(ID_{z,j}))}\right)\right)^{\frac{1}{\prod_{z=1, z\neq k}^{r} H(ID_{z,j})}} \\ &= \left(e\left(g,h\right)^{-s\prod_{z=1, z\neq k}^{r}(\gamma + H(ID_{z,j})) + s\prod_{z=1, z\neq k}^{r} H(ID_{z,j})} \cdot \right. \\ &\quad \left. e\left(g,h\right)^{s\prod_{z=1, z\neq k}^{r}(\gamma + H(ID_{z,j}))}\right)^{\frac{1}{\prod_{z=1, z\neq k}^{r} H(ID_{z,j})}} \\ &= e(g,h)^s. \end{aligned}$$

Thus, $M = C_0/M'$.

For a user (ID_k, i), if his codeword has a 0 in the j-th position, then he only has the private key for $ID_k|j|0$. Thus, he can only decrypt the ciphertext component CT_0. Otherwise, he can only decrypt CT_1. The point is that a user can have a key for either $ID_k|j|0$ or $ID_k|j|1$ in the position j.

Digital Forensics: When the owner suspects a $\mathcal{PD}$ from having access to his files stored in the cloud without his authorization, he can call the TTP to run this procedure and find the users who have taken part in creating this $\mathcal{PD}$. Usually, the $\mathcal{PD}$ is *imperfect* in the sense that it decrypts the ciphertexts only with a probability less than one. This issue has been extensively discussed and a robust fingerprint code can be employed in this case [4]. Hence, in the rest of this paper, we assume that the $\mathcal{PD}$ is perfect, i.e.,

$$\Pr[\mathcal{PD}(\text{Encrypt}(MPK, S, M)) = M] = 1.$$

When TTP receives a request to enforce digital forensics on a $\mathcal{PD}$ and a set S of group identities, it first calls the Trace algorithm.

$\mathbb{T} \leftarrow \text{Trace}(MPK, \mathcal{PD}, S)$: This algorithm takes as inputs MPK and a $\mathcal{PD}$ able to decrypt the ciphertexts generated with $S = \{ID_z\}_{z=1}^{r}$. It works in two steps. First, this algorithm conducts the following experiments to find the feasible

codeword associated with the illegal access credential used by $\mathcal{PD}$ to access the data owner's file.

This algorithm chooses each j from 1 to l and does the following.

1. Randomly choose two distinct symmetric keys M_j and M'_j.
2. Compute the ciphertexts

$$CT_0 \leftarrow \text{Encrypt}'(MPK, \{ID_z|j|0\}_{z=1}^{r}, M_j),$$

$$CT'_1 \leftarrow \text{Encrypt}'(MPK, \{ID_z|j|1\}_{z=1}^{r}, M'_j).$$

3. Take ciphertext $CT^* = (j, CT_0, CT'_1)$ as the input of $\mathcal{PD}$. Define the output of $\mathcal{PD}$ as M_j^* and set

$$\omega_{\text{j}}^* = \begin{cases} 0 & \text{if } M_j^* = M_j, \\ 1 & \text{otherwise.} \end{cases}$$

Finally, the algorithm uses all ω_j^* for $j = 1, 2, \cdots, l$ to form

$$\omega^* = \omega_1^* \omega_2^* \cdots \omega_l^*,$$

which is set as the feasible codeword associated with the illegal credential.

Second, the algorithm takes this codeword ω^* as input of the tracing algorithm $\mathbf{Tra}_{FC}$ of the underlying fingerprint code scheme. For a suspected group, this tracing algorithm then outputs a set

$$\mathbb{T} \subseteq \{1, \cdots, n\}.$$

Since our system is traceable against t-collusion attack (we give the proof in next section), the set $\mathbb{T}$ is a subset of the users whose access credentials or private keys were leaked. TTP returns this set to the data owner as the response for the digital forensics request so that the owner can take further countermeasures (e.g., lawsuit).

User Revocation: When the digital forensics procedure traces a set of users in group ID_k who have leaked their access credentials, the data owner can revoke the access rights of this group to access his files. To revoke group ID_k from set S, the owner asks the TTP to compute twice

$$C_2 \leftarrow (C_2)^{\frac{1}{\gamma + H(ID_{k,j})}}$$

with

$$ID_{k,j} = ID_k|j|0 \quad \text{for} \quad CT_0$$

and

$$ID_{k,j} = ID_k|j|1 \quad \text{for} \quad CT_1,$$

respectively. Note that although this procedure is invoked by the data owners, it does not require them perform the revocation since all the tasks are done by the TTP.

5. FORMAL SECURITY ANALYSIS

In this section, we evaluate the security of our TRLAC scheme. Intuitively, we require TRLAC to have semantical security and traceability, where the former states that someone having no access credential cannot get any useful information about the file created by the data owner, and the latter states that, if someone accesses the uploaded file with an illegal access credential, then the TTP can find at least one original access credential used to produce the illegal one. The two security properties are formally defined by two games: one is the semantic security game (*Game 1*) and the other is the traceability game (*Game 2*).

Game 1: In this game, to capture the access attempts from unauthorized users colluding with CSP, we define an attacker which is able to query the users' private keys. To illustrate the security against this attack, we challenge the attacker with a ciphertext that cannot be decrypted by any of the queried private keys. The attacker has to output its guess on the challenge ciphertext. More formally, the attacker is able to choose to be challenged on an encryption to a set of group identities, i.e., $S^* = \{ID_z^*\}_{z=1}^{r}$ and ask for any private key of $(ID_k, i \in [1, n])$ on the condition that $ID_k \notin S^*$. This game is defined as follows between an adversary $\mathcal{A}$ and a challenger.

Init: The adversary $\mathcal{A}$ outputs a set $S^* = \{ID_z^*\}_{z=1}^{r}$ of group identities that it wants to attack.
Setup: The challenger runs the setup algorithm to obtain a master public key MPK and give it to adversary $\mathcal{A}$.
Phase 1: The adversary $\mathcal{A}$ specifies a user's identity (ID_k, i). In response, the challenger creates a key for this identity by calling the KeyGen algorithm, and sends this key to $\mathcal{A}$.
Challenge: The adversary $\mathcal{A}$ outputs two equal-length messages M_0 and M_1 with an extra restriction that any queried group identity is not involved in S^*. The challenger flips a coin $\beta \in \{0, 1\}$, encrypts M_β under identity S^* and returns the ciphertext CT^* to $\mathcal{A}$.
Phase 2: This is the same as Phase 1 with the constraint that any group identity queried in this phase must not be included in S^*.
Guess: The attacker outputs a guess $\beta' \in \{0, 1\}$.

The advantage of an attacker $\mathcal{A}$ in this game is defined as

$$Adv_{\mathcal{A}}^{S} = |P[\beta = \beta'] - 1/2|.$$

DEFINITION 1. *A TRLAC system is semantically secure against chosen-plaintext attacks if any polynomial-time adversary $\mathcal{A}$ has only a negligible advantage in the above game.*

The semantic security states that any polynomial-time attacker cannot distinguish the ciphertexts of two equal-length messages encrypted to the challenge set, provided that the attacker cannot query for private keys which can be used to decrypt the challenge ciphertext.

Game 2: In this game, we define an adversary which can collude with authorized users by querying their private keys (these users are then regarded as traitors who leaked their keys). Then the adversary can use these keys to create a $\mathcal{PD}$ through which one can illegally access data without directly identifying the traitors. The adversary terminates the key queries by outputting the pirate decryptor as a challenge. Formally, this game is defined as follows.

Setup: The challenger runs the setup algorithm to obtain the master public key MPK and give it to adversary $\mathcal{A}$.
Query: For a query on (ID, i) from $\mathcal{A}$, the challenger responds by calling the KeyGen algorithm and returning the key $SK_{ID,i}$ to $\mathcal{A}$.
Challenge: The adversary $\mathcal{A}$ outputs a pirate decryptor $\mathcal{PD}$ for a set S^* of group identities.
Trace: The challenger runs the Trace$(MPK, \mathcal{PD}, S^*)$ algorithm and outputs a set $\mathbb{T} \subseteq \{1, \cdots, n\}$ for the suspected

group. Let $\mathbb{S}$ denote the set of users with private keys queried by $\mathcal{A}$. The adversary wins this game if the following conditions hold:

(1) The set $\mathbb{T}$ is empty or is not a subset of $\mathbb{S}$.
(2) There are at most t private key queries with group identities included in S^*.
(3) The pirate decryptor $\mathcal{PD}$ is perfect, i.e.,

$$\Pr[\mathcal{PD}(\text{Encrypt}(MPK, S^*, M)) = M] = 1$$

The first condition is straightforward and the third one is required by the assumption about $\mathcal{PD}$ as described in the digital forensics procedure. The second condition is required by the underlying fingerprint codes [24]. Since this scheme is secure against at most t colluders, then it is also required that there are at most t queried private keys that can be used to directly decrypt the encryption with S^*.

The advantage of an attacker $\mathcal{A}$ in this game is defined as

$$Adv_{\mathcal{A}}^{T} = \Pr[\mathcal{A} \text{ wins}].$$

DEFINITION 2. *A TRLAC system is t-collusion resistant if for all polynomial-time adversary $\mathcal{A}$ we have that $Adv_{\mathcal{A}}^{S}$ and $Adv_{\mathcal{A}}^{T}$ are negligible in* Game 1 *and* Game 2*, respectively.*

The t-collusion resistant security implies that: 1) there is semantic security and 2) at least one member of any collusion of at most t members can be found out.

The following formal claim guarantees that our TRLAC is semantically secure and traceable.

THEOREM 1. *Our TRLAC scheme is t-collusion resistant if the fingerprint codes scheme [24] is t-collusion resistant and the underlying IBBE scheme [11] is semantically secure against chosen-plaintext attacks. Formally, let l denote the length of the fingerprint code and $|\mathcal{M}|$ the size of the message space. Then, for all $t > 0, n \geq t$, any polynomial-time attacker breaks our TRLAC system with advantage at most*

$$Adv_{\mathcal{A}}^{T} \leq l \cdot Adv_{\mathcal{A}}^{S} + \epsilon + \frac{l}{|\mathcal{M}|}.$$

If the TRLAC scheme is semantically secure, then the advantage $Adv_{\mathcal{A}}^{S}$ of any adversary breaking its semantic security is negligible. The code length l is much less than the message space size $|\mathcal{M}|$, so $l/|\mathcal{M}|$ is a negligible quotient. Since the underlying fingerprint codes scheme has been shown ϵ-secure, the advantage $Adv_{\mathcal{A}}^{T}$ of any adversary breaking the scheme's traceability is negligible, which means that the TRLAC scheme is t-collusion resistant. The proof of this theorem is given in the Appendix.

6. PERFORMANCE EVALUATION

In this section, we evaluate the TRLAC scheme both theoretically and experimentally. The analyses show that the introduction of traceability hardly affects the most frequent processes of file outsourcing and file access. The TRLAC scheme is efficient to be employed in cloud storage systems to provide traceability of leaked access credentials, as well as protection of the outsourced data.

6.1 Theoretical Analysis

We first analyze the computational complexity of the proposed scheme in each of its procedures. In this analysis, we treat the underlying fingerprint scheme as a black-box and let $O(\mathbf{Gen}_{FC})$ and $O(\mathbf{Tra}_{FC})$ denote the computation complexity of $\mathbf{Gen}_{FC}$ and $\mathbf{Tra}_{FC}$, respectively. We focus on the most time-consuming operations, i.e., exponentiation and bilinear pairing map, conducted in groups $\mathbb{G}$ and $\mathbb{G}_T$. Their time is respectively denoted by τ_{exp} and τ_{pair}, without discriminating exponentiations in $\mathbb{G}$ and $\mathbb{G}_T$. Also, we do not discriminate a multi-base exponentiation from a single-base one, as the latter is only slightly more efficient. The time to sample a random element in $\mathbb{G}$ is approximated to (indeed, slightly less than) the time to compute an exponentiation. We do not take into account the time costs of the symmetric encryption and decryption in our evaluation, since both are relatively negligible compared to the asymmetric encryption and decryption.

Table 1: Computation

Operation	Computational Complexity
System Setup	$1\tau_{pair} + (m+2)\tau_{exp} + O(\mathbf{Gen}_{FC})$
User Admission	$l \cdot \tau_{exp}$
File Creation	$2 \cdot (\|S\| + 4)\tau_{exp}$
File Access	$2 \cdot \tau_{pair} + \|S\| \cdot \tau_{exp}$
Digital Forensics	$2l \cdot (\|S\| + 4)\tau_{exp} + O(\mathbf{Tra}_{FC})$
User Revocation	$2 \cdot \tau_{exp}$

Table 1 summarizes the computation cost of each procedure in our TRLAC scheme. In this table, m denotes the maximal size of the set of groups in which each member can access a stored file and $|S| \leq m$ denotes the size of set S specified in encryption. Table 1 clearly reveals the performance cost incurred by the *a posteriori* approach of digital forensics in outsourcing data to clouds. In addition to the unavoidable process of digital forensics, only the system setup and user admission procedures are affected. We stress that both procedures are only executed once in an offline stage with respect to outsourcing files. Surprisingly, adding credential leakage traceability affects neither file creation nor access: the users can create files and access them normally without any signiticant overhead. These features render our TRLAC scheme an outstanding solution to *a posteriori* protection of data outsourced to an untrusted third party.

Table 2 compares our TRLAC scheme with other schemes achieving traceability of leaked access credentials in terms of performance, tracing manner (black-box or white-box), bilinear group type (prime-order or composite-order) and revocation support. In this table, the schemes in [5], [6] and [12] are devised for the broadcast encryption and N denotes the number of total users in the systems. The works from [16] to [18] following [6] aim to provide traceability for attribute-based encryption, thus their performance depends on the number of attributes, in which $|\mathcal{U}|$ denotes the size of the attribute universe, $|\gamma|$ the size of the set γ of attributes associated with a user, $|\mathbb{A}|$ the size of an access structure associated with a ciphertext and $|\gamma^*|$ denotes the size of a set γ^* satisfying the access structure. The work [14] supports identity-based traitor tracing system, in which a data owner can choose only one group to access his data at a time, in contrast to a set of groups in our TRLAC scheme. It can be seen that all the schemes except ours, [12] and [14] are constructed in composite-order bilinear groups.

Note that the performance of composite-order bilinear groups is much (about one order) lower than in prime-order

bilinear groups for the same security level. For instance, the scheme [12] is shown to encrypt 6 times faster and decrypt 10 times faster than [5] for the same level of security. Compared to our TRLAC scheme, however, the decryption time of [12] is linear with the size of the vector space which is used to simulate the orthogonality of composite-order groups. Moreover, this scheme [12], as well as [5, 6, 17, 18], requires the ciphertext to be linear with the quadratic root of the number N of total users, which means that the encrypted files to be stored in the cloud will be very large.

Table 2 reveals the superior practicality of our scheme. Let m denote the maximal size of the set of groups. It can be seen that the ciphertext size and pairing computations in decryption are both constant, which provides a cost-effective way for users to securely outsource and access their files. Although the scheme of [14] achieves the same encryption and decryption efficiency, it does not support revocation, which is seen as an important mechanism to protect data security after the tracing results come out. Compared to other schemes, the public key size of our TRLAC is also short, since the parameter m is usually less than the size of total attributes in [16, 17, 18], the quadratic root of the number of total users in [5, 6, 12] and the code length in [14]. Although the private key size is linear with the code length l (which can be relatively short by setting reasonably big ϵ and small t in a practical scenario), the key generation algorithm is only executed once by the TTP in an offline stage with respect to outsourcing files. This confirms the practical attractiveness of our solution.

6.2 Experimental Analysis

We conducted a series of experiments on a PC with 4-core Intel Core i3-2130 CPU running at 3.4GHz, 2.0G RAM, using the C programming language in Windows 7. The involved cryptographic operations (e.g., pairing, exponentiation) were implemented by using the Pairing-Based Cryptography library version 0.5.12 (available at `http://crypto.stanford.edu/pbc`). In the experiments, we evaluated the performance of each procedure of the TRLAC scheme.

We employed Nuida *et al.*'s fingerprint codes which have shorter length than Tardos codes. Recall that to achieve t-collusion resistant traceability with a total number n of users with error probability ϵ, the length of their fingerprint codes is given by Expression (1). Since the traceability is uniquely determined by the fingerprint code length l, we conduct experiments with different code length. Consider a practical scenario where the created file is allowed to be accessed by different departments of different companies and each department has usually a staff size n at most 100. It seems reasonable to assume that at most $t \leq 5$ staff member's access credentials are compromised. Suppose that tracing to an incorrect staff member has probability $\epsilon \approx 10^{-5}$. For such a practical scenario, according to Expression (1), we have that the fingerprint code has length $l \leq 50$. Thus, in the experiments, we set $l = 50, 100, \cdots, 500$. Besides, the system performance is also related to the maximal size m of set, which is determined at the setup phase, and the size $|S|$ of set S decided in encryption. For the above scenario, it seems reasonable to set the number of different companies to 100, i.e., $m = 100$. We thus conduct our experiments with $|S|$ equal to 20, 40, 60 and 80 as well.

The system setup procedure run by TTP outputs a system master public key of size linear in m. For $m = 100$, the time consumed by this procedure turns out to be about 440.5ms, which is negligible for a TTP which will usually have more powerful equipment than our PC.

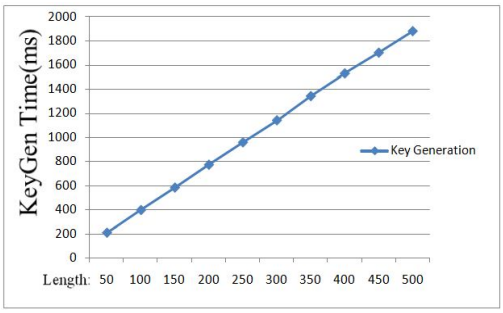

Figure 2: Time consumed in user admission

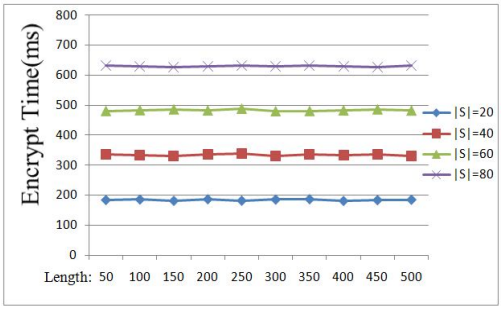

Figure 3: Time consumed in file creation

Fig. 2 shows the time to register a user to the system. It can be seen that the time consumed by the user admission procedure grows linearly with the length of fingerprint codes. The lowest point means that to trace $t \leq 5$ employees who leaked their credentials in a company's department of 100 staff, therequired code lengt is about 50 bits, which yields a time about 200ms to produce a private key. If the number of dishonest employees leaking credentials increases to $t \leq 30$, the code length needs to grow to 500 bits and the time to produce a private key is near 1.9s (represented by the highest point). These results show that our system can tolerate a large number of traitors who leaked their credentials while keeping a manageable cost for TTP.

Fig. 3 shows the time to create a file to be stored in the cloud. The four lines represent four different sizes of the set of encryption groups, respectively: 20, 40, 60 and 80. It can be seen that none of these lines varies as the code length grows. The time consumed by file creation is only linear in

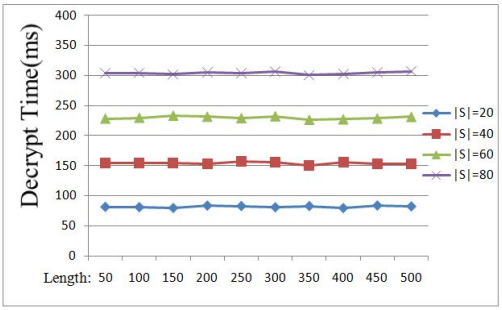

Figure 4: Time consumed in file access

Table 2: Comparison with related work

	Public key size	Private key size	Ciphertext size	Pairings in decryption	Black-box	Prime-order groups	Revocation
[5]	$3+4\sqrt{N}$	1	$6\sqrt{N}$	3	$\surd$	$\times$	$\times$
[6]	$5+9\sqrt{N}$	$1+\sqrt{N}$	$7\sqrt{N}$	4	$\surd$	$\times$	$\surd$
[12]	$2+3\sqrt{N}$	$1+\sqrt{N}$	$7\sqrt{N}$	$2+2\|\vec{v}\|$	$\surd$	$\surd$	$\surd$
[16]	$\|\mathcal{U}\|+4$	$\|\gamma\|+4$	$2\|\mathbb{A}\|+3$	$2\|\gamma^*\|+1$	$\times$	$\times$	$\times$
[17]	$\|\mathcal{U}\|+7+8\sqrt{N}$	$\|\gamma\|+3$	$2\|\mathbb{A}\|+8\sqrt{N}$	$2\|\gamma^*\|+5$	$\surd$	$\times$	$\times$
[18]	$\|\mathcal{U}\|+3+4\sqrt{N}$	$\|\gamma\|+4$	$2\|\mathbb{A}\|+9\sqrt{N}$	$2\|\gamma^*\|+6$	$\surd$	$\times$	$\times$
[14]	$6+2l$	2	6	2	$\surd$	$\surd$	$\times$
Ours	$3+m$	l	6	2	$\surd$	$\surd$	$\surd$

the size of encryption groups. Even for the largest value 80, which means the file is intended to be shared with 80 groups, the time consumed by this procedure is about 630ms, which is a low cost for users to encrypt their outsourced data.

Fig. 4 shows the time to access a file. It can be seen that the time consumed by the file access procedure does not depend on the code length either. It only grows linearly in the size of encryption groups. The four lines represent the time costs associated with different sizes of the set of encryption groups, respectively. For the largest size of 80 groups, the time to access a file is only about 300ms. Therefore, with millisecond-level file creation and file access times, our TRLAC scheme provides an efficient protection for users to secure their outsourced data.

The group revocation procedure in our scheme is also very efficient in the sense that it only requires two exponentiations for a file. We tested that in our PC the time taken by the TTP to revoke access rights of a group is near 16.4ms. We note that this procedure is only executed by the TTP and does not take the user any time.

From the experimental analyses above, our TRLAC scheme provides an efficient way for users to protect the security of their outsourced data. Users can encrypt their data at a low cost before outsourcing them to clouds and the time for any authorized user to access the stored files is even less. The introduction of traceability incurs almost no extra cost for the file creation and access. Even for user admission, the procedure that is most affected by the introduction of traceability, the cost is acceptable. A user can also revoke access privileges of a group to access his files without having to do himself any computation.

7. CONCLUSION

This paper has presented countermeasures to fight illegal leakage of access credentials. We presented TRLAC, a system for tracing and revoking leaked access credentials. Formal proofs show that the proposed scheme is t-collusion resistant. Finally, we conducted a comprehensive performance evaluation. Our analyses shows that the proposed scheme has reasonable performance after being equipped with a mechanism to trace and revoke compromised access credentials.

8. ACKNOWLEDGEMENTS

This paper was supported by the National Key Basic Research Program (973 program) through project 2012CB315905, the Natural Science Foundation of China through projects 61370190, 61173154, 61003214, 61070192, 61021004, 61272501 and 61202465, the Beijing Natural Science Foundation through project 4132056, the Fundamental Research Funds for the Central Universities through project 2012211020212 of Wuhan University and the Research Funds of Renmin University of China through project 14XNLF02, the Open Research Fund of The Academy of Satellite Application and the Open Research Fund of Beijing Key Laboratory of Trusted Computing, and by the European Commission under FP7 projects DwB(INFRA-2010-262608) and Inter-Trust(FP7-ICT-317731), the Spanish Government through projects IPT-2012-0603-430000, TIN2011-27076-C03-01, TIN2012-32757, and TIN2011-27076-C03-01. Sherman S. M. Chow is supported by the Early Career Scheme and the Early Career Award of the Research Grants Council, Hong Kong SAR (CUHK 439713), and Direct Grant (4055018) of the Chinese University of Hong Kong. Josep Domingo-Ferrer is partly supported as an ICREA-Acadàmia researcher by the Government of Catalonia. He holds the UNESCO Chair in Data Privacy, but the views expressed in this paper are his own and do not commit UNESCO.

9. REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson and D. Song, "Provable data possession at untrusted stores," in *ACM CCS'07*, pp. 598-609, 2007.

[2] M. Blaze, G. Bleumer and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *EUROCRYPT'98*, Springer, pp. 127-144, 1998.

[3] D. Boneh and M. Franklin, "Identity-based encryption from the Weil pairing," in *CRYPTO'01*, Springer, pp. 213-229, 2001.

[4] D. Boneh and M. Naor, "Traitor tracing with constant size ciphertext," in *ACM CCS'06*, pp. 501-510, 2008.

[5] D. Boneh, A. Sahai and B. Waters, "Fully collusion resistant traitor tracing with short ciphertexts and private keys," in *EUROCRYPT'06*, Springer, pp. 573-592, 2006.

[6] D. Boneh and B. Waters, "A fully collusion resistant broadcast, trace, and revoke system," in *ACM CCS'06*, pp. 211-220, 2006.

[7] N. Cao, Z. Yang, C. Wang, K. Ren and W. Lou, "Privacy-preserving query over encrypted graph-structured data in cloud computing," in *IEEE ICDCS'11*, pp. 393-402, 2011.

[8] S. Chow, C. Chu, X. Huang, J. Zhou and R. Deng, "Dynamic secure cloud storage with provenance," in *Cryptography and Security, Festschrift Jean-Jacques Quisquater*, Springer, pp. 442-464, 2012.

[9] S. Chow, Y. He, L. Hui and S. Yiu, "SPICE-Simple Privacy-preserving Identity-management for Cloud Environment," in *ACNS'12*, pp. 526-543, 2012.

[10] C. K. Chu and W. G. Tzeng, "Identity-based proxy re-encryption without random oracles," in *ISC'07*, Springer, pp. 189-202, 2007.

[11] C. Delerablée, "Identity-based broadcast encryption with constant size ciphertexts and private keys," in *ASIACRYPT'07*, Springer, pp. 200-215, 2007.

[12] S. Garg, A. Kumarasubramanian, A. Sahai and B. Waters, "Building efficient fully collusion-resilient traitor tracing and revocation schemes," in *ACM CCS'10*, pp. 121-130, 2010.

[13] M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *ACNS'07*, Springer, pp. 288-306, 2007.

[14] F. Guo, Y. Mu and W. Susilo, "Identity-based traitor tracing with short private key and short ciphertext," in *ESORICS'12*, Springer, pp. 609-626, 2012.

[15] K. Kent, S. Chevalier, T. Grance and H. Dang, "Integrating forensic techniques into incident response," *NIST Special Publication 800-86 Notes*, [online]: `http://cybersd.com/sec2/800-86Summary.pdf`

[16] Z. Liu, Z. F. Cao and D. S. Wong, "White-box traceable ciphertext-policy attribute-based encryption supporting any monotone access structures," in *IEEE Trans. Information Forensics and Security*, vol. 8, no. 1, pp. 76-88, 2013.

[17] Z. Liu, Z. F. Cao and D. S. Wong, "Expressive black-box traceable ciphertext-policy attribute-based encryption," IACR ePrint Archive 669, 2012.

[18] Z. Liu, Z. F. Cao and D. S. Wong, "Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay," in *ACM CCS'13*, pp. 475-486, 2013.

[19] H. Lin, J. Shao, C. Zhang and Y. Fang, "CAM: Cloud-assisted privacy preserving mobile health monitoring," in *IEEE Trans. Information Forensics and Security*, vol. 8, no. 6, pp. 985-997, 2013.

[20] M. Li, S. Yu, W. Lou and Y. T. Hou, "Toward privacy-assured cloud data services with flexible search functionalities," in *IEEE ICDCS Workshops'12*, pp. 466-470, 2012.

[21] M. Li, S. Yu, K. Ren and W. Lou, "Securing personal health records in cloud computing: patient-centric and fine-grained data access control in multi-owner settings," in *SecureCom'10*, Springer, pp. 89-106, 2010.

[22] 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," in *IEEE Trans. Parallel and Distributed Systems*, vol. 24, no. 1, pp. 131-143, 2013.

[23] L. Nguyen and R. Safavi-Naini, "Efficient and provably secure trapdoor-free group signature schemes from bilinear pairings," in *ASIACRYPT'04*, Springer, pp. 372-386, 2004.

[24] K. Nuida, S. Fujitsu, M. Hagiwara, T. Kitagawa, H. Watanabe, K. Ogawa and H. Imai, "An improvement of discrete Tardos fingerprinting codes," in *Designs, Codes and Cryptography*, vol. 52, no. 3, pp. 339-362, 2009.

[25] V. Pappas, V. P. Kemerlis, A. Zavou, M. Polychronakis and A. D. Keromytis, "CloudFence: data flow tracking as a cloud service," [online]: `http:www.cs.columbia.edu/~angelos/Papers/\\ 2013/cloudfence.pdf`, 2013.

[26] B. Qin, H. Wang, Q. Wu, J. Liu and J. Domingo-Ferrer, "Simultaneous authentication and secrecy in identity-based data upload to cloud," *Cluster Computing*, vol. 16, no. 4, pp. 845-859, 2013.

[27] F. Sebé, J. Domingo-Ferrer, A. Martínez-Ballesté, Y. Deswarte and J.-J. Quisquater, "Efficient remote data possession checking in critical information infrastructures," in *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 8, pp. 1034-1038, 2008.

[28] U. Somani, K. Lakhani and M. Mundra, "Implementing digital signature with RSA encryption algorithm to enhance the data security of cloud in cloud computing," in *IEEE PDGC'10*, pp.211-216. 2010.

[29] G. Tardos, "Optimal probabilistic fingerprint codes," in *STOC'03*, ACM, pp.116-125, 2003.

[30] Y. S. Tan, R. K. Ko, P. Jagadpramana, C. H. Suen, M. Kirchberg, T. H. Lim, B. S. Lee, A. Singla, K. Mermoud, D. Keller and H. Duc, "Tracking of data leaving the cloud," in *TrustCom'12*, IEEE, pp. 137-144, 2012.

[31] B. Wang, S. Chow, M. LI and H. Li. "Storing shared data on the cloud via security-mediator," in *IEEE ICDCS'13*. To appear, 2013.

[32] C. Wang, S. Chow, Q. Wang, K. Ren and W. Lou, "Privacy-preserving public auditing for secure cloud storage," in *IEEE Trans. Computers*, vol. 62, no. 2, pp. 362-375, 2013.

[33] F. Wang, J. Liu and M. Chen, "CALMS: Cloud-Assisted Live Media Streaming for globalized demands with time/region diversities," in *IEEE INFOCOM'12*, pp. 199-207, 2012.

[34] H. Wang, R. Shea, F. Wang and J. Liu, "On the impact of virtualization on dropbox-like cloud file storage/synchronization services," in *IEEE Conf. International Workshop on Quality of Service*, pp. 1-9, 2012.

[35] B. Waters, "Dual system encryption: realizing fully secure IBE and HIBE under simple assumptions," in *CRYPTO'09*, Springer, pp.619-636, 2009.

[36] K. Yang, X. Jia and K. Ren, "Attribute-based fine-grained access control with efficient revocation in cloud storage systems," in *ASIACCS'13*, pp. 523-528, 2013.

[37] J. Yu, P. Lu, Y. Zhu, G. Xue and M. Li, "Toward secure multikeyword top$-k$ retrieval over encrypted cloud data," in *IEEE Trans. Dependable and Secure Computing*, vol. 10, no. 4, pp. 239-250, 2013.

[38] J. Yuan and S. Yu, "Privacy preserving back-propagation neural network learning made practical with cloud computing," in *IEEE Trans. Parallel and Distributed Systems*. To appear, 2013.

[39] O. Q. Zhang, R. K. Ko, M. Kirchberg, C. H. Suen, P. Jagadpramana and B. S. Lee, "How to track your data: rule-based data provenance tracing algorithms," in *TrustCom'12*, IEEE, pp. 1429-1437, 2012.

APPENDIX

A. PROOF OF THEOREM 1

To prove Theorem 1, we first observe that our TRLAC system is an encryption scheme derived from the IBBE scheme (without tracing) [11], and the latter is semantically secure against chosen-plaintext attacks. Compared to this scheme, in both the key generation and the encryption algorithms of our TRLAC, the input of the hash function is not a single identity but a concatenation of identity, position, and bit of codeword. In particular, producing a private key of the TRLAC scheme is identical to separately calling the key generation of the IBBE scheme l times, and the encryption of TRLAC runs the encryption of the IBBE scheme twice for different inputs. Hence, the semantic security proofs of the IBBE scheme [11] can also be applied to our scheme.

We now prove that our TRLAC system satisfies the t-collusion resistance defined by *Game 2*. The following lemma shows that our system has this resistance and its proof follows Theorem 1 in [4].

LEMMA 1. *Let l denote the length of the fingerprint codes and $|\mathcal{M}|$ the size of the message space. Then, for all $t > 0, n \geq t$, any polynomial-time attacker breaks the t-collusion resistance of our TRLAC system with advantage at most*

$$Adv_{\mathcal{A}}^{T} \leq l \cdot Adv_{\mathcal{A}}^{S} + \epsilon + \frac{l}{|\mathcal{M}|}.$$

PROOF. In the definition of t-collusion resistance, the challenger first runs the system setup algorithm and gives the system master public key to the adversary $\mathcal{A}$. The adversary $\mathcal{A}$ is able to query the private key for any group identity. In response, the challenger runs the key generation algorithm and gives the queried private keys to $\mathcal{A}$. At some point, the adversary $\mathcal{A}$ outputs a pirate decryptor $\mathcal{PD}$ which is created by some or all the queried private keys and can decrypt ciphertexts with set S^* of group identities.

We let $\mathbb{W}$ denote the set of codewords associated with the queried private keys. Then $F(\mathbb{W})$ is the feasible set of $\mathbb{W}$. If the tracing procedure finds a codeword $\omega^* \in F(\mathbb{W})$, then we immediately have $Adv_{\mathcal{A}}^{T} \leq \epsilon$.

In case of $\omega^* \notin F(\mathbb{W})$, which means the tracing procedure cannot trace a codeword feasible for the set of codewords associated with the private keys used in creating $\mathcal{PD}$, we have to bound this probability $\Pr[\omega^* \notin F(\mathbb{W})]$. To do so, we slightly modify the algorithm Trace as follows.

Given the $\mathcal{PD}$ able to decrypt ciphertexts generated with set $S^* = \{ID_z^*\}_{z=1}^{r}$, the modified tracing procedure picks each j from 1 to l and does the following experiments.

1. Randomly choose two messages $M_j \neq M_j'$.
2. If all codewords in $\mathbb{W}$ have a 1 or 0 in position j, compute

   $$CT_0^* \leftarrow \text{Encrypt}'(MPK, \{ID_z^*|j|0\}_{z=1}^{r}, M_j')$$

   and

   $$CT_1^* \leftarrow \text{Encrypt}'(MPK, \{ID_z^*|j|1\}_{z=1}^{r}, M_j');$$

   else, compute

   $$CT_0^* \leftarrow \text{Encrypt}'(MPK, \{ID_z^*|j|0\}_{z=1}^{r}, M_j)$$

   and

   $$CT_1^* \leftarrow \text{Encrypt}'(MPK, \{ID_z^*|j|1\}_{z=1}^{r}, M_j).$$

3. Query the pirate decryptor $\mathcal{PD}$ on the ciphertext

   $$CT^* = (j, CT_0^*, CT_1^*).$$

   Define the output of $\mathcal{PD}$ as M_j^*. If all codewords in $\mathbb{W}$ have a 1 in position j, set the bit $\theta_j^* = 0$ if $M_j^* = M_j$; otherwise, set it to $\theta_j^* = 1$.

   If all codewords in $\mathbb{W}$ have a 0 in position j, set the bit $\theta_j^* = 1$ if $M_j^* = M_j$; otherwise, set it to $\theta_j^* = 0$.

This modified tracing procedure then obtains the traced feasible codeword $\theta^* = \theta_1^*\theta_2^* \cdots \theta_l^*$. In this procedure, if all the codewords in $\mathbb{W}$ have a 1 in position j, then the pirate decryptor outputs a random message M_j^*. With probability $1/|\mathcal{M}|$ at most, $M_j^* = M_j$ holds. Thus, the modified procedure outputs $\theta_j^* = 0$ holds only with the negligible probability $1/|\mathcal{M}|$. Similarly, if all the codewords in $\mathbb{W}$ have a 0 in position j, $\theta_j^* = 1$ holds with the probability $1/|\mathcal{M}|$. We ignore the case that the j-th bit of some codewords in $\mathbb{W}$ is equal to either 1 or 0, since this will not essentially affect the tracing procedure. The experiment above is run for l times, thus we upper-bound the probability by $l/|\mathcal{M}|$, i.e., $\Pr[\omega^* \notin F(\mathbb{W})] \leq l/|\mathcal{M}|$.

We show that the probability of $\mathcal{A}$ being able to distinguish the original tracing procedure from the modified tracing procedure is upper bounded by $l \cdot Adv_{\mathcal{A}}^{S}$. The only difference between these two procedures is the generation of ciphertexts, i.e., CT_0^* or CT_1^*. Then, to distinguish the original tracing procedure from the modified one, the adversary $\mathcal{A}$ needs to distinguish the ciphertexts. Suppose that all the codewords in $\mathbb{W}$ have a 1's at the j-th position. Then the ciphertexts of the original tracing procedure and the modified one are respectively $CT_0 \leftarrow \text{Encrypt}'(MPK, ID^*|j|0, M_j)$ and $CT_0^* \leftarrow \text{Encrypt}'(MPK, ID^*|j|0, M_j')$. Distinguishing CT_0 from CT_0^* is identical to breaking the semantic security. Given the advantage $Adv_{\mathcal{A}}^{S}$ of $\mathcal{A}$ in breaking the semantic security, the probability of $\mathcal{A}$ begin able to distinguish the two tracing procedures is upper-bounded by $l \cdot Adv_{\mathcal{A}}^{S}$.

In summary, the advantage of any polynomial-time adversary in breaking the t-collusion resistance of our TRLAC scheme is at most $Adv_{\mathcal{A}}^{T} \leq l \cdot Adv_{\mathcal{A}}^{S} + \epsilon + l/|\mathcal{M}|$. This completes the proof of Theorem 1. □