1-2018

Attribute-based cloud storage with secure provenance over encrypted data

Hui CUI
Royal Melbourne Institute of Technology

Robert H. DENG
Singapore Management University, robertdeng@smu.edu.sg

Yingjiu LI
Singapore Management University, yjli@smu.edu.sg

DOI: https://doi.org/10.1016/j.future.2017.10.010

Published in Future Generation Computer Systems, 2018 February, Volume 26, Issue 4, Pages 461-472
https://doi.org/10.1016/j.future.2017.10.010

Attribute-based cloud storage with secure provenance over encrypted data

Hui Cui [a,b,*], Robert H. Deng [a], Yingjiu Li [a]

[a] School of Information Systems, Singapore Management University, Singapore
[b] School of Science, RMIT University, Melbourne, Australia

HIGHLIGHTS

- This paper focused on solving the problems in existing cloud storage systems with secure provenance.
- The proposed storage system with secure provenance in this paper protects data privacy, allows fine-grained access control, enables dynamic user management, supports data provider anonymity and traceability, and provides secure data provenance.
- This paper defined the formal security model and analysed the security for the proposed storage system.
- This paper conducted experiments on the proposed storage system to evaluate its performance.

ARTICLE INFO

Article history:
Received 15 August 2016
Received in revised form 30 August 2017
Accepted 5 October 2017
Available online 23 October 2017

Keywords:
Cloud storage
Secure provenance
Access control
Scalability
Confidentiality
Anonymity
Traceability
Revocation

ABSTRACT

To securely and conveniently enjoy the benefits of cloud storage, it is desirable to design a cloud data storage system which protects data privacy from storage servers through encryption, allows fine-grained access control such that data providers can expressively specify who are eligible to access the encrypted data, enables dynamic user management such that the total number of data users is unbounded and user revocation can be carried out conveniently, supports data provider anonymity and traceability such that a data provider's identity is not disclosed to data users in normal circumstances but can be traced by a trusted authority if necessary, and equally important, provides secure data provenance by presenting irrefutable evidence on who has created and modified the data in the cloud. However, most of the existing cloud storage systems with secure provenance either lack the expressiveness in access control or incur too much performance overhead or do not support dynamic user management. In this paper, we solve these problems by presenting an attribute-based cloud storage system with secure provenance. We first give a simple construction without achieving user revocation, and then extend it with an efficient revocation mechanism to prevent revoked data users from accessing the newly encrypted data. Thereafter, we implement the algorithms in the proposed two constructions to evaluate their performance. Our experimental results show that the proposed systems are acceptable to be applied in practice.

1. Introduction

Due to the exponential growth of data, outsourcing data to the cloud is becoming increasingly attractive to both individuals and enterprises. However, the public cloud infrastructure introduces significant security and privacy risks since the cloud is outside the trust domain of data providers. A promising solution for the protection of data security and privacy in cloud storage systems is to ask data providers to encrypt their data and upload the resulting ciphertexts to the cloud. One challenge in the design of such a cloud storage system is how to enable fine-grained access control of encrypted data over data users. Consider a scenario where a company stores its encrypted data to the cloud and entitles its employees different attributes (or credentials) to decrypt data from the cloud or upload encrypted data to the cloud. Since there may be a group of employees in the company that are authorized to access the same data, in addition to fine-grained access control, it is also important to track who executes which type of operations to the encrypted data in the cloud. In general, to securely and conveniently enjoy the benefits of cloud storage, it is desirable to build a cloud storage system which has the following properties.

1. *Data privacy* to protect the confidentiality of the outsourced data such that the cloud cannot learn any information about the real data stored in the cloud;

* Corresponding author at: School of Science, RMIT University, Melbourne, Australia.
E-mail address: hui.cui@rmit.edu.au (H. Cui).

2. *Fine-grained access control* such that data providers can specify who are able to access the encrypted data in an expressive manner;
3. *Scalability* to make the storage space independent of the number of data users, i.e., the size of each ciphertext does not grow linearly to the number of its privileged data users;
4. *Dynamic user management* to enable unbounded number of data users in the system and revocation of data users;
5. *Data provider anonymity and traceability* such that it preserves data provider anonymity in normal circumstances while keeping his/her identity traceable by a trusted authority in case that the data provider misbehaves;
6. *Secure data provenance* to provide irrefutable evidence on who creates and modifies data in the cloud storage system.

To our knowledge, several cloud storage systems with secure provenance (e.g., [1–3]) have been proposed, but they all have certain drawbacks. The system in [1] is built from the composite-order groups, which is too inefficient [4] to be applied in practice. The systems in [1,2] can only support a simple access policy with one attribute, and the maximum number of data users allowed in the instantiation of [2] must be predefined, which is not favorable for the scalable environment of cloud computing. The system in [3] assumes that the cloud is honest such that it does not collude with data users and there is a secure channel between the cloud and data providers/data users, which is a too strong assumption to be practical. In this paper, we overcome the disadvantages of the existing solutions by presenting an attribute-based cloud storage system with secure provenance.

1.1. Challenges and our contributions

In a traditional public-key encryption (PKE) scheme, each ciphertext is targeted for decryption by a single data user, and thus it lacks the expressiveness and scalability needed for the data sharing. To address this problem, Sahai and Waters [5] introduced attribute-based encryption (ABE), where each data user is issued a private attribute key associated with a set of attributes belonging to him/her by a trusted attribute authority (AA), and a data provider can specify an access policy (or access structure) over a set of attributes when encrypting data. Any data user whose set of attributes satisfies the access policy associated with a ciphertext is able to decrypt this ciphertext using his/her private attribute key. Therefore, ABE provides a one-to-many encryption scheme with ability to perform data encryption and decryption defined over certain descriptive attributes, and the required space for storing ciphertexts is independent of the number of data users.

In the proposed attribute-based cloud storage system with secure provenance, we will use the large universe ciphertext-policy ABE (CP-ABE) scheme [6] which puts no limit on the number of attributes as well as the number of data users and is built in the prime-order groups. Due to the one-to-many encryption property of the underlying CP-ABE scheme, the size of each ciphertext in the proposed storage system depends on the access policy rather than the number of data users. We then equip this large universe CP-ABE scheme in [6] with an efficient revocation mechanism following the revocation technique proposed in [7] such that the revoked data users are not able to access the newly encrypted data stored in the cloud storage system.

It remains to overcome the challenge of secure data provenance, which requires to provide irrefutable and unforgeable evidence on who creates and modifies the data in the cloud storage system. With the goal of equipping the proposed storage system with such a capability, each data provider/user who creates, modifies or writes to the data, in addition to encrypting the data, also generates a "tricky" signature on the ciphertext which does not reveal his/her identity. The cloud accepts an uploaded ciphertext only if the signature on the ciphertext is a valid one. Also, the system administrator (SA) in the proposed system who de facto plays the role of the AA in a CP-ABE scheme keeps a "trapdoor" on each data provider/user when issuing the private attribute key. The trapdoor in essence links the identity of a data provider/user to his/her private attribute key and plays the rule that one stone kills two birds. First, the signature on a ciphertext is generated by a data provider using his/her private attribute key, and the trapdoor can be used by the SA to trace who have uploaded a ciphertext by tracking the identity of the signer from the appended signature to the ciphertext. Second, private attribute keys in a standard CP-ABE scheme are anonymous since they are defined over attributes shared by multiple data users, so the keys might be abused by malicious data users who can share their private attribute keys with others without having the risk of being caught, but a data user's trapdoor links his/her identity with the private attribute key, making the data user reluctant to share his/her private key with others.

The main contributions of this paper can be summarized as follows.

- We present a framework of an attributed-based cloud storage system with secure provenance, and formally define its security;
- We propose a concrete construction on attribute-based cloud storage with secure provenance, which is built from bilinear pairings in the prime-order groups, and provides data privacy, fine-grained access control, scalability, dynamic user management, and data provider anonymity and traceability;
- We implement the proposed storage system and conduct experiments to evaluate its performance.

1.2. Related work

Attribute-Based Encryption. Sahai and Waters [5] introduced the notion of ABE, and Goyal et al. [8] formulated key-policy ABE (KP-ABE) and CP-ABE as two complimentary forms of ABE. In CP-ABE, a private attribute key is associated with a set of attributes and a ciphertext is associated with an access policy, while the situation is reversed in KP-ABE. Nevertheless, we believe that KP-ABE is less flexible than CP-ABE because the access policy is determined once a user's private attribute key is issued.[1] Bethencourt, Sahai and Waters [9] proposed the first CP-ABE construction, but it is secure under the generic group model. Cheung and Newport [10] presented a CP-ABE scheme secure under the standard model, but it only supports the AND access structures. A CP-ABE system with expressive access structures was proposed by Goyal et al. [11] based on the number theoretic assumption. Rouselakis and Waters [6] built a large universe CP-ABE system in the prime-order groups to improve the efficiency of ABE in the composite-order groups while overcoming the limitation of bounded attribute space.

Secure Provenance. With the goal of guaranteeing data authenticity, cloud storage systems with secure provenance were proposed in [1–3,12]. Hasan, Sion and Winslett [12] first considered the security and privacy issues of a provenance system, but they did not present any detail on how to build a secure provenance system. Lu et al. [1] proposed a provenance system which efficiently achieves user privacy and data confidentiality using group signature, but it is built in the inefficient composite-order groups, and it does not support expressive access control. On the basis of the work in [1], Chow et al. [2] built an efficient and

[1] In this paper, unless otherwise specified, what we talk about is CP-ABE.

secure cloud storage system which supports dynamic users and data provenance, but it only supports simple one-attribute access policies, and the number of data users allowed for the system is bounded. Li et al. [3] proposed a provenance system based on techniques of group signature, attribute-based signature and broadcast encryption, which supports fine-grained access control policies, but it assumes that the cloud is honest in the sense that it does not collude with data users and it has secure channels with data providers/users.

1.3. Organization

The remainder of this paper is organized as follows. In Section 2, we briefly review the notions and definitions relevant to this paper. In Section 3, after depicting the architecture of an attribute-based cloud storage system with secure provenance, we present its security model. In Section 4, we give a concrete attribute-based storage scheme with secure provenance, and analyze its security, and then compare it with related systems in the literature qualitatively as well as quantitatively through computer simulations. We conclude the paper in Section 5.

2. Preliminaries

In this section, we review some basic cryptographic notions and definitions to be used in this paper.

2.1. Bilinear pairings and complexity assumptions

Let G be a group of a prime order p with a generator g. We define $\hat{e}$: $G \times G \to G_1$ to be a bilinear map if it is bilinear such that for all $g \in G$, and $a, b \in Z_p$, we have $\hat{e}(g^a, g^b) = \hat{e}(g, g)^{ab}$, and non-degenerate such that $\hat{e}(g, g) \neq 1$ [13,14]. We say that G is a bilinear group if the group operation in G is efficiently computable and there exists a group G_1 and an efficiently computable bilinear map $\hat{e}$: $G \times G \to G_1$ as above.

Decisional $(q-1)$ **Assumption** [6]. The decisional $(q-1)$ problem is that for any probabilistic polynomial-time (PPT) algorithm, given $\overrightarrow{y} =$

$$
\begin{array}{ll}
g, g^s, & \\
g^{a^i}, g^{b_j}, g^{s \cdot b_j}, g^{a^i b_j}, g^{a^i/b_j^2} & \forall (i,j) \in [q,q], \\
g^{a^i/b_j} & \forall (i,j) \in [2q,q] \text{ with } i \neq q+1, \\
g^{a^i b_j/b_{j'}^2} & \forall (i,j,j') \in [2q,q,q] \text{ with } j \neq j', \\
g^{sa^i b_j/b_{j'}}, g^{sa^i b_j/b_{j'}^2} & \forall (i,j,j') \in [q,q,q] \text{ with } j \neq j',
\end{array}
$$

it is difficult to distinguish $(\overrightarrow{y}, \hat{e}(g,g)^{a^{q+1}s})$ from $(\overrightarrow{y}, Z)$, where $g \in G, Z \in G_1, a, s, b_1, \ldots, b_q \in Z_p$ are randomly chosen.

2.2. Access structures and linear secret sharing schemes

Definition 1 (*Access Structures [15,16]*)**.** Let $\{P_1, \ldots, P_n\}$ be a set of parties. A collection $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}}$ is monotone if $\forall B, C$: if $B \in \mathbb{A}$ and $B \subseteq C$, then $C \subseteq \mathbb{A}$. An (monotone) access structure is a (monotone) collection $\mathbb{A}$ of non-empty subsets of $\{P_1, \ldots, P_n\}$, i.e., $\mathbb{A} \subseteq 2^{\{P_1,\ldots,P_n\}} \setminus \{\emptyset\}$. The sets in $\mathbb{A}$ are called the authorized sets, and the sets not in $\mathbb{A}$ are called the unauthorized sets.

Definition 2 (*Linear Secret Sharing Schemes (LSSSs) [15,16]*)**.** Let P be a set of parties. Let $\mathbb{M}$ be a matrix of size $l \times n$. Let ρ: $\{1, \ldots, l\} \to P$ be a function that maps a row to a party for labeling. A secret sharing scheme Π over a set of parties P is a linear secret-sharing scheme over Z_p if

1. The shares for each party form a vector over Z_p.
2. There exists a matrix $\mathbb{M}$ which has l rows and n columns called the share-generating matrix for Π. For $i = 1, \ldots, l$, the xth row of matrix $\mathbb{M}$ is labeled by a party $\rho(i)$, where ρ: $\{1, \ldots, l\} \to P$ is a function that maps a row to a party for labeling. Considering that the column vector $v = (\mu, r_2, \ldots, r_n)$, where $\mu \in Z_p$ is the secret to be shared and $r_2, \ldots, r_n \in Z_p$ are randomly chosen, then $\mathbb{M}v$ is the vector of l shares of the secret μ according to Π. The share $(\mathbb{M}v)_i$ belongs to a party $\rho(i)$.

Every LSSS also enjoys the linear reconstruction property [15]. Suppose that Π is an LSSS for access structure $\mathbb{A}$. Let $\mathbf{A}$ be an authorized set, and define $I \subseteq \{1, \ldots, l\}$ as $I = \{i | \rho(i) \in \mathbf{A}\}$. Then the vector $(1, 0, \ldots, 0)$ is in the span of rows of matrix $\mathbb{M}$ indexed by I, and there exist constants $\{w_i \in Z_p\}_{i \in I}$ such that, for any valid shares $\{v_i\}$ of a secret μ according to Π, we have $\sum_{i \in I} w_i v_i = \mu$. These constants $\{w_i\}$ can be found in polynomial time with respect to the size of the share-generating matrix $\mathbb{M}$ [17].

Boolean Formulas [15]. Access policies can also be described in terms of monotonic boolean formulas. LSSS access structures are more general, and can be derived from representations as boolean formulas. There are standard techniques to convert any monotonic boolean formula into a corresponding LSSS matrix. The boolean formula can be represented as an access tree, where the interior nodes are AND and OR gates, and the leaf nodes correspond to attributes. The number of rows in the corresponding LSSS matrix will be the same as the number of leaf nodes in the access tree.

2.3. Signature-of-knowledge

In a zero-knowledge proof protocol [18], the verifier is convinced that the prover knows a certain quantity w satisfying some kinds of relation R with respect to a commonly known string x. That is, the prover convinces the verifier that he knows some w such that $(w, x) \in R$. If a proof-of-knowledge protocol can be done in such a way that the verifier learns nothing other that the validity of the statement, this protocol is called a zero-knowledge Proof of Knowledge (PoK) protocol [18].

A PoK protocol for a binary relation R is a 3-round ZKPoK protocol between two parties, namely, a prover P and a verifier V. For every input $(w, x) \in R$ to P and x to V, the first round of the protocol consists of P sending a commitment t to V. V then replies with a challenge c in the second round and P concludes by sending a response z in the last round. At the end of the protocol, V outputs 1 meaning "accept" or 0 meaning "reject". We say that a protocol transcript (t, c, z) is *valid* if the output of an honest verifier V is accept, which is also know as the completeness property. A PoK protocol has to satisfy the following two properties.

- Soundness. A cheating prover can at most answer one of the many possible challenges. Specifically, there exists an efficient algorithm KE, called knowledge extractor, that on input x, a pair of valid transcripts (t, c, z) and (t, c', z') with $c \neq c'$, outputs w such that $(w, x) \in R$.
- Zero-Knowledge. There exists an efficient algorithm KS, called zero-knowledge simulator, that on input x and a challenge c, outputs a pair (t, z) such that (t, c, z) is a valid transcript having the same distribution as a real protocol transcript resulted from the interaction between a prover P with input $(w, x) \in R$ and an honest verifier V.

Any PoK protocol can be turned into non-interactive form, which is called Signature of Knowledge (SoK) [19], by setting the challenge to the hash value of the commitment together with the message to be signed [20].

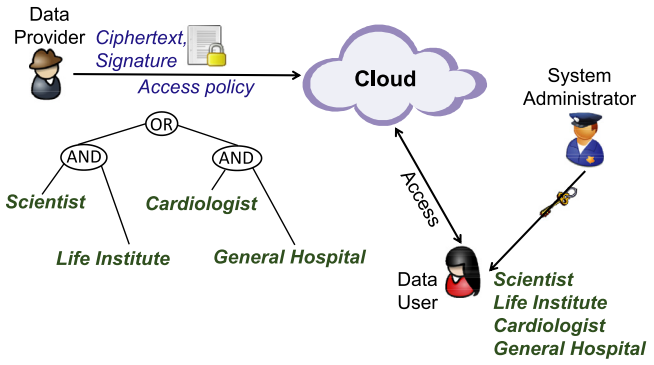


Fig. 1. Architecture of attribute-based cloud storage with secure provenance.

3. System architecture and security model

In this section, we describe the system architecture and formal security definition of an attribute-based cloud storage system with secure provenance.

3.1. System architecture

As shown in Fig. 1, the architecture of the proposed attribute-based cloud storage system with secure provenance involves four types of entities: data providers, system administrator (SA), cloud and data users. The SA, who generates a master private key and publishes the corresponding public parameter, issues every data provider/user a private attribute key which is associated with his/her attributes and identification information, and keeps a list of identities and trapdoors of all registered data providers/users. A data provider/user may be authorized to read, write to or modify the data. When sending a storage request, a data provider/user encrypts the document under an access policy over a set of attributes, and subtly signs the encrypted document using his/her private attribute key without leaking his/her identity information. To read the encrypted data, a data provider/user decrypts the ciphertext using his/her private attribute key if his/her attributes satisfy the access policy associated with the ciphertext. If the data provider/user intends to write to or modify the data, after processing the data, he/she creates a ciphertext and attaches a signature to the ciphertext. The cloud checks the validity of the signature without learning the signer's identity, and accepts the storage request if the signature is a valid one. In the event that a ciphertext stored in the cloud is involved in a dispute, the SA can trace who creates the ciphertext based on the signature associated with the ciphertext and the list of identities and trapdoors the SA keeps.

Concerning the adversarial model, we assume that the cloud is "curious-but-honest" in the sense that it may attempt to obtain the underlying plaintexts of the encrypted data but it honestly follows the specified system operations such as store and check. We assume that data users may try to access data beyond their authorized privileges.

3.2. Framework

An attribute-based cloud storage system with secure provenance consists of the following algorithms.

- Setup(1^λ) $\rightarrow$ (*pars*, *msk*). Taking the security parameter λ as the input, this algorithm, run by the SA, outputs the public parameter *pars* and the master private key *msk* for the system.
- KeyGen(*pars*, *msk*, ID, $\mathbf{A}$) $\rightarrow$ $sk_{\mathbf{A}}^{\mathrm{ID}}$. Taking the public parameter *pars*, the master private key *msk*, an identity ID and an attribute set $\mathbf{A}$ as the input, this algorithm, run by the SA, outputs a private attribute key $sk_{\mathbf{A}}^{\mathrm{ID}}$ over the attribute set $\mathbf{A}$ for user ID, and adds ID and its tracing trapdoor[2] to an initially empty user list *ul*.
- Encrypt(*pars*, M, $(\mathbb{M}, \rho)$) $\rightarrow$ CT. Taking the public parameter *pars*, a message M and an access structure $(\mathbb{M}, \rho)$ (here ρ is a function that associates the rows of $\mathbb{M}$) over the universe of attributes as the input, this algorithm, run by the data provider or the user who has access to the encrypted data, outputs a ciphertext CT.
- Sign(*pars*, ID, $sk_{\mathbf{A}}^{\mathrm{ID}}$, CT) $\rightarrow$ σ. Taking the public parameter *pars*, an identity ID, a private key $sk_{\mathbf{A}}^{\mathrm{ID}}$ and a ciphertext CT as the input, this algorithm, run by the data provider/user who has access to the encrypted data, outputs a signature σ.
- Decrypt(*pars*, CT, $sk_{\mathbf{A}}^{\mathrm{ID}}$) $\rightarrow$ $M/\perp$. Taking the public parameter *pars*, a ciphertext CT and an private attribute key $sk_{\mathbf{A}}^{\mathrm{ID}}$ associated to an attribute set $\mathbf{A}$ and an identity ID as the input, this algorithm, run by the user, outputs either the message M when the private key $sk_{\mathbf{A}^{\mathrm{ID}}}$ satisfies the access structure, or a symbol $\perp$ indicating the failure of the decryption.
- Verify(*pars*, CT, σ) $\rightarrow$ 1/0. Taking the public parameter *pars*, a ciphertext CT and a signature σ as the input, this algorithm, run by the cloud, outputs 1 if σ is a valid signature for CT or 0 otherwise.
- SignTrace(*pars*, σ, *ul*) $\rightarrow$ ID/$\perp$. Taking the public parameter *pars*, a signature σ and a user list *ul* as input, this algorithm, run by the SA, outputs an identity ID of the signer or $\perp$ otherwise.

We require that an attribute-based cloud storage system with secure provenance is correct, meaning that for all messages M and all access structures $(\mathbb{M}, \rho)$ with authorized attribute sets $\mathbf{A}$, if (*pars*, *msk*) $\leftarrow$ Setup(1^λ), $sk_{\mathbf{A}}^{\mathrm{ID}} \leftarrow$ KeyGen(*pars*, *msk*, ID, $\mathbf{A}$), CT $\leftarrow$ Encrypt(*pars*, M, $(\mathbb{M}, \rho)$), $\sigma \leftarrow$ Sign(*pars*, ID, $sk_{\mathbf{A}}^{\mathrm{ID}}$, CT), then Decrypt(*pars*, CT, $\mathbf{A}$, $sk_{\mathbf{A}}^{\mathrm{ID}}$) $= M$, and Verify(*pars*, CT, σ) $\rightarrow$ 1, SignTrace(*pars*, σ, *ul*) $\rightarrow$ ID.

3.3. Security definitions

An attribute-based cloud storage system with secure provenance should ensure confidentiality, anonymity, unforgeability and traceability, of which the latter two are combined into one game.

Confidentiality. Assuming that the adversary makes the key generation queries adaptively, we define the confidentiality for attribute-based cloud storage with secure provenance by the following game between a challenger algorithm $\mathcal{C}$ and an adversary algorithm $\mathcal{A}$, based on the security model of indistinguishability under chosen-plaintext attacks (IND-CPA) for ciphertext-policy attribute-based encryption (CP-ABE) [16].

- Setup. Algorithm $\mathcal{C}$ runs the setup algorithm. Algorithm $\mathcal{C}$ gives the public parameter *pars* to algorithm $\mathcal{A}$, and keeps the master private key *msk*.
- Phase 1. Algorithm $\mathcal{A}$ issues key generation queries to algorithm $\mathcal{C}$. Algorithm $\mathcal{A}$ sends pairs ($\mathbf{A}_i$, ID_i) to algorithm $\mathcal{C}$, where $\mathbf{A}_i$ is an attribute set of user ID_i. Algorithm $\mathcal{C}$ responds by returning the corresponding private attribute key $sk_{\mathbf{A}_i}^{\mathrm{ID}_i}$ to algorithm $\mathcal{A}$.

[2] Note that the generation of $sk_{\mathbf{A}}^{\mathrm{ID}}$ involves a tracing trapdoor for user ID.

- Challenge. Algorithm $\mathcal{A}$ chooses two messages M_0^* and M_1^* and an access structure $(\mathbb{M}^*, \rho^*)$ with the constraint that the key generation queries $\{sk_{\mathbf{A}_i}^{\mathrm{ID}_i}\}$ in Phase 1 do not satisfy the access structure $(\mathbb{M}^*, \rho^*)$. Algorithm $\mathcal{C}$ chooses a random bit $\beta \in \{0, 1\}$, and sends algorithm $\mathcal{A}$ a challenge ciphertext CT^* which is an encryption of M_β^* under the access matrix $(\mathbb{M}^*, \rho^*)$.
- Phase 2. Algorithm $\mathcal{A}$ continues issuing the key generation queries with the constraint that the key generation queries $\{sk_{\mathbf{A}_i}^{\mathrm{ID}_i}\}$ do not satisfy the access structure $(\mathbb{M}^*, \rho^*)$ in the challenge phase.
- Guess. Algorithm $\mathcal{A}$ makes a guess β' for β, and it wins the game if $\beta' = \beta$.

Algorithm $\mathcal{A}$'s advantage in this game is defined as $\Pr[\beta = \beta'] - 1/2$. We say that an attribute-based cloud storage system with secure provenance is secure under IND-CPA model if all PPT adversaries have at most a negligible advantage in the security parameter λ. In addition, an attribute-based cloud storage system with secure provenance is said to be selectively secure under the IND-CPA model if an Init stage is added before the Setup phase where algorithm $\mathcal{A}$ commits to the challenge access structure $(\mathbb{M}^*, \rho^*)$.

Anonymity. Anonymity prevents an adversary from distinguishing the signer's identity of a signature. In the anonymity game, the adversary is given the public parameters, as well as the access to the key generation and signing oracles, and its goal is to guess which of two identities generates the signature in the challenge phase, without being given either of the private keys. We define the anonymity game between a challenger algorithm $\mathcal{C}$ and an adversary algorithm $\mathcal{A}$ as follows.

- Setup. Algorithm $\mathcal{C}$ runs the setup algorithm, gives the public parameter *pars* to algorithm $\mathcal{A}$ and keeps the master private key *msk*.
- Phase 1. Algorithm $\mathcal{A}$ makes the following queries to algorithm $\mathcal{C}$.
  - The key generation query. Algorithm $\mathcal{A}$ sends pairs $(\mathbf{A}_i, \mathrm{ID}_i)$ to algorithm $\mathcal{C}$, where $\mathbf{A}_i$ is an attribute set of user ID_i. Algorithm $\mathcal{C}$ responds by returning the corresponding private key $sk_{\mathbf{A}_i}^{\mathrm{ID}_i}$ to algorithm $\mathcal{A}$.
  - The signing query. Algorithm $\mathcal{A}$ sends $(\mathrm{ID}_i, \mathrm{CT}_i)$ to algorithm $\mathcal{C}$. Algorithm $\mathcal{C}$ responds by returning the corresponding signature to algorithm $\mathcal{A}$.
- Challenge. Algorithm $\mathcal{A}$ chooses a ciphertext CT^* and two identities ID_0^*, ID_1^* with the constraint that there are no key generation queries $\{sk_{\mathbf{A}_i}^{\mathrm{ID}_i}\}$ in Phase 1 on ID_i^* for $i \in \{0, 1\}$. The challenger chooses a random bit $\beta \in \{0, 1\}$, and sends algorithm $\mathcal{A}$ a challenge signature σ^*, a signature on CT^* under ID_β^*.
- Phase 2. Algorithm $\mathcal{A}$ continues issuing the key generation and signing queries as follows.
  1. The key generation query. The same as in Phase 1 except that the key generation queries $\{sk_{\mathbf{A}_i}^{\mathrm{ID}_i}\}$ on ID_0^* and ID_1^* are disallowed.
  2. The signing query. The same as in Phase 1.
- Guess. Algorithm $\mathcal{A}$ makes a guess β' for β, and it wins the game if $\beta' = \beta$.

Algorithm $\mathcal{A}$'s advantage in this game is defined as $\Pr[\beta = \beta'] - 1/2$. We say that an attribute-based cloud storage system with secure provenance is anonymous if all PPT adversaries have at most a negligible advantage in the security parameter λ.

Unforgeability and Traceability. Unforgeability requires that nobody can forge a valid signature on behalf of a legal data user except the data user himself/herself. Traceability means that the identity of a data user who creates a signature can be traced, which prevents any malicious data user from misbehavior. As the definitions of unforgeability and traceability are similar to each other, we define them into one game. In the unforgeability and traceability game, given access to the key generation and signing queries, the adversary aims to output a signature that cannot trace to any data user on which a key generation query has been issued via the signature tracing algorithm. Below we define the unforgeability and traceability game between a challenger algorithm $\mathcal{C}$ and an adversary algorithm $\mathcal{A}$.

- Setup. Algorithm $\mathcal{C}$ runs the setup algorithm, gives the public parameter *pars* to algorithm $\mathcal{A}$ and keeps the master private key *msk*.
- Phase 1. Algorithm $\mathcal{A}$ makes the following queries to algorithm $\mathcal{C}$.
  - The key generation query. Algorithm $\mathcal{A}$ sends pairs $(\mathbf{A}_i, \mathrm{ID}_i)$ to algorithm $\mathcal{C}$, where $\mathbf{A}_i$ is an attribute set of user ID_i. Algorithm $\mathcal{C}$ responds by returning the corresponding key $sk_{\mathbf{A}_i}^{\mathrm{ID}_i}$ to algorithm $\mathcal{A}$, and stores $(\mathrm{ID}, sk_{\mathbf{A}_i}^{\mathrm{ID}_i})$ to a user list *ul*.
  - The signing query. Algorithm $\mathcal{A}$ sends $(\mathrm{ID}_i, \mathrm{CT}_i)$ to algorithm $\mathcal{C}$. If there is no key generation query on ID_i that has been issued, algorithm $\mathcal{C}$ generates a private key $sk_{\mathbf{A}_i}^{\mathrm{ID}_i}$, and creates a signature using $sk_{\mathbf{A}_i}^{\mathrm{ID}_i}$ for CT_i. Otherwise, algorithm $\mathcal{C}$ responds by returning the corresponding signature σ_i to algorithm $\mathcal{A}$.
- Output. Algorithm $\mathcal{A}$ outputs a ciphertext and signature pair $(\mathrm{CT}^*, \sigma^*)$, and wins the game if $1 \leftarrow \mathrm{Verify}(pars, \mathrm{CT}^*, \sigma^*)$, and σ^* cannot trace to any ID_i on which a key generation query has been issued, CT^* has never been issued to the signing query.

Algorithm $\mathcal{A}$'s advantage in this game is defined as $\Pr[\mathcal{A}\ \text{wins}]$. We say that an attribute-based cloud storage system with secure provenance is unforgeable and traceable if all PPT adversaries have at most a negligible advantage in the security parameter λ.

4. Attribute-based cloud storage supporting secure provenance

In this section, we give two concrete constructions on attribute-based cloud storage supporting secure provenance, one disallows user revocation while the other enables, and analyze their security and performance.

4.1. Construction

On the basis of the large universe CP-ABE scheme proposed in [6], we present a concrete construction on attribute-based cloud storage with secure provenance, using techniques including embedding identification information in the private attribute key, and generating signatures via signature of knowledge to subtly achieve user traceability without losing the advantages of ABE and simultaneously preserving user anonymity and provenance unforgeability. Let G be a bilinear group of a prime order p with a generator g, and $\hat{e}$: $G \times G \rightarrow G_1$ denote the bilinear map.

- Setup. This algorithm takes the security parameter 1^λ as the input. It randomly chooses $u, h, v, w \in G, \alpha \in Z_p$, and two collision resistant hash functions H: $G^m \times {G_1}^2 \rightarrow Z_p$, H_0: $G^m \times G_1 \rightarrow G$ for m being the number of elements from G, and sets an initially empty user list *ul*. The public parameter is $pars = (H, H_0, g, u, h, w, v, z)$ where $Z = \hat{e}(g, g)^\alpha$, and the master private key is $msk = g^\alpha$. This algorithm is similar to that in [6] except with two additional hash functions.

- KeyGen. This algorithm takes the public parameter *pars*, the master private key *msk*, an identity ID and an attribute set **A** as the input. Denote k by the size of **A**, and $A_1, \ldots, A_k \in Z_p$ be the elements of **A**. It randomly chooses $\beta_{id}, r, r_1, \ldots, r_k \in Z_p$, and computes

  $$sk_1 = (g^{\alpha})^{\beta_{id}} w^r, \quad sk_2 = g^r,$$
  $$\forall i \in [1,k] \;\; sk_{i,1} = (u^{A_i} h)^{r_i} v^{-r}, \quad sk_{i,2} = g^{r_i}.$$

  It adds (ID, β_{id}) to the user list *ul*, and outputs the private attribute key $sk_{\mathbf{A}}^{\mathrm{ID}} = (\beta_{id}, sk_1, sk_2, \{sk_{i,1}, sk_{i,2}\}_{i\in[1,k]})$ associated with a set of attributes **A** for user ID. This algorithm mostly follows that in [6] except the computation of sk_1.
- Encrypt. This algorithm takes the public parameter *pars*, a message $M \in G_1$ and an LSSS access structure $(\mathbb{M}, \rho)$ where the function ρ associates the rows of $\mathbb{M}$ to attributes as the input. Let $\mathbb{M}$ be an $l \times n$ matrix. It randomly chooses a vector $\overrightarrow{v} = (\mu, y_2, \ldots, y_n) \in Z_p^n$. These values will be used to share the encryption exponent μ. For $i = 1$ to l, it calculates $v_i = \overrightarrow{v} \cdot \mathbb{M}_i$, where $\mathbb{M}_i$ is the vector corresponding to the ith row of $\mathbb{M}$. Also, it randomly chooses $z_1, \ldots, z_l \in Z_p$, and computes

  $$C = M \cdot Z^{\mu}, \quad D = g^{\mu},$$
  $$\forall i \in [1,l] \;\; C_i = w^{v_i} v^{z_i}, \; D_i = g^{z_i}, \; E_i = (u^{\rho(i)} h)^{-z_i}.$$

  It outputs the ciphertext $\mathrm{CT} = ((\mathbb{M}, \rho), C, D, \{(C_i, D_i, E_i)\}_{i\in[1,l]})$. Note that to meet the encryption requirement for messages of large sizes, the Encrypt algorithm can be changed by using a symmetric key K to replace M, and then running a symmetric encryption algorithm to encrypt M using K. This algorithm is the same as that in [6].
- Sign. This algorithm takes the public parameter *pars*, a private key $sk_{\mathbf{A}}^{\mathrm{ID}}$ and a ciphertext CT as the input. It computes the signature of knowledge (SoK)

  $$\mathrm{SoK}\left\{ \begin{pmatrix} \beta_{id}, sk_1, \\ sk_2 \end{pmatrix} : \begin{matrix} \hat{e}(sk_1, g) = Z^{\beta_{id}} \cdot \hat{e}(w, sk_2) \\ \wedge \; B = H_0(\mathrm{CT})^{\beta_{id}} \end{matrix} \right\} (\mathrm{CT})$$

  as follows, which proves the ownership of a valid private attribute key without leaking any secret key and identity information. Firstly, it randomly chooses $s \in Z_p$, $g_1, g_2, g_3 \in G$, and computes

  $$R_0 = g^s, \quad R_1 = sk_1 \cdot {g_1}^s, \quad R_2 = sk_2 \cdot {g_2}^s.$$

  Then, it randomly chooses $d_0, d_1 \in Z_p$, and computes

  $$K_1 = \hat{e}(w, g_2)^{-d_0} \hat{e}(g_1, g)^{d_0} Z^{d_1}, \; K_2 = H_0(\mathrm{CT})^{d_1},$$
  $$K_0 = g^{d_0}, \quad c = H(B, R_0, R_1, R_2, K_0, K_1, K_2, \mathrm{CT}),$$
  $$\theta_0 = d_0 - c \cdot s, \quad \theta_1 = d_1 - c \cdot \beta_{id}.$$

  Finally, it outputs the signature $\sigma = (g_1, g_2, B, R_0, R_1, R_2, \theta_0, \theta_1, c)$.
- Decrypt. This algorithm takes the public parameter *pars*, a ciphertext $((\mathbb{M}, \rho), C, D, E, \{(C_i, D_i, E_i)\})$ and a private key $sk_{\mathbf{A}}^{\mathrm{ID}}$ for an identity ID and an attribute set **A** as the input. Suppose that **A** satisfies $(\mathbb{M}, \rho)$. Define I as $I = \{i : \rho(i) \in \mathbf{A}\}$. Denote by $\{w_i \in Z_p\}_{i\in I}$ a set of constants such that if $\{v_i\}$ are valid shares of any secret μ according to $\mathbb{M}$, then $\sum_{i\in I} w_i v_i = \mu$. It computes

  $$B = \frac{\hat{e}(D, sk_1)}{\prod_{i\in I} (\hat{e}(C_i, sk_2) \hat{e}(D_i, sk_{i,1}) \hat{e}(E_i, sk_{i,2}))^{w_i}} = \hat{e}(g,g)^{\alpha\mu\beta_{id}},$$

  and cancels out $B^{1/\beta_{id}}$ from C to obtain the plaintext M. This algorithm follows that in [6] but with a different intermediate value **B**.
- Verify. This algorithm takes the public parameter *pars*, a ciphertext CT, a signature $(g_1, g_2, g_3, B, R_0, R_1, R_2, R_3, \theta_0, \theta_1, \theta_2, c)$ as the input. It computes

  $$K_0' = g^{\theta_0} {R_0}^c, \quad R' = \frac{\hat{e}(R_1, g)}{\hat{e}(w, R_2)},$$
  $$K_1' = \hat{e}(w, g_2)^{-\theta_0} \hat{e}(g_1, g)^{\theta_0} Z^{\theta_1} {R'}^c, \quad K_2' = H_0(\mathrm{CT})^{\theta_1} B^c.$$

  If $c = H(B, R_0, R_1, R_2, K_0', K_1', K_2', \mathrm{CT})$, it accepts the signature. Otherwise, it rejects.
- SignTrace. This algorithm takes the public parameter *pars*, a signature σ and a user list *ul* as the input. If there exists a pair (ID, β_{id}) such that $B = H(\mathrm{CT})^{\beta_{id}}$, it outputs ID.

Theorem 1. *Assuming that the* $(q-1)$ *assumption holds in G, and SoK is a secure signature of knowledge, the above scheme is secure.*

Proof. We elaborate the proof in Appendix A. The Rouselakis–Waters scheme [6] is known to be selectively IND-CPA secure assuming that the decisional $(q-1)$ assumption holds in G. Since the encryption part of the proposed construction is similar to that in [6] the proof for indistinguishability mostly follows that in [6].

Due to the property of SoK, the signatures will only tell the fact that the user has a valid private key but nothing else and cannot be forged by anyone without a valid private key. Thus, it remains to prove the security of SoK to show that the proposed scheme is anonymous and unforgeable.

In addition, it is clear that the signature is traceable via the signature tracing algorithm.

4.2. Extension to user revocation

In the previous construction, new data users can decrypt previously generated ciphertexts by requesting the private attribute key associated with their credentials, but it fails to deter the revoked data users from accessing the encrypted data. Since a data user's access to a cloud storage system could be terminated at some point, it is desirable to assure that the encrypted data will immediately become unavailable to those revoked data users. To date, there are several approaches introduced to address this issue (e.g., [7,13,21–23]). An efficient revocation method using the binary tree data structure [24] was presented in [7,21] to reduce the size of key updates and remove secure channels from the key update phase (note that this methodology is then further improved in [23] to reduce the workloads of data users in revocation and decryption), where the SA issues a long term private key to each data user and publicly broadcasts key updates at the beginning of each time period, and only non-revoked data users can generate decryption keys to decrypt newly created ciphertexts for the current time period from their long term private keys and the key updates. Another feasible solution to this crux was introduced in [22], where the cloud updates the ciphertexts stored in the system using only publicly available information such that when the access right of a user is revoked, all stored files immediately become unavailable to this user after the update process. Either the technique in [7] or that in [23] or that in [22] can be applied to the previously proposed attribute-based cloud storage system supporting secure provenance to achieve user revocation. Below we simply show how to combine the previous construction with binary tree data structure [24] to accomplish user revocation (See Fig. 2), where the SA issues a long term private attribute key to each data user and publicly broadcasts key updates at the beginning of each time period, but only non-revoked data users can generate decryption/signing keys from their long term private attribute keys and the key updates to decrypt ciphertexts or create signatures under the current time period.

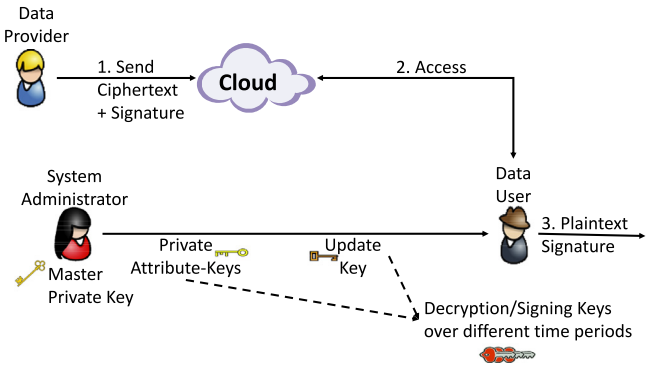


Fig. 2. Revocation in attribute-based cloud storage with secure provenance.

Recall the definition about binary tree described in [7]. Denote BT as a binary tree with N leaves corresponding to N users. Let **root** be the root node of the tree BT. If θ is a leaf node, then Path(θ) denotes the set of nodes on the path from θ to **root**, which includes both θ and **root**. If θ is a non-leaf node, then θ_l, θ_r denote left and right child of θ. Assume that nodes in the tree are uniquely encoded as strings, and the tree is defined by all of its node descriptions. An algorithm called KUNodes is used to compute the minimal set of nodes for which key update needs to be published so that only the non-revoked data users at a time period t are able to decrypt the ciphertexts, which takes a binary tree BT, a revocation list rl and a time period t as the input, and outputs a set of nodes which is the minimal set of nodes in BT such that none of the nodes in rl with corresponding time period before or at t (data users revoked at or before t) have any ancestor (or, themselves) in the set, and all other leaf nodes (corresponding to non-revoked data users) have exactly one ancestor (or, themselves) in the set.

- Setup. The same as that in the above construction except with additional elements $u_0, h_0 \in G$, BT, rl and st, where rl is an empty list storing revoked users, BT is a binary tree with at least N leaf nodes, and st is a state which is set to be BT.
- KeyGen. This algorithm firstly chooses an undefined leaf node θ from the binary tree BT, and stores ID in this node. Then, for each node $x \in$ Path(θ), it runs as follows.
  1. It fetches g_x from the node x. If x has not been defined, it randomly chooses β_{id}, $g_x \in G$, computes $g'_x = (g^{\alpha})^{\beta_{id}}/g_x$, and stores g_x in the node x.
  2. It randomly chooses $r_x, r_{x,1}, \ldots, r_{x,k} \in Z_p$, and computes
  $$sk_{x,1} = g'_x w^{r_x}, \quad sk_{x,2} = g^{r_x},$$
  $$\forall i \in [1,k] \ sk^{(i)}_{x,1} = (u^{A_i} h)^{r_{x,i}} v^{-r_x}, \quad sk^{(i)}_{x,2} = g^{r_{x,i}}.$$
  3. It outputs the private attribute key $sk^{\mathrm{ID}}_{\mathbf{A}} = (\beta_{id}, \{x, sk_{x,1}, sk_{x,2}, \{sk^{(i)}_{x,1}, sk^{(i)}_{x,2}\}_{i\in[1,k]}\}_{x\in\mathrm{Path}(\theta)})$ associated with a set of attributes $\mathbf{A}$ for user ID and an updated state st.
  4. The SA adds (ID, β_{id}) to the user list ul.
- KeyUp. This algorithm takes the public parameter $pars$, the master private key msk, a time period t, a revocation list rl and a state st as the input. For all $x \in$ KUNodes(BT, rl, t), it fetches g_x from the node x. Then, it randomly chooses $s_x \in Z_p$, and computes
  $$ku_{x,1} = g_x \cdot ({u_0}^t h_0)^{s_x}, \quad ku_{x,2} = g^{s_x}.$$
  It outputs $ku_t = \{x, ku_{x,1}, ku_{x,2}\}_{x\in\mathrm{KUNodes(BT,\ rl,\ t)}}$ as the key update information.
- DecKG. This algorithm takes the public parameter $pars$, an identity ID with a private attribute key $sk^{\mathrm{ID}}_{\mathbf{A}}$ and the transformation key update information ku_t as the input. Denote I as Path(θ), J as KUNodes (BT, rl, t). It parses $sk^{\mathrm{ID}}_{\mathbf{A}}$ as $(\beta_{id}, \{x, sk_{x,1}, sk_{x,2}, sk_{x,3}, \{sk^{(i)}_{x,1}, sk^{(i)}_{x,2}\}_{i\in[1,k]}\}_{x\in I})$, ku_t as $\{x, ku_{x,1}, ku_{x,2}, ku_{x,3}\}_{x\in J}$ for some set of nodes I, J. If $I \cap J = \emptyset$, it returns $\perp$. Otherwise, for any node $x \in I \cap J$, it randomly chooses $r'_x, s'_x \in Z_p$, and computes
  $$\begin{aligned} dk_1 &= sk_{x,1} \cdot ku_{x,1} \cdot w^{r'_x} \cdot ({u_0}^t h_0)^{s'_x} \\ &= (g^{\alpha})^{\beta_{id}} \cdot w^{r_x+r'_x} \cdot ({u_0}^t h_0)^{s_x+s'_x}, \\ dk_2 &= sk_{x,2} \cdot g^{r'_x} = g^{r_x+r'_x}, \quad dk_3 = ku_{x,2} \cdot g^{s'_x} = g^{s_x+s'_x}, \\ dk_{i,1} &= sk^{(i)}_{x,i} \cdot v^{-r'_x} = (u^{A_i} h)^{r_{x,i}} v^{-(r_x+r'_x)}, \quad dk_{i,2} = sk^{(i)}_{x,2}. \end{aligned}$$
  It outputs the decryption/signing key $dk^{\mathrm{ID}}_{\mathbf{A},t} = (\beta_{id}, dk_1, dk_2, \{dk_{i,1}, dk_{i,2}\}_{i\in[1,k]}, dk_3)$.
- Encrypt. The same as that in the above construction except that the ciphertext $\mathrm{CT} = ((\mathbb{M}, \rho), t, C, D, E, \{(C_i, D_i, E_i)\}_{i\in[1,l]})$, where t is a time period, $E = ({u_0}^t h_0)^{\mu}$.
- Sign. This algorithm computes the signature of knowledge (SoK)
  $$\mathrm{SoK}\left\{\begin{pmatrix}\beta_{id}, dk_1,\\ dk_2, dk_3\end{pmatrix} : \begin{matrix}\hat{e}(dk_1, g) = Z^{\beta_{id}} \cdot \hat{e}(w, dk_2)\\ \hat{e}({u_0}^t h_0, dk_3)\\ \wedge\ B = H_0(\mathrm{CT})^{\beta_{id}}\end{matrix}\right\}(\mathrm{CT})$$
  as follows. Firstly, it randomly chooses $s \in Z_p, g_1, g_2, g_3 \in G$, and computes
  $$R_0 = g^s, \quad R_1 = dk_1 \cdot {g_1}^s, \quad R_2 = dk_2 \cdot {g_2}^s, \quad R_3 = dk_3 \cdot {g_3}^s.$$
  Then, it randomly chooses $d_0, d_1 \in Z_p$, and computes
  $$\begin{aligned} K_1 &= Z^{d_1} \hat{e}(w, g_2)^{-d_0} \hat{e}(g_1, g)^{d_0} \hat{e}({u_0}^t h_0, g_3)^{-d_0}, \\ K_0 &= g^{d_0}, \quad K_2 = H_0(\mathrm{CT})^{d_1}, \\ c &= H(B, R_0, R_1, R_2, R_3, K_0, K_1, K_2, \mathrm{CT}), \\ \theta_0 &= d_0 - c \cdot s, \quad \theta_1 = d_1 - c \cdot \beta_{id}. \end{aligned}$$
  Finally, it outputs the signature $\sigma = (g_1, g_2, g_3, B, R_0, R_1, R_2, R_3, \theta_0, \theta_1, c)$.
- Decrypt. This algorithm computes the message M as
  $$\begin{aligned} B &= \frac{\hat{e}(D, dk_1)}{\prod_{i\in I}(\hat{e}(C_i, dk_2)\hat{e}(D_i, dk_{i,i})\hat{e}(E_i, dk_{i,2}))^{w_i}\hat{e}(E, dk_3)} \\ &= \hat{e}(g, g)^{\alpha\beta_{id}\mu}, \end{aligned}$$
  and then cancels out $B^{1/\beta_{id}}$ from C to obtain the plaintext M.
- Verify. This algorithm computes
  $$\begin{aligned} K'_0 &= g^{\theta_0} {R_0}^c, \quad R' = \frac{\hat{e}(R_1, g)}{\hat{e}(w, R_2)\hat{e}({u_0}^t h_0, R_3)}, \\ K'_2 &= H_0(\mathrm{CT})^{\theta_1} B^c, \\ K'_1 &= Z^{\theta_1} \hat{e}(w, g_2)^{-\theta_0} \hat{e}(g_1, g)^{\theta_0} \hat{e}({u_0}^t h_0, g_3)^{-\theta_0} {R'}^c. \end{aligned}$$
  If $c = H(B, R_0, R_1, R_2, R_3, K'_0, K'_1, K'_2, \mathrm{CT})$, it accepts the signature. Otherwise, it rejects.
- SignTrace. The same as that in the construction in Section 4.

Note that the security games for this construction are similar to those defined in Section 3 except that the adversary is allowed to issue additional queries on the key updates and decryption/signing keys. The security proof for this scheme is similar to that in Section 4, where the simulation for the key update and decryption/signing key can be done by using the technique in [23], and we omit the details about the proof.

Remarks. Notice that the efficiency of the above construction can be further improved by using the technique introduced in [23],

Table 1
Comparison of properties between previous schemes and our system, where "Exp" denotes expressive, "Unb" denote unbounded, "Sca" denotes scalable, "Con" denotes confidential, "Ano" denotes anonymous, "Tra" denotes traceable, "Unf" denotes unforgeable, "Col" denotes collusion resistant, and "Rev" denotes user revocation.

	Exp	Unb	Sca	Con	Ano	Tra	Unf	Coll	Rev
[1]	×	√	√	√	√	√	√	√	×
[2]	×	×	√	√	√	√	√	√	√
[3]	√	√	√	√	√	√	√	×	√
Construction	√	√	√	√	√	√	√	√	×
Extension	√	√	√	√	√	√	√	√	√

where an untrusted server (the server does not keep any secret, and thus the server's role can be played by the cloud) is introduced to mitigate the workloads to data users involved in revocation and decryption, thereby making each data user only need to perform several exponentiations to accomplish key update and decryption.

4.3. Property analysis

To our knowledge, besides our work, concrete constructions in [1–3] can also be used for data forensics in the setting of cloud storage. The provenance system in [1] efficiently achieves user anonymity and message confidentiality using group signature, but it was built in the inefficiency composite-order group, and it did not support expressive data access control. On the basis of the work in [1], an efficient and secure cloud storage system supporting data provenance was given in [2], but it only worked for a simple one-attribute access policy, and the number of data users allowed for the system was bounded. The provenance system supporting fine-grained access control policies in [3] was built under the assumption that the cloud is honest by using techniques of group signature, attribute-based signature and broadcast encryption, but it required secure channels between the cloud and data providers/users, i.e., the cloud and data providers/users cannot collude.

We compare the features of the proposed storage systems and others in Table 1. It is clear that the proposed system is comparable to existing schemes, since it supports expressiveness such that the data provider can specify who can access the encrypted files by attributes and scalability such that the storage space is independent to the number of privileged data users, allows dynamic data users such that the total number of data users is unbounded and any new data user can decrypt previously encrypted messages (and revoked data user is not able to access the newly encrypted data) and collusion attacks such that the cloud and malicious data users can collude together, and achieves confidentiality such that the content of the message is unaccessible to those non-privileged data users, anonymity such that the cloud could not tell who processes which file, unforgeability and traceability such that no data provider/user is able to forge a valid signature that could not be traced to identity.

4.4. Performance evaluation

Denote l as the number of attributes in an access structure, k as the size of an attribute set associated with the private attribute key. Since the scheme in [1] is built from the composite-order group and thus is incomparable to those built in the prime-order group [6], and the system in [3] is presented without detailing the specific encryption algorithm, we compare the computational costs incurred for encryption, signing, decryption and verification between the encryption system in [2] and our storage system(s) supporting data forensics in Table 2. It is not difficult to see that the proposed schemes are less efficient than the one in [2], but it supports additional properties such as expressiveness and unbounded users. We will show that even though the proposed storage schemes are less efficient, but they are still acceptable for the applications in practice by simulation.

Table 2
Computational costs of the proposed storage systems and the one in [2].

	Encrypt	Sign	Decrypt	Verify
System in [2]	$3 \cdot E$	$8 \cdot E$ $+3 \cdot P$	$2 \cdot E$ $+2 \cdot P$	$8 \cdot E$ $+5 \cdot P$
Construction	$(5l+2) \cdot E$	$9 \cdot E$ $+2 \cdot P$	$\leq k \cdot E$ $+(3k+1) \cdot P$	$8 \cdot E$ $+4 \cdot P$
Extension	$(5l+4) \cdot E$	$18 \cdot E$ $+9 \cdot P$	$\leq k \cdot E$ $+(3k+2) \cdot P$	$10 \cdot E$ $+6 \cdot P$

We implement our proposed storage systems[3] in Charm [25], which is a framework developed to facilitate rapid prototyping of cryptographic schemes and protocols. Since all Charm routines are designed under the asymmetric groups, our constructions are transformed to the asymmetric setting before the implementation. That is, three groups G, $\hat{G}$ and G_1 are used and the pairing $\hat{e}$ is a function from $G \times \hat{G}$ to G_1. Notice that it has been stated in [6] that the assumptions and the security proofs can be converted to the asymmetric setting in a generic way.

The Charm-0.43 and Python 3.4 are used in the implementation. The experiments are run on a laptop with Intel Core i5-4210U CPU @ 1.70 GHz and 8.00 GB RAM running 64-bit Ubuntu 14.04.

All of the experiments are conducted over four elliptic curves: SS512, MNT159, MNT201 and MNT224, of which SS512 is a supersingular elliptic curve with the bilinear pairing on it being symmetric Type 1 pairing, and the pairings on the other three curves are asymmetric Type 3 pairings. These four curves provides security levels of 80-bit, 80-bit, 100-bit and 112-bit, respectively. Table 3 lists the average computation time for the exponentiation and pairing calculations over the four curves.

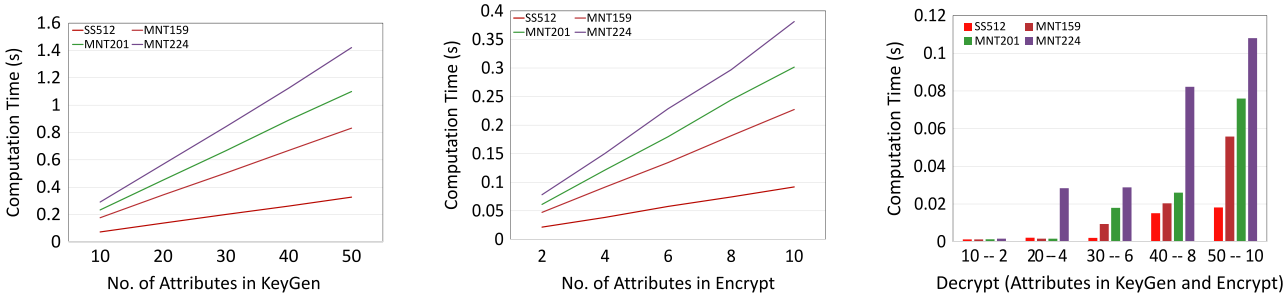
In the proposed two schemes, the computation time of the Setup, Sign or Verify algorithm will not change with the number of attributes (which are summarized in Table 3), and we focus on the computational costs of the KeyGen (or DecKG), Encrypt and Decrypt algorithms. In our experiments, all private attribute keys and ciphertexts are randomly generated over randomly chosen attribute sets and access structures.

To begin with, we implement the scheme presented in Section 4.1 which does not support user revocation. We test the average computation time of generating private attribute keys over 10 to 50 attributes (see Fig. 3(a)), and the average computation time of creating ciphertexts over access structures composed of 2 to 10 attributes (see Fig. 3(b)). Also, we test the average computation time of decrypting ciphertexts over access structures composed of 2 to 10 attributes using private attribute keys over 10 to 50 attributes (see Fig. 3(c)). For the four curves tested in the experiments, the average computation time of generating private attribute keys for 10 to 50 attributes ranges from 0.08 s to 1.5 s, the average computation time of creating ciphertexts for access structures with 2 to 10 attributes ranges from 0.03 s to 0.4 s, and the average computation time of decrypting ciphertexts for access

[3] We will not implement the scheme in [2], as it is easy to see from Table 2 that it is more efficient than ours.

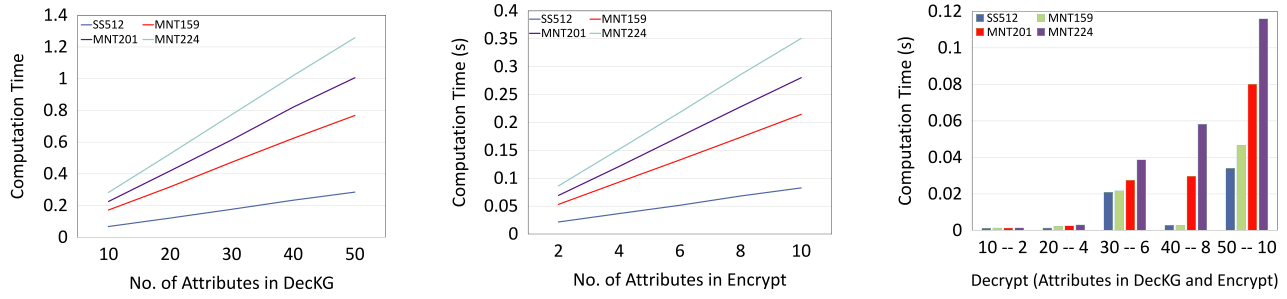
Table 3
The average computation time (in ms) for the operations on group, pairing and Setup, Sign, Verify algorithms in the proposed construction (first line) and extension (second line) over different elliptic curves.

ECs	Exp. G	Exp. $\hat{G}$	Exp. G_1	Pairing	Setup	Sign	Verify
SS512	0.262	0.263	0.185	1.333	26.339 26.686	18.251 31.974	12.895 22.249
MNT159	0.073	0.799	1.219	4.273	42.727 45.972	40.266 72.224	38.373 68.838
MNT201	0.103	0.943	1.569	5.551	56.622 59.479	53.128 94.961	51.794 92.851
MNT224	0.164	1.261	1.981	6.836	69.608 73.522	64.898 117.561	65.506 113.953



(a) KeyGen. (b) Encrypt. (c) Decrypt.

Fig. 3. Computation cost of the KeyGen algorithm (left), Encrypt algorithm (middle) and Decrypt algorithm (right) in the scheme without user revocation.



(a) DecKG. (b) Encrypt. (c) Decrypt.

Fig. 4. Computation cost of the DecKG algorithm (left), Encrypt algorithm (middle) and Decrypt algorithm (right) in the extension to support user revocation.

structures with 2 to 10 attributes using private attribute keys of 10 to 50 attributes ranges from 1.3 ms to 110 ms.

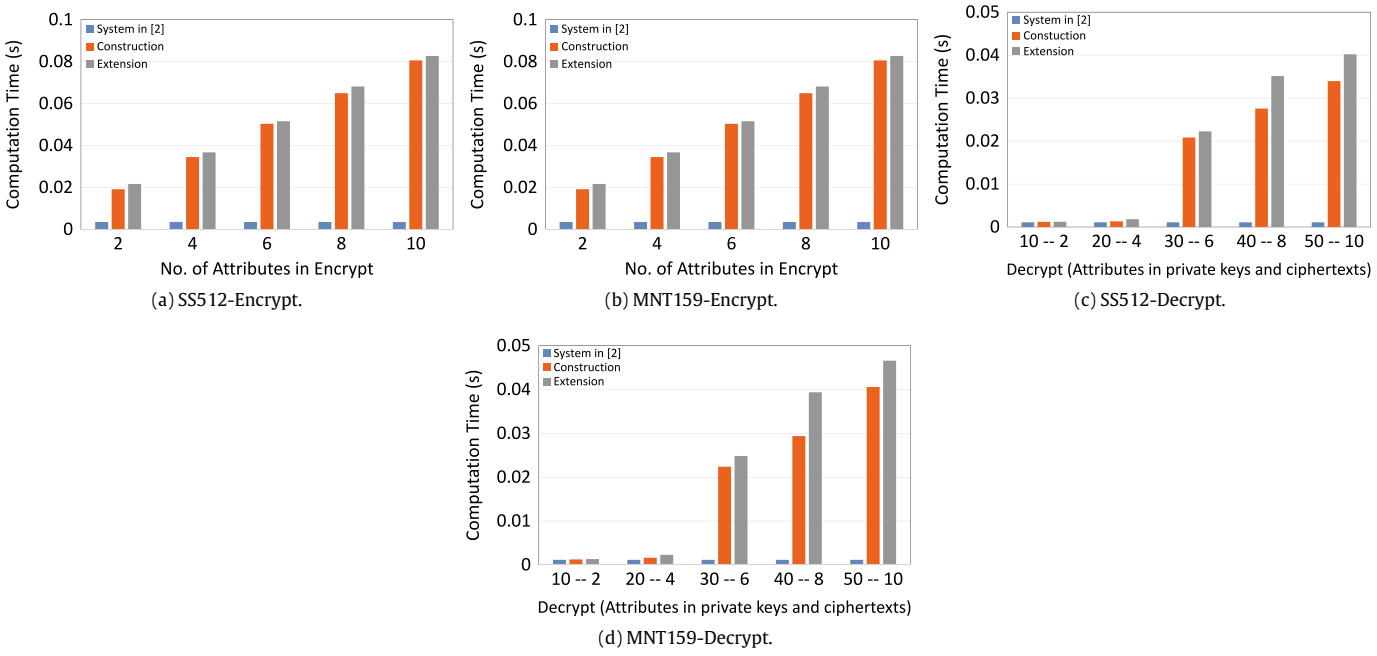
Next, we implement the scheme given in Section 4.2 which enables user revocation. We test the average computation time of generating decryption/signing keys over 10 to 50 attributes (see Fig. 4(a)), and the average computation time of creating ciphertexts over access structures composed of 2 to 10 attributes (see Fig. 4(b)). In addition, we test the average computation time of decrypting ciphertexts over access structures composed of 2 to 10 attributes using private attribute keys over 10 to 50 attributes (see Fig. 4(c)). For the four curves tested in the experiments, the average computation time of generating private attribute keys for 10 to 50 attributes ranges from 0.07 s to 1.3 s, the average computation time of outputting ciphertexts for access structures with 2 to 10 attributes ranges from 0.03 s to 0.4 s, and the average computation time of decrypting ciphertexts for access structures with 2 to 10 attributes using private attribute keys of 10 to 50 attributes ranges from 1.2 ms to 120 ms. The experimental results clearly show that the proposed scheme with user revocation does not add much cost to the original scheme and are acceptable to be applied in practice.

5. Conclusions

Several cloud storage systems with secure provenance have been proposed in the literature (e.g., [1–3,12]) but they either lack the expressiveness in access control or incur too much performance overhead or do not support dynamic user management. In this paper, we attempted to solve these problems by presenting an attribute-based cloud storage system which protects data privacy from storage servers, enables fine-grained access control, allows dynamic user management, supports data provider anonymity and traceability by an authority, and provides secure data provenance. We started with providing a concrete construction that does not support user revocation, and then extended it with the revocation functionality. We also implemented the proposed two systems to study their performance.

Acknowledgments

This work is supported by the Singapore National Research Foundation under the NCR Award Number NRF2014NCR-NCR001-012 and the AXA Research Fund.



(a) SS512-Encrypt. (b) MNT159-Encrypt. (c) SS512-Decrypt.

(d) MNT159-Decrypt.

Fig. 5. Computation cost of the Encrypt and Decrypt algorithms in the scheme [2] and our proposed two schemes.

Appendix A. Security reduction

Lemma 1. *Assuming that the* $(q-1)$ *assumption holds in* G*, then the proposed scheme is selectively indistinguishable.*

Proof. The Rouselakis–Waters scheme [6] is known to be selectively indistinguishable under the decisional $(q-1)$ assumption. Since the encryption part of our construction is similar to that in [6], our proof for Theorem 1 mostly follows that in [6]. Assume that there exists an adversary algorithm $\mathcal{A}$ that can break the selective security of our scheme. Then we can build a challenger algorithm $\mathcal{B}$ that solves the $(q-1)$ assumption.

- Init. Algorithm $\mathcal{A}$ gives algorithm $\mathcal{B}$ a challenge access structure $(\mathbb{M}^*, \rho^*)$, where $\mathbb{M}^*$ is an $l \times n$ matrix.
- Setup. Algorithm $\mathcal{B}$ randomly chooses hash functions H and H_0. For other elements in the public parameter, algorithm $\mathcal{B}$ generates as in [6].
- Phase 1 and Phase 2. In both Phase 1 and Phase 2, algorithm $\mathcal{B}$ has to output the private keys for the pairs of identity ID and set of attributes $\mathbf{A} = \{A_1, \ldots, A_{|\mathbf{A}|}\}$ issued by algorithm $\mathcal{A}$. Since $\mathbf{A}$ does not satisfy $(\mathbb{M}^*, \rho^*)$, there exists a vector $\overrightarrow{w} = (w_1, \ldots, w_n)^{\perp} \in Z_p^n$ such that $w_1 = -\beta_{id}$, $(\mathbb{M}_i^*, \overrightarrow{w}) = 0$ for all $i \in I = \{i | i \in [l] \wedge \rho(i)^* \in \mathbf{A}\}$. Algorithm $\mathcal{B}$ computes $\overrightarrow{w}$ using linear algebra. In addition, it randomly chooses $\tilde{r} \in Z_p$, and implicitly sets

$$r = \tilde{r} + w_1 a^q + w_2 a^{q-1} + \cdots + w_n a^{q+1-n} = \tilde{r} + \sum_{i \in [n]} w_i a^{q+1-i},$$

and then computes

$$sk_1 = g^{\alpha \beta_{id}} w^r = (g^{a^{q+1}} g^{\tilde{\alpha}})^{\beta_{id}} g^{a\tilde{r}} \prod_{i \in [n]} g^{w_i a^{q+2-i}},$$

$$sk_2 = g^r = g^{\tilde{r}} \prod_{i \in [n]} (g^{a^{q+1-i}})^{w_i},$$

$$sk_3 = g^{ar} = (g^{\tilde{r}} \prod_{i \in [n]} (g^{a^{q+1-i}})^{w_i})^a.$$

On the other hand, for all $i \in [|\mathbf{A}|]$, it randomly chooses $\tilde{r}_i \in Z_p$, and implicitly sets

$$r_i = \tilde{r}_i + \beta_{id}{}^{-1} \cdot r \cdot \sum_{\substack{j \in [l] \\ \rho^*(j) \notin \mathbf{A}}} \frac{b_j}{A_i - \rho^*(j)}$$

$$= \tilde{r}_i + \beta_{id}{}^{-1} \cdot (\tilde{r} \cdot \sum_{\substack{j \in [l] \\ \rho^*(j) \notin \mathbf{A}}} \frac{b_j}{A_i - \rho^*(j)} + \sum_{\substack{j' \neq j \in [n,l] \\ \rho^*(j) \notin \mathbf{A}}} \frac{w_{j'} b_j a^{q+1-j'}}{A_i - \rho^*(j)}),$$

and then computes $sk_{i,1} = (u^{A_i} h)^{r_i} v^{-r}$, $sk_{i,2} = g^{r_i}$ as required, of which the details about the calculation are similar to that in [6] and we omit them there.

- Challenge. Algorithm $\mathcal{A}$ sends algorithm $\mathcal{B}$ two messages M_0^* and M_1^*. Algorithm $\mathcal{B}$ randomly chooses a bit $\beta \in \{0, 1\}$, and computes the challenge ciphertext as that in [6].
- Guess. Algorithm $\mathcal{A}$ output a guess β' for β.

If $Z = \hat{e}(g, g)^{s\alpha^{a^{q+1}}}$, then the view of algorithm $\mathcal{A}$ about this simulation is identical to the original game, because $C^* = M_\beta^* \cdot Z \cdot \hat{e}(g, g^s)^{\tilde{\alpha}} = M_\beta^* \cdot \hat{e}(g, g)^{\alpha s}$. On the other hand, if Z is a random term of G_1, then all the information about the message M_β^* is hidden in the challenge ciphertext. Therefore the advantage of algorithm $\mathcal{A}$ is 0. As a result, if algorithm $\mathcal{A}$ breaks the selective security with a non-negligible advantage, then algorithm $\mathcal{B}$ has a non-negligible advantage in breaking the $(q-1)$ assumption.

Lemma 2. *The SoK protocol above is a secure signature of knowledge of a witness* $(\beta_{id}, sk_1, sk_2, sk_3)$ *in the random oracle model.*

Proof. Since the *completeness* of SoK is straightforward, we focus on its *soundness* and *zero-knowledge*.

Soundness. Assume there are two transcripts with the same (B, R_0, R_1, R_2) but different challenges c', c and different responses (θ_0', θ_1') and (θ_0, θ_1).

Then (β_{id}, sk_1, sk_2) can be extracted from

$$R_0 = g^s = g^{\frac{\theta_0' - \theta_0}{c - c'}}, \quad B = H_0(\mathrm{CT})^{\beta_{id}} = H_0(\mathrm{CT})^{\frac{\theta_1' - \theta_1}{c - c'}},$$

$$R = \hat{e}(w, g_2)^{-s} \hat{e}(g_1, g)^s Z^{\beta_{id}} = \hat{e}(w, g_2)^{-\frac{\theta_0' - \theta_0}{c - c'}} \hat{e}(g, g_1)^{\frac{\theta_0' - \theta_0}{c - c'}} Z^{\frac{\theta_1' - \theta_1}{c - c'}}$$

Zero-knowledge. The simulator randomly chooses $R_0, R_1, R_2 \in G$, $\theta_0, \theta_1 \in Z_p$, $c \in Z_p$, and computes

$$K_0 = R_0{}^c g^{\theta_0}, \quad R = \frac{\hat{e}(R_1, g)}{\hat{e}(w, R_2)}, \quad K_2 = B^c H_0(\mathrm{CT})^{\theta_1},$$
$$K_1 = R^c \hat{e}(w, g_2)^{-\theta_0} \hat{e}(g, g_1)^{\theta_0} Z^{\theta_1}.$$

Then it sets $c = H(B, R_0, R_1, R_2, K_0, K_1, K_2, \mathrm{CT})$.

Appendix B. Performance evaluation of an existing scheme

Also, we implement the scheme given in [2], where expressive access policies are not supported and the number of data users allowed is predefined and bounded. From the analysis in Table 2, despite of the limitations in real applications, it is straightforward to conclude that the scheme in [2] is more efficient than our proposed schemes. To confirm such a conclusion, we test the average time of running the Encrypt and Decrypt algorithms in the scheme in [2] and our two proposed schemes in terms of the curves SS512 and MNT159, respectively (see Fig. 5). It is not difficult to see that the experimental results are consistent with the theoretical results where the average computation time of both encryption and decryption in [2] (which does not support expressiveness in access control) is independent to the complexity of the access structures (and the number of attributes involved), while that in the proposed two schemes is linear to the complexity of the access structures (and the number of attributes involved).

References

[1] R. Lu, X. Lin, X. Liang, X.S. Shen, Secure provenance: the essential of bread and butter of data forensics in cloud computing, in: Proceedings of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS 2010, Beijing, China, April 13-16, 2010, ACM, 2010, pp. 282–292.

[2] S.S.M. Chow, C. Chu, X. Huang, J. Zhou, R.H. Deng, Dynamic secure cloud storage with provenance, in: Cryptography and Security: From Theory to Applications - Essays Dedicated to Jean-Jacques Quisquater on the Occasion of His 65th Birthday, in: Lecture Notes in Computer Science, vol. 6805, Springer, 2012, pp. 442–464.

[3] J. Li, X. Chen, Q. Huang, D.S. Wong, Digital provenance: Enabling secure data forensics in cloud computing, Future Gener. Comput. Syst. 37 (2014) 259–266.

[4] D.M. Freeman, Converting pairing-based cryptosystems from composite-order groups to prime-order groups, in: Advances in Cryptology - EUROCRYPT 2010, 29th Annual International Conference on the Theory and Applications of Cryptographic Techniques, French Riviera, May 30 - June 3, 2010. Proceedings, in: Lecture Notes in Computer Science, vol. 6110, Springer, 2010, pp. 44–61.

[5] A. Sahai, B. Waters, Fuzzy identity-based encryption, in: Advances in Cryptology - EUROCRYPT 2005, 24th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Aarhus, Denmark, May 22–26, 2005, Proceedings, in: Lecture Notes in Computer Science, vol. 3494, Springer, 2005, pp. 457–473.

[6] Y. Rouselakis, B. Waters, Practical constructions and new proof methods for large universe attribute-based encryption, in: 2013 ACM SIGSAC Conference on Computer and Communications Security, CCS'13, Berlin, Germany, November 4–8, 2013, ACM, 2013, pp. 463–474.

[7] A. Boldyreva, V. Goyal, V. Kumar, Identity-based encryption with efficient revocation, in: Proceedings of the 2008 ACM Conference on Computer and Communications Security, CCS 2008, Alexandria, Virginia, USA, October 27–31, 2008, ACM, 2008, pp. 417–426.

[8] V. Goyal, O. Pandey, A. Sahai, B. Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, Ioctober 30 - November 3, 2006, in: Lecture Notes in Computer Science, vol. 5126, Springer, 2006, pp. 89–98.

[9] J. Bethencourt, A. Sahai, B. Waters, Ciphertext-Policy attribute-based encryption, in: 2007 IEEE Symposium on Security and Privacy (S&P 2007), 20–23 May 2007, Oakland, California, USA, IEEE Computer Society, 2007, pp. 321–334.

[10] L. Cheung, C.C. Newport, Provably secure ciphertext policy ABE, in: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007, Alexandria, Virginia, USA, October 28–31, 2007, ACM, 2007, pp. 456–465.

[11] V. Goyal, A. Jain, O. Pandey, A. Sahai, Bounded ciphertext policy attribute based encryption, in: Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7–11, 2008, Proceedings, Part II - Track B: Logic, Semantics, and Theory of Programming & Track C: Security and Cryptography Foundations, in: Lecture Notes in Computer Science, vol. 5126, Springer, 2008, pp. 579–591.

[12] R. Hasan, R. Sion, M. Winslett, Introducing secure provenance: problems and challenges, in: Proceedings of the 2007 ACM Workshop on Storage Security and Survivability, StorageSS 2007, Alexandria, VA, USA, October 29, 2007, ACM, 2007, pp. 13–18.

[13] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: CRYPTO, in: Lecture Notes in Computer Science, vol. 2139, Springer-Verlag, 2001, pp. 213–219.

[14] D. Boneh, X. Boyen, Short signatures without random oracles and the SDH assumption in bilinear groups, J. Cryptology 21 (2) (2008) 149–177.

[15] A.B. Lewko, B. Waters, Decentralizing attribute-based encryption, in: Advances in Cryptology - EUROCRYPT 2011 - 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15–19, 2011. Proceedings, in: Lecture Notes in Computer Science, vol. 6632, Springer, 2011, pp. 568–588.

[16] B. Waters, Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization, in: Public Key Cryptography - PKC 2011 - 14th International Conference on Practice and Theory in Public Key Cryptography, Taormina, Italy, March 6–9, 2011. Proceedings, in: Lecture Notes in Computer Science, vol. 6571, Springer, 2011, pp. 53–70.

[17] A. Beimel, Secure Schemes for Secret Sharing and Key Distribution (Ph.D. thesis), Israel Institute of Technology, Israel Institute of Technology, 1996.

[18] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof systems, SIAM J. Comput. 18 (1) (1989) 186–208.

[19] J. Camenisch, M. Stadler, Efficient group signature schemes for large groups (extended abstract), in: Advances in Cryptology - CRYPTO '97, 17th Annual International Cryptology Conference, Santa Barbara, California, USA, August 17–21, 1997, Proceedings, in: Lecture Notes in Computer Science, vol. 1294, Springer, 1997, pp. 410–424.

[20] A. Fiat, A. Shamir, How to prove yourself: practical solutions to identification and signature problems, in: CRYPTO, 1986, pp. 186–194.

[21] N. Attrapadung, H. Imai, Attribute-based encryption supporting direct/indirect revocation modes, in: Cryptography and Coding, 12th IMA International Conference, Cryptography and Coding 2009, Cirencester, UK, December 15–17, 2009. Proceedings, in: Lecture Notes in Computer Science, vol. 5921, Springer, 2009, pp. 278–300.

[22] A. Sahai, H. Seyalioglu, B. Waters, Dynamic credentials and ciphertext delegation for attribute-based encryption, in: Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19–23, 2012. Proceedings, in: Lecture Notes in Computer Science, vol. 7417, Springer, 2012, pp. 199–217.

[23] H. Cui, R.H. Deng, Y. Li, B. Qin, Server-aided revocable attribute-based encryption, in: Computer Security - ESORICS 2016 - 21st European Symposium on Research in Computer Security, Heraklion, Greece, September 26–30, 2016, Proceedings, Part II, in: Lecture Notes in Computer Science, vol. 9879, Springer, 2016, pp. 570–587.

[24] D. Naor, M. Naor, J. Lotspiech, Revocation and tracing schemes for stateless receivers, in: Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19–23, 2001, Proceedings, in: Lecture Notes in Computer Science, vol. 2139, Springer, 2001, pp. 41–62.

[25] J.A. Akinyele, C. Garman, I. Miers, M.W. Pagano, M. Rushanan, M. Green, A.D. Rubin, Charm: a framework for rapidly prototyping cryptosystems, J. Cryptographic Engineering 3 (2) (2013) 111–128.

Hui Cui received her Ph.D. degree in the School of Computing and Information Technology, University of Wollongong, Australia. She is currently a research fellow in the Secure Mobile Centre under the School of Information Systems, Singapore Management University, Singapore.

Robert H. Deng has been a Professor at the School of Information Systems, Singapore Management University since 2004. Prior to this, he was Principal Scientist and Manager of Infocomm Security Department, Institute for Infocomm Research, Singapore. His research interests include data security and privacy, multimedia security, network and system security. He has served/is serving on the editorial boards of many international journals in security, such as IEEE Transactions on Information Forensics and Security, IEEE Transactions on Dependable and Secure Computing, the International Journal of Information Security, and IEEE Security and Privacy Magazine. He is the chair of the Steering Committee of the ACM Asia Conference on Computer and Communications Security (ASIACCS). He received the University Outstanding Researcher Award from the National University of Singapore in 1999 and the Lee Kuan Yew Fellow for Research Excellence from the Singapore Management University in 2006. He was named Community Service Star and Showcased Senior Information Security Professional by (ISC)2 under its Asia-Pacific Information Security Leadership Achievements program in 2010. He is Fellow of IEEE.

Yingjiu Li is currently an Associate Professor in the School of Information Systems at Singapore Management University (SMU). His research interests include RFID Security and Privacy, Mobile and System Security, Applied Cryptography and Cloud Security, and Data Application Security and Privacy. He has published over 130 technical papers in international conferences and journals, and served in the program committees for over 80 international conferences and workshops. Yingjiu Li is a senior member of the ACM and a member of the IEEE Computer Society. The URL for his web page is http://www.mysmu.edu/faculty/yjli/.