Future Generation Computer Systems 37 (2014) 259–266

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Digital provenance: Enabling secure data forensics in cloud computing

Jin Li [a,*], Xiaofeng Chen [b], Qiong Huang [c], Duncan S. Wong [d]

[a] School of Computer Science, Guangzhou University, Guangzhou, China
[b] State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China
[c] College of Informatics, South China Agricultural University, Guangzhou, China
[d] The Department of Computer Science, City University of Hong Kong, Hong Kong

HIGHLIGHTS

- We propose a practical secure provenance scheme with fine-grained access control.
- A broadcast encryption technique is utilized to decrease the data owner's computational overhead.
- An attribute-based signature is applied to realize efficient anonymous authentication.

ARTICLE INFO

Article history:
Received 27 March 2013
Received in revised form
10 August 2013
Accepted 4 October 2013
Available online 18 October 2013

Keywords:
Provenance
Cloud computing
Privacy
Attribute-based signature

ABSTRACT

Secure provenance that records the ownership and process history of data objects is vital to the success of data forensics in cloud computing. In this paper, we propose a new secure provenance scheme based on group signature and attribute-based signature techniques. The proposed provenance scheme provides confidentiality on sensitive documents stored in a cloud, unforgeability of the provenance record, anonymous authentication to cloud servers, fine-grained access control on documents, and provenance tracking on disputed documents. Furthermore, it is assumed that the cloud server has huge computation capacity, while users are regarded as devices with low computation capability. Aiming at this, we show how to utilize the cloud server to outsource and decrease the user's computational overhead during the process of provenance. With provable security techniques, we formally demonstrate the security of the proposed scheme under standard assumptions.

1. Introduction

Cloud computing is a promising next-generation computing paradigm which integrates multiple existing and new technologies such as virtualization and distributed computing. It provides unlimited "virtualized" resources to users as services across the Internet while abstracting the details from users. With the emergence of commercial cloud computing platforms such as Amazon's EC2 and S3 [1], Google's App Engine [2], and Microsoft's Azure [3], cloud computing has become more a reality than just a concept [4].

As in any existing application and system, security and privacy play an extremely important role for the success of cloud computing, and certainly raise a lot of challenges among the many others that cloud computing is confronted with. It is hard to imagine that a cloud customer, say a company, would like to store all its sensitive information on cloud computing platforms, e.g., Amazon's S3, and put the security protection of the information at the mercy of the cloud computing operator.

Besides the confidentiality of this sensitive information, the user's identity privacy, a fundamental right to privacy, is also expected in cloud computing. If the access to a cloud discloses a user's real identity, the user could still be unwilling to accept this paradigm. Thus, anonymous authentication [5] is desirable in cloud computing. Although anonymous authentication can provide privacy of a user's identity, it is required to only provide conditional anonymity. For example, when a group of users is authorized to access a document, if some dispute arises in a modification, the real user can be tracked by some designated party.

The provenance systems [6–8] have been developed to record provenance meta-data. Given its provenance, a data object can report who created and who modified its contents. Practical provenance systems use a specialized recording instrument to collect information about data processing at runtime. The instrument annotates data with information on the relevant

* Corresponding author.
E-mail addresses: jin71@gmail.com, jinli71@gmail.com (J. Li), xfchen@xidian.edu.cn (X. Chen), csqhuang@cityu.edu.hk (Q. Huang), duncan@cityu.edu.hk (D.S. Wong).

0167-739X/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.future.2013.10.006

operations performed on it. The ordered collection of provenance annotations becomes an unalterable record of data evolution called a provenance chain. Therefore, once a dispute arises in a document stored in a cloud, provenance is important for data forensics to provide digital evidences for post investigation. Provenance information has a wide range of critical application areas. For example, scientific data processing needs to keep track of data ownership and processing workflow to ensure the trust assigned to the output data. In business environments, provenance of documents is even more critical for regulatory and legal reasons. A company's financial reports are required to contain provenance information on the path the data took during various stages of processing and the principals who performed various actions on it.

Therefore, cloud computing should also provide provenance [9] to record the ownership and process history of data objects in the cloud in order to gain wide acceptance to the public. However, there are many challenges to provenance in cloud computing [8], in which we need protect the security of provenance information, i.e., to not violate the information confidentiality and user privacy in cloud computing. Specifically, these requirements [9] include confidentiality of documents, unforgeability of the provenance record, and conditional anonymity of the user's identity.

Though secure provenance is vital to the success of data forensics in cloud computing, before its deployment in cloud computing, two critical issues have to be addressed, namely, (1) fine-grained access control: when a document is being created, the data owner can specify a fine-grained access control policy for the documents stored remotely in the cloud servers; (2) low computation and communication overhead at the data owner/user side: in cloud computing, the computational ability is not required to be high except for the cloud server. Actually, the devices are always assumed to be devices with low computational capability. Thus, a provenance system with low computation for data owners and users is preferred in cloud computing.

Aiming at this, we propose a practical secure provenance scheme with fine-grained access control based on the bilinear pairing technique in this paper, which can provide trusted evidence for data forensics in cloud computing. Our contribution in this paper is as follows.

(1) The computation and communication overhead for the data owner is low. Compared with the previous work [5], two new techniques are utilized here to decrease the data owner's computational overhead. The first is broadcast encryption, which is used by the cloud server to control the user's access. The other is the attribute-based signature, which is computed by users, instead of data owners, as part of their access requests.
(2) The computational overhead for the data owner/user has been significantly reduced by outsourcing the cryptographic operation of exponentiation in a bilinear group. More specifically, the computation is moved from the data owner/user side to the cloud server by using the following two techniques. The first is to use the two-server model [10] to compute the exponentiation cooperatively. The second is to use the proxy re-signature method [11]. As a result, we significantly reduce the complexity at the user/data owner side with respect to the computation of modular exponentiation from $O(k)$ to $O(1)$ in terms of the number of modular multiplications required [12], where k represents the number of bits of the exponent.

1.1. Organization

The rest of the paper is organized as follows. In Section 2, we present the related work for a secure provenance system. In Section 3, the architecture and the security model for a secure provenance system are given. In Section 4, we show some basic tools which will be used in this paper, which include the attribute-based signature scheme and the group signature scheme. In Section 5, a new and efficient secure provenance scheme is given, as well as its security analysis. We also discuss how to provide fine-grained access control and better efficiency in this section. Finally, we draw our conclusions in Section 6.

2. Related work

Provenance has been studied extensively in archival theory for the purpose of asserting authenticity. In recent years, provenance has also gained importance in digital realms and e-Science [13,14]. However, most schemes require trustworthiness of the server. Provenance systems that do not rely on a trusted server have also been developed [15].

Although provenance of workflow and documents has been studied extensively in the past, very little work has been done on securing the provenance information. To address such security issues in provenance, Hasan et al. [8] first formally defined the security and privacy issues of a provenance system, including confidentiality and user privacy.

Recently, Lu et al. [5] proposed a new secure provenance system to achieve user privacy and message confidentiality. The basic tool they used is the group signature technique. For the first time, they showed how to achieve both user anonymity and message confidentiality efficiently. However, the system can only support a simple access policy, that is, only one attribute is issued to each user. It is critical to achieve a fine-grained access control system because such a system facilitates granting differential access rights to a set of users and allows flexibility in specifying the access rights of individual users.

One naive approach to support provenance in cloud computing is as follows. Each user registers to a third party and is issued a certificate of the group signature to achieve conditional anonymous authentication. Each data owner encrypts each of his/her document sets with a broadcast encryption by including users owning corresponding privilege. The data owner also sends a broadcast decryption key to each valid user for each document set. As a result, when there is a user to be revoked, the data owner has to update by re-computing all the broadcast encryptions associated with this revoked user. Thus, such a naive approach is not practical, especially in cloud computing.

The notion of attribute-based encryption (ABE) [16,17], which stemmed from fuzzy identity-based encryption proposed by Sahai and Waters [18], enables for the first time public key based one-to-many encryption with fine-grained access control. Therefore, it is envisioned as a highly promising public key primitive for realizing scalable and fine-grained access control systems, where differential yet flexible access rights can be assigned to individual users. To address a complex and general access policy, two kinds of ABE have been proposed [16]: key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP-ABE). In KP-ABE, the access policy is assigned in an attribute private key, whereas, in CP-ABE, the access policy is specified in the ciphertext.

Attribute-based signature (ABS) [19–22] is also proposed to realize fine-grained access control in anonymous authentication systems, in which a signer is defined by a set of attributes instead of a single string representing the signer's identity. Compared with ABE, it does not require interaction between two participants to realize the access control. In ABS, a user obtains a certificate for a set of attributes from an attribute-certification authority known as the attribute authority. An attribute-based signature assures the verifier that a signer, whose set of attributes satisfies a (possibly) complex predicate, has endorsed the message.

Though both ABE and ABS can be used to provide a fine-grained access control system, they cannot revoke the user anonymity and find out the user's identity in a provenance record when a dispute arises. Thus, the provenance system constructed directly from ABE

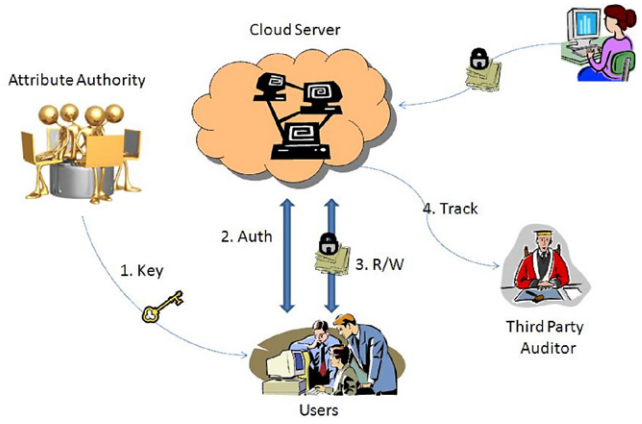

Fig. 1. Architecture of the provenance system.

or ABS is not sufficient to provide the security required in [5,8]. One may argue that the following straightforward approach can be used to achieve conditional anonymity and fine-grained access control. The users first register to use the cloud computing system. They are issued certificates for a group signature from a third party and attribute private keys from attribute authorities. The user also uses the group signature to perform conditional anonymous authentication to enter the cloud computing system. The data owner uses ABE to encrypt each document set with the same access control policy. If the authentication passes, the users are allowed to access encrypted files of each data owner. As a result, the data owner loses the control of his/her own privileged user set because all of the users in the system with such attributes can access his data. However, such an issue does not exist because of the broadcast encryption technique in the above naive approach.

3. Provenance with multiple authorities

3.1. System model

In this paper, we consider a cloud data system consisting of data owners W, data users U, a cloud server, attribute authorities $\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_N$, and a third-party auditor TPA; see Fig. 1. W stores his/her sensitive data on the cloud server. U is issued attributes from $\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_N$. To access and operate the remote stored data documents shared by W, user U needs to show his/her access privilege to the cloud server. The cloud server is always online and is operated by the Cloud Service Provider (CSP). The cloud server is assumed to have abundant storage capability and computation power. The TPA is used for tracing the identity of a dishonest user when a dispute in some document arises. In addition, we assume that the owner W can not only store data but also run his/her own code on the cloud server to manage his/her data.

3.2. Design goals

In this paper, we address the problem of enabling efficient provenance of data files stored on cloud servers, while achieving user privacy, a fine-grained access policy, and data confidentiality. Specifically, we want to enable the data owner to enforce an access structure on each file, which precisely designates the set of operations that the user is allowed to perform. Cloud servers are prevented from learning the plaintexts of data files and the identity of users who access and operate on the files. All these security goals should be achieved efficiently in the sense that the system is scalable.

In summary, a secure and practical provenance system in cloud computing includes the following five properties: (1) confidentiality: the content of documents should be kept secret from unauthorized users, including the cloud server; (2) unforgeability: a provenance record attests the ownership and process history of data files, which cannot be forged by unauthorized users; (3) anonymity: the user privacy of the provenance should be protected from the cloud server; (4) traceability: the user's identity can be traced from any disputed provenance record; (5) fine-grained access control: the data owner can specify a fine-grained access control policy over documents stored in the cloud server; (6) low computational overhead at the data owner/user side: in cloud computing, only limited computational capability is assumed for data owners.

3.3. Access structure

In our system, each document is labeled by data owner W with an access structure that specifies which types of user are allowed access to the document. In this paper, the access structure is described by a tree T. The tree-access structure can be regarded as a generalization of traditional threshold secret sharing, where a secret is divided into many shares such that only users with enough shares can reconstruct the secret. In a tree-access structure, each non-leaf node consists of AND and OR gates. In other words, these nodes are described by their children and a threshold value to represent a threshold gate. We give some notation and functions for convenience. Let num_x be the number of children of node x and let k_x be its threshold value, where $0 < k_x \leq num_x$. We denote the parent of node x in the tree by $parent(x)$. Denote by T_x the subtree of T rooted at node x. Denote the relation $R(T_x, \omega) = 1$ if a set of attributes ω satisfies the access tree T_x. In our paper, the access structure is specified in a private key. We note that this setting is reminiscent of secret sharing schemes. For example, one can specify a tree-access structure in which the interior nodes consist of AND and OR gates and the leaves consist of different attributes. Any user is allowed to access a document if the attributes satisfy the access structure specified.

3.4. Basic framework

We denote $\mathbb{A} = \{\mathbb{A}_k\}$ as the attributes associated with a user, where each $\mathbb{A}_k$ is a set of values $\{v_j\}$ and v_j is the value corresponding to the jth attribute. For example, $v_1 =$ "Male" and $v_2 = 29$, where the first attribute is about gender and the second one is about age. We restrict ourselves to the case that an user is entitled to only one value corresponding to each attribute.

Recall that, in the definition of digital provenance, we have many attribute authorities $\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_N$ and many users. The TPA is responsible for the traceability. Each A_i is responsible for the issue of a disjoint set of attributes. Each user U is entitled to a number of attributes and can obtain a key corresponding to those attributes from the attribute authorities. W can update documents accompanied with specified access structure, which is associated with attributes.

Definition 1. A provenance system with N-authority consists of five algorithms, which are described as follows.

- Via $(\mathsf{params}, \{(apk_k, ask_k)\}_{k\in\{1,\ldots,N\}}) \leftarrow \mathsf{Setup}$, the randomized key generation algorithm takes a security parameter $\lambda \in \mathbb{N}$ and the number of authorities $N \in \mathbb{N}$, and outputs the system parameters params and N public/private key pairs (apk_k, ask_k), one for each attribute authority A_k, where $k \in \{1, \ldots, N\}$. Furthermore, the public and private key (tpk, tsk) of TPA is also returned as the output of Setup. For simplicity, we assume that params and $(\{apk_k\}_{k\in\{1,\ldots,N\}}, tpk)$ are the implicit inputs of the rest of the algorithms.

- Via $usk_k[\mathsf{GID}, \mathbb{A}_k] \leftarrow \mathsf{AKeyGen}(ask_k, \mathsf{GID}, \mathbb{A}_k)$, the attribute authority A_k uses its secret key ask_k and runs the algorithm AKeyGen to output a signing key corresponding to the attribute set $\mathbb{A}_k$ for the user with identity GID.
- Via $\sigma \leftarrow \mathsf{AnoAuth}_R(\{\mathbb{P}_k, usk_k[\mathsf{GID}, \mathbb{A}_k]\}_{k\in\{1,\ldots,N\}}, m)$, a user signs a message m for the policy $\{\mathbb{P}_k\}$, resulting in a signature, where $\mathbb{P}_k$ denotes a subset of the attribute domain responsible by the authority k, and R is a predicate indicating how the policy is related to $\{\mathbb{A}_k\}_{k\in\{1,\ldots,N\}}$.
- Via $1/0 \leftarrow \mathsf{Ver}(\{\mathbb{P}_k\}_{1\le k\le N}, m, \sigma)$, the cloud server verifies if the user possesses a matching set of private keys $\{usk_k[\mathsf{GID}, \mathbb{A}_k]\}$ from each authority k. The user and cloud server achieve fine-grained authorized access to a document after a successful anonymous authentication if the user's privilege satisfies the access policy specified by the document. The user will be allowed to operate on (i.e., create, modify) this document in cloud computing systems attested with a provenance signature generated by the user. Later, anyone can check the validity, but only authorized users can read the document.
- Via $\mathsf{GID} \leftarrow \mathsf{Tracking}(\sigma, tsk)$, the TPA takes as input the data stored by the cloud server, which are appended with the disputed document. Furthermore, it also takes as input TPA's private key tsk and runs the Tracking algorithm. The output of this algorithm is the identity of a user whose operation is disputed.

3.5. Security models

In this work, we just consider an honest but curious cloud server. That is to say, the cloud server will follow our proposed protocol, but will try to find out as much secret information as possible based on the input. More specifically, we assume that the cloud server is more interested in records of operations and contents of data files stored. A secure communication channel between data owners/users and the cloud server is required. Users will try to access data files either within or out of the scope of their access privilege. Two kinds of attacker are considered in this system. (1) An external attacker, including the server, revoked users, and other unauthorized users. In this paper, we do not consider collision attacks between the cloud server and users because the cloud server is assumed to be honest. We also do not consider an attack of sharing secrets among users, which is also difficult to prevent in other cryptographic protocols. (2) An internal attacker, i.e., a legal user who does not obey the rules. An internal attacker could try to access documents with a policy which his/her attributes do not satisfy.

There are four security requirements for a secure provenance system: confidentiality, unforgeability, anonymity, and traceability. The definition of confidentiality requires that any unauthorized users cannot get any information about the documents stored in cloud servers.

The definition for unforgeability requires that any user cannot pretend to be a legal user and generate a signature for some attributes if he/she does not have such privileges. There are two oracles provided to the adversary: a private key extraction oracle and a signing oracle. The definition for unforgeability also implies security against collusion attacks, in which a group of users could combine their secret keys and sign a message with attributes they could not do individually.

For anonymity, we require that the signer is kept anonymous among users with the same attributes in a signature, even to the attribute authorities. Such a security definition is also defined in attribute-based signature schemes. We will adopt the same security definition as defined in attribute-based signatures in our provenance security definition.

For traceability, we require that there is an entity to reveal the real identity recorded in the provenance when a dispute arises in a document.

4. Basic tools

4.1. Pairing

Let $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p. Pairing $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_T$ is a bilinear map with the following properties.

- *Bilinearity*: $\hat{e}(g_1^a, g_1^b) = \hat{e}(g_1, g_1)^{ab}$ for all $a, b \in \mathbb{Z}_p^*$.
- *Non-degeneracy*: $\hat{e}(g_1, g_1) \neq 1$.
- *Computability*: It is efficient to compute $\hat{e}(g_1, g_2)$ for all $g_1, g_2 \in \mathbb{G}_1$.

4.2. Attribute-based signature with multi-authority

An ABS scheme consists of four algorithms: a setup algorithm Setup, private key extraction algorithm Extract, signing algorithm Sign, and verification algorithm Verify. We describe the ABS scheme with multiple authorities proposed in [22]. A basic (d_k, m_k)-threshold access control policy is supported, where d_k is a number predefined by each A_k. Define the Lagrange coefficient $\Delta_{i,S}$ for $i \in \mathbb{Z}_p$ and a set, S, of elements in $\mathbb{Z}_p$ as follows:

$$\Delta_{i,S}(x) = \prod_{j\in S, j\neq i} \frac{x-j}{i-j}.$$

Setup. Define two hash functions $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$. For each authority A_k, choose x_k as his/her secret key and $T_k = \hat{e}(g_1, g_2)^{x_k}$ as his/her public key, which is also sent to the other authorities. $T = \prod_{k=1}^{N} T_k$ is also computed and published as a public key. Authority A_k, where $1 \le k \le N$, also shares a secret pseudorandom function (PRF) seed $s_{kk'}$ with each other authority $\mathsf{A}_{k'}$. A_k also defines a PRF seed a_k and computes $y'_k = g_1^{a_k}$, which are sent to all the other authorities. For a user with identity GID, define a pseudorandom function $\mathrm{PRF}_{kk'}(\mathsf{GID}) = g_1^{a_k a_{k'}/s_{kk'}+\mathsf{GID}}$.

Extract. To get an attribute private key for an attribute set $\mathbb{A}_k = (v_{k,1}, v_{k,2}, \ldots, v_{k,n_k})$ from A_k where $1 \le k \le N$, the user with identity GID first gets D_{kj} for $k \neq j$ by using the anonymous key-issuing protocol [22], as shown in the algorithm of key issuing in Fig. 1.

Furthermore, A_k randomly picks a degree-d_k polynomial $p_k(\cdot)$ with $p_k(0) = x_k - \sum_{k'\in\{1,2,\ldots,N\}\setminus\{k\}} R_{kk'} \bmod p$. The attribute private key for each eligible attribute $v_{k,i}$ of $\mathbb{A}_k$ is computed as $(D_{k,i,0}, D_{k,i,1}) = (g_1^{p_k(i)} H_1(v_{k,i})^{r_i}, g_1^{r_i})$.

Sign. Suppose a user is issued a private key $(D_{k,i,0}, D_{k,i,1})$ for $\mathbb{A}_k$, where $1 \le k \le N$. To sign a message m with the proof of owning d_k attributes in an m_k-element attribute set $\mathbb{A}_k^*$ for $1 \le k \le N$, he/she selects a d_k-element subset $\mathbb{A}'_k$ and m_k random values $r'_i \in \mathbb{Z}_p$ for $i \in \mathbb{A}_k^*$. He/she computes $\sigma_0 = \prod_{k=1}^{N} [\prod_{i\in\mathbb{A}'_k} D_{k,i,0}^{\Delta_{i,S}(0)} \prod_{i\in\mathbb{A}_k^*} H_1(i)^{r'_i}]\, DH_2(m)^s$, $\{\sigma_{k,i} = D_{k,i,1}^{\Delta_{i,S}(0)} g_1^{r'_i}\}_{i\in\mathbb{A}'_k}$, $\{\sigma_{k,i} = g_1^{r'_i}\}_{\mathbb{A}_k^*\setminus\mathbb{A}'_k}$, and $\sigma'_0 = g_1^s$, with a randomly chosen value $s \in \mathbb{Z}_p$. He/she outputs the signature $\sigma = (\sigma_0, \{\{\sigma_i\}_{i\in\mathbb{A}_k^*}\}_{k\in\{1,2,\ldots,N\}}, \sigma'_0)$.

Verify. After receiving the signature $\sigma = (\sigma_0, \{\sigma_i\}_{i\in\mathbb{A}_k^*}, \sigma'_0)$ of message m with threshold d_k, check if the following equation holds:

$$\frac{\hat{e}(g_1, \sigma_0)}{\left[\prod_{1\le k\le N} \prod_{i\in\mathbb{A}_k^*} \hat{e}(H_1(i), \sigma_i)\right] \hat{e}(H_2(m), \sigma'_0)} \stackrel{?}{=} T.$$

4.3. Group signature

Group signature, introduced by Chaum and van Heyst [23], allows any member of a group to sign on behalf of the group. Anyone can verify the signature with a group public key while no one

Setup Define a bilinear group $\mathbb{G}_1$ of order p with a generator g_1, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$, with the properties of *Bilinearity* and *Non-degeneracy*. Let $H_1, H_2 : \{0,1\}^* \to \mathbb{Z}_p^*$ be two cryptographic hash functions. For each authority A_k, choose x_k as his secret key and $T_k = \hat{e}(g_1, g_1)^{x_k}$ as his public key, which is also sent to the other authorities. This algorithm also selects $h \in \mathbb{G}_1$ and $\xi_1, \xi_2 \in \mathbb{Z}_p$, and sets $u, v \in \mathbb{G}_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. The system public parameter is $\mathsf{params} = (g_1, u, v, h, T = \prod_{k=1}^{N} T_k, H_1, H_2)$. The private key of TPA is $gmsk = (\xi_1, \xi_2)$.

Key Issuing For each A_k, U starts an independent invocation of the anonymous protocol [26] for $(y_j^{a_k}, g_1, \delta_{kj}R_{kj}, s_{kj}, \delta_{kj})$ where R_{kj} is randomly chosen by A_k from $\mathbb{Z}_p^*$. δ_{kj} is defined as 1 if $k > j$ and -1 otherwise. Finally, U obtains $D_{kj} = g_1^{R_{kj}}\,\mathrm{PRF}_{kj}(\mathsf{GID})$ if $k > j$. Otherwise, U gets $D_{kj} = g_1^{R_{kj}}/\mathrm{PRF}_{kj}(\mathsf{GID})$. With D_{kj} for $1 \le k \le N$, the user can compute $D = g_1^R$, where $R = \prod_{k,k' \in \{1,2,\cdots,N\} \times \{\{1,2,\cdots,N\} \setminus \{k\}} R_{kk'}$.

Fig. 2. Algorithm description [26].

can know the identity of the signer except the group manager. We will use the group signature scheme proposed by [24]. It consists of five algorithms: the setup algorithm GSetup, key generation algorithm GKeyGen, signing algorithm GSign, verification algorithm GVerify, and tracing algorithm Trace. We show the construction of the group signature scheme proposed by [24]. It consists of five algorithms: the setup algorithm GSetup, key generation algorithm GKeyGen, signing algorithm GSign, verification algorithm GVerify, and tracing algorithm Trace.

GSetup: This algorithm selects $h \in G_1$ and $x, \xi_1, \xi_2 \in Z_p$, computes $y = e(g_1, g_1)^x$, and sets $u, v \in G_1$ such that $u^{\xi_1} = v^{\xi_2} = h$. The group manager's private key is (x, ξ_1, ξ_2).

GKeyGen: If a user with identity ID is allowed to join this group, the group manager computes $A = g_1^{\frac{1}{x+ID}}$ and returns it to the user.

GSign: To generate a group signature with the certificate A, the user selects exponents $\alpha, \beta \in Z_p$, and computes a linear encryption of A: $T_1 = u^{\alpha}, T_2 = v^{\beta}, T_3 = Ah^{\alpha+\beta}$. He/she also computes two values $\delta_1 = x\alpha$ and $\delta_2 = x\beta$. He/she picks blinding values $r_{\alpha}, r_{\beta}, r_x, r_{\delta_1}$, and r_{δ_2} at random from $\mathbb{Z}_p$. To sign a message m, the user computes five values based on all the following: $R_1 = u^{r_{\alpha}}, R_2 = u^{r_{\beta}}, R_3 = e(T_3, g_1)^{r_x} e(h, w)^{-\alpha-\beta}\, e(h, g_1)^{-\delta_1-\delta_2}, R_4 = T_1^{r_x} u^{-r_{\delta_1}}$, and $R_5 = T_2^{r_x} v^{r_{\delta_2}}$. He/she then computes $c = H(m, T_1, T_2, T_3, R_1, R_2, R_3, R_4, R_5), s_{\alpha} = r_{\alpha} + c\alpha, s_{\beta} = r_{\beta} + c\beta, s_x = r_x + cx, s_{\delta_1} = r_{\delta_1} + c\delta_1$, and $s_{\delta_2} = r_{\delta_2} + c\delta_2$. The values of $(R_1, R_2, R_3, R_4, R_5, s_{\alpha}, s_{\beta}, s_x, s_{\delta_1}, s_{\delta_2})$ will be sent as the group signature.

GVerify: To verify the correctness of the signature, just compute $R_1' = u^{s_{\alpha}} T_1^{-c}, R_2' = u^{s_{\beta}} T_2^{-c}, R_3' = e(T_3, g_1)^{s_x} e(h, w)^{-s_{\alpha}-s_{\beta}}\, e(h, g_2)^{-s_{\delta_1}-s_{\delta_2}}\, (e(T_3, w)/e(g_1, g_1))^c, R_4 = T_1^{s_x} u^{-s_{\delta_1}}$, and $R_5 = T_2^{s_x} v^{s_{\delta_2}}$. The signature is valid if $c = H(m, T_1, T_2, T_3, R_1', R_2', R_3', R_4', R_5')$.

Trace: To trace the identity of a signature, this algorithm takes as input T_1T_2, T_3 and computes and outputs the result as $T_3/T_1^{\xi_1}T_2^{\xi_2}$.

4.4. Broadcast encryption

A broadcast encryption scheme is required in this system. Assume that $\mathsf{BE} = (KeyGen_{BE}, Enc_{BE}, Dec_{BE})$ is a broadcast encryption scheme providing revocation-scheme security against a coalition of all revoked users [25]. More specially, $KeyGen_{BE}$ is the key generation algorithm that is used to generate a long-lived key for the user, and Enc_{BE} is the encryption algorithm that is used to encrypt documents for a privileged user group G. The group G can be dynamically changing, as users can be added to or removed from G; Dec_{BE} is the decryption algorithm that is used to decrypt the ciphertext with a non-revoked secret key at the time the message was encrypted.

4.5. Symmetric encryption

We also need a secure symmetric encryption scheme in the following constructions. Assume that $\mathsf{SE} = (KeyGen_{SE}, Enc_{SE}, Dec_{SE})$ is a symmetric encryption scheme, where $KeyGen_{SE}$ is the setup algorithm with a predefined security parameter λ, and Enc_{SE} and Dec_{SE} are the encryption and decryption algorithms, respectively.

5. Our proposed scheme

5.1. The construction

In this section, we show the construction of the proposed provenance system based on ABS. One of the reasons that we use ABS instead of ABE is to reduce the computational overhead of the data owner, which will be explained in the next section. For simplicity, we provide the construction with a basic (d_k, m_k)-threshold access control policy, where d_k is some prefixed number for each authority A_k. We will also show how to improve this construction and support a more fine-grained tree structure. In our construction, we will also use message authentication code $\mathsf{MAC}_k(\cdot)$, where k is a secret key. Such message authentication code can be constructed from the hash function with a secret key.

Setup: For N authorities $\mathsf{A}_1, \mathsf{A}_2, \ldots, \mathsf{A}_N$, A_k is in charge of the issue of attribute subset with n_k attributes, for $1 \le k \le N$. Let the value set for attribute i of authority A_k be $V_{k,i}$. Each attribute is a multi-value set. For example, the attribute "year" in a system can be defined as a multi-value set $\{2000, 2001, \ldots, 2010\}$. Then, choose a security parameter 1^{λ} and run the algorithm as shown in Fig. 2 to obtain the public parameter and the secret key. A_k also shares a secret PRF seed $s_{kk'}$ with each other authority k', where $1 \le k \le N$. A_k defines a PRF seed a_k and computes $y_k' = g_1^{a_k}$, which are sent to all the other authorities.

The TPA chooses a secret $x \in \mathbb{Z}_p$ and outputs $y = g^x$ as a public key. The public key y is sent to the cloud server.

Data owner W chooses a secret key K and encrypts all of his/her documents with $Enc(K, \cdot)$. The access control policy is also attached after each document.

For each W and valid user set S, the cloud server runs $KeyGen_{BE}$ to get the public/private key of a broadcast encryption. The server further encrypts a random number r by running the algorithm Enc_{BE} and publishes this ciphertext C_{BE}.

AKeyGen: To get an attribute private key for an attribute set $\mathbb{A}_k = (v_{k1}, v_{k2}, \ldots, v_{k,n_k})$ from authority A_k, where $1 \le k \le N$, the user with identity GID first gets D_{kj} for $k \ne j$ by using the anonymous key-issuing protocol as shown in Fig. 2.

From the TPA, a certificate A is issued to the user as described in Fig. 2. From the cloud server, a broadcast decryption key to be used in the broadcast encryption algorithm is also sent to the user.

AnoAuth: Two steps will be applied to prove that a user U is eligible to access a document.

First, U selects $x_u \in \mathbb{Z}_p$, computes $Y = g_1^{x_u}$ and runs the algorithm GSign with certificate A. The output of GSign is $(Y \parallel ts, R_1, R_2, R_3, R_4, R_5, s_{\alpha}, s_{\beta}, s_x, s_{\delta_1}, s_{\delta_2})$, which is used to show that he/she is a valid group member. Here Y and a timestamp ts are treated as a message. U also decrypts C_{BE}, the ciphertext of the broadcast encryption, to get r and sends $h' = \mathsf{MAC}_r(Y \parallel ts)$ to

the cloud server. The cloud server verifies the correctness of the signature by running GVerify and checks if $h' = MAC_r(Y \parallel ts)$. The request is accepted if the verification passes, and a signature on Y will be generated from the cloud server. Otherwise, the request will be rejected.

The user first verifies the signature from the cloud server on Y. If it is valid, then he/she can continue to further access a set of documents. Assuming that the document set is attached with access structure $\mathbb{P}$ that only allows users with at least d_k attributes in an m_k-element attribute set $\mathbb{A}_k^*$ for $1 \le k \le N$ (that is, (d_k, m_k)-threshold access structure), the user proceeds as follows.

- Suppose that U has private keys of attribute set $\mathbb{A}_k$ for $1 \le k \le N$. He/she selects a d_k-element subset $\mathbb{A}_k' \subseteq \mathbb{A}_k^* \cap \mathbb{A}_k$ and runs the algorithm Sign to generate an attribute-based signature on some randomly chosen message m and a timestamp ts.
- After receiving the signature $\sigma = (\sigma_0, \{\sigma_i\}_{i \in \mathbb{A}_k^*}, \sigma_0')$ of message m and ts with threshold d_k, the cloud server runs the algorithm Verify to check its correctness. If the output of Verify is valid, this indicates that the signature is indeed from some user with d_k attributes in $\mathbb{A}_k^*$, and the request is accepted. Otherwise, the request is rejected. If U is eligible, the cloud server will provide the encrypted documents encrypted with algorithm $Enc(K, \cdot)$. The user can decrypt with $Dec(K, \cdot)$ to get the documents. Then he/she can operate on (i.e., create, modify) the set of documents.
- After finish processing on M and getting M', U runs the symmetric encryption algorithm $C = Enc(K, M)$ and authenticates C as $\sigma' = H_1(C)^{x_u} \bmod p$. Upon receiving σ' and C, the cloud server first verifies the validity by checking $\hat{e}(\sigma', g_1) = \hat{e}(H_1(C), Y)$. Once the above equation holds, the cloud server accepts the operation, and stores (C, σ') as well as the previous provenance chain in the cloud computing system. Otherwise, the cloud server rejects (C, σ').

Tracking: If some dispute arises in a stored document, the TPA can track the dishonest user's identity as follows. The TPA first gets the group signature $(R_1, R_2, R_3, R_4, R_5, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$ from the cloud server, which is stored with (C, σ', Y). From the group signature, the identity GID can be pinpointed by utilizing algorithm Trace in the group signature, where the tracing key is the auditor's private key.

This provenance system can efficiently add or revoke users.

- Suppose that a registered user is endowed with the privilege to access data files of W. W only needs to perform as in algorithm AKeyGen. He/she sends a request with the user identity to the cloud server and asks the cloud server to include the new user in the broadcast encryption. Then, a broadcast decryption key to be used in the broadcast encryption algorithm will be sent to the user from the cloud server. Furthermore, the cloud server updates the broadcast encryption by adding this new user.
- To revoke a user, W just needs to ask the cloud server to update the valid user set of the broadcast encryption with another random number r' such that the revoked user is unable to get r'.

5.2. Supporting more fine-grained access structure

The above construction only provides the basic threshold access structure. In this section, we show how to extend it to the fine-grained access control policy denoted by a tree.

Each authority A_k chooses a polynomial $p(x)$ for each node x (including the leaves) in an access control policy tree T_k for a user U. These polynomials are chosen in a top-down manner, starting from the root node r. For each node x in the tree, denote the degree d_x of the polynomial $p(x)$ to be $d_x = k_x - 1$, where k_x is the threshold value for that node. For the root node r, set $p_r(0) = x_k - \sum_{k' \in \{1,2,\ldots,N\} \setminus \{k\}} R_{kk'} \bmod p$, and d_r other points of the polynomial p_r randomly for completeness. For another node x different from r, let $p_x(0) = p_{parent}(x)(index(x))$, and choose d_x other points randomly to completely define $p(x)$. For each polynomial, publish the additional values for verifiable secret sharing.[1]

For each leaf node x, let i be some value of attribute x, and choose $r_i \in_R \mathbb{Z}_p$. Finally, the user retains the private key $D_{ki0} = g_1^{p_x(0)} H_1(i)^{r_i}$ and $D_{ki1} = g_1^{r_i}$.

Our provenance also provides flexible threshold values d_k just by introducing a dummy attribute set as [18]. To support $1 \le d_k' \le d_k$, just issue a private key of a dummy attribute set with $d_k - d_k'$ elements.

5.3. Improved construction by outsourcing cryptographic operations

At the user side, there is a requirement to generate a group signature and an attribute-based signature. Next, we show how to reduce the computational cost at the user side by outsourcing the exponentiation g^a in group $\mathbb{G}_1$ to the cloud server, where g is the group element and a is uniformly chosen from $\mathbb{Z}_p$. Many secure outsourcing techniques have been proposed [15,28,11,29].

For the first technique [15], two independent servers are required in order to compute some exponentiation g^a. The main idea of this technique is to divide g into two parts as $(g_1 g_2)^a$, where $g = g_1 g_2$. Then compute $g_1^b g_3^c g_4^c g_2^{d+e}$, where $g_1 = g_3 g_4$ and $a = d + e$. Send $\{(g_3, c)(g_2, d)\}$ to a server, and $\{(g_4, c)(g_2, e)\}$ to another server. The two servers will compute (g_3^c, g_2^d) and (g_4^c, g_2^e) independently. g_1^b is computed in advance and stored by the user. To guarantee the validity of the results, two additional values $\{(a_1, g'), (a_2, g')\}$ will be sent to the two servers at the same time. If the results returned from these two servers are not the same, this means that at least one of the severs has not computed them honestly. From the description, it can be easily seen that the information of both g and a has been kept private. This means that neither group information nor information about the message will be leaked.

The second technique is to use the method of proxy re-signature. Suppose that the public keys of Alice and Bob are g^a and g^b, respectively. There is also a proxy who can transform Alice's signature to Bob's with a signature released by Alice, for example $H(m)^a$, where H is a hash function and m is a message. Before transformation, the proxy is given some helper secret b/a computed by Alice and Bob cooperatively. Then, with this helper secret, the signature $H(m)^a$ from Alice can be transformed to Bob's signature by computing $(H(m)^a)^{b/a} = H(m)^b$. However, the proxy cannot generate a signature on behalf of Alice or Bob for a new message which Alice or Bob have not released. This is the first time in our paper that we have pointed out that such a technique can be used to outsource exponentiation computation while keeping the exponentiation secret. The main idea of utilizing proxy re-signature to outsource exponentiation computation is as follows. The user first computes $A_0 = g^{a_0}$ and stores this value in the cloud server. When the user needs to compute any exponentiation, such as g^a, he/she computes a/a_0 and sends it to the cloud server, which functions as a proxy re-signing key. With the value of a/a_0, the cloud server can compute and return the value of $A = A_0^{a/a_0} = g^a$ to the user. The secret a is kept hidden from the server; it can be derived from the security of the proxy re-signature scheme [11]. Compared with the first technique with the two-server model, the second one cannot guarantee the validity of the result returned from server.

5.4. Efficiency and security analysis

We mainly focus on the analysis of the computational overhead at the user side because the cloud server is assumed to have huge

[1] We do not go into details here, for simplicity. It just outputs $\hat{e}(g, g_1)^{a_i}$ for the coefficient a_i in the polynomial. Please refer to [27] for details. The correctness of the private key in the following algorithm can be checked through verifiable secret sharing.

computational resource and ability. At the user side, only one key generation algorithm is required in the system for the subsequent operations. Therefore, most of the computational overhead for the user is the computation in the algorithm AnoAuth. In this algorithm, one group signing $GSig$, broadcast decryption Dec_{BE}, and symmetric encryption and decryption are required, respectively. As we have analyzed the computational cost of these algorithms above, the details are omitted here.

There are four security requirements of a provenance system. The confidentiality of documents can be derived directly because of the secure symmetric encryption scheme. External attackers cannot generate a valid group signature or valid message authentication from the broadcast encryption to get the encrypted documents. The cloud server does not know the secret K used to encrypt the documents. Internal attackers, if they do not possess the privileges specified, cannot convince the cloud server because of the unforgeability of ABS, which is given in the following theorem (Theorem 3). Thus, the cloud server will not provide the encrypted document to the user. From the above analysis, only non-revoked users with specified attributes are allowed to access the encrypted documents and decrypt with the symmetric encryption key. Note that, in our security model, we do not consider collusion attacks between the cloud server and users. Next, we show the unforgeability of the provenance scheme based on the following theorem. Note that the unforgeability of the provenance record also implies that the user will not be framed by the cloud server.

Theorem 1. *The provenance scheme is existentially unforgeable.*

Proof. There are two steps in the AnoAuth algorithm. The group signature $(Y, R_1, R_2, R_3, R_4, R_5, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, which is output by the GSign algorithm, is used to show that the user is a valid group member. If the group signature is unforgeable, any invalid user cannot pretend to be a group user. The second step is to use the ABS to generate a signature $\sigma = (\sigma_0, \{\sigma_i\}_{i\in A_k^*}, \sigma_0')$ for some predicate specified by the data owner. The proof cannot be forged if the ABS is unforgeable. Because we have already shown the security of the group signature and the ABS scheme, we can get that the provenance scheme is existentially unforgeable.

Theorem 2. *The provenance scheme achieves anonymity.*

Proof. The information of the user's identity could only be leaked from the group signature $(Y, R_1, R_2, R_3, R_4, R_5, s_\alpha, s_\beta, s_x, s_{\delta_1}, s_{\delta_2})$, or the ABS signature σ. However, from the anonymity of the group signature, we know that no user except the TPA knows the user's identity from the group signature. As a result, the provenance scheme achieves user anonymity.

Theorem 3. *The provenance scheme has traceability.*

Proof. The traceability is achieved because we use the algorithm Trace in the group signature [24]. More specifically, from a valid group signature, we can get (T_1, T_2, T_3). Then, with TPA's private key (ξ_1, ξ_2), the user's identity can be computed as $T_3/T_1^{\xi_1}T_2^{\xi_2}$.

6. Conclusion

In this paper, we have proposed a new provenance system with fine-grained access control based on an ABS scheme. In this new provenance system, the anonymity of the user is guaranteed by using the techniques of group signature and ABS. Furthermore, because the user's attribute private key is issued from multiple attribute authorities with an anonymous key-issuing protocol, the user's privacy is also protected from the attribute authorities. The computation and communication overhead for the data owner is low because user access is moved to the cloud server by using broadcast encryption. Furthermore, the outsourcing computation of exponentiation is also shown to reduce the user/data owner's computation overhead.

Acknowledgments

We are grateful to the anonymous referees for their invaluable suggestions. This work is supported by the National Natural Science Foundation of China (Nos. 61100224, 61272455).

References

[1] Amazon Web Services, AWS. Online at: http://aws.amazon.com.
[2] Google App Engine. Online at: http://code.google.com/appengine/.
[3] Microsoft Azure. http://www.microsoft.com/azure/.
[4] R. Buyya, C.S. Yeo, S. Venugopal, J. Broberg, I. Brandic, Cloud computing and emerging IT platforms: vision, hype, and reality for delivering computing as the 5th utility, Future Generation Computer Systems 25 (6) (2009) 599–616. Elsevier.
[5] Rongxing Lu, Xiaodong Lin, Xiaohui Liang, Xuemin Shen, Secure provenance: the essential of bread and butter of data forensics in cloud computing, in: ASIACCS 2010, ACM, 2010.
[6] R. Aldeco-Perez, L. Moreau, Provenance-based auditing of private data use, in: Proceedings of the 2008 international conference on Visions of Computer Science: BCS International Academic Conference, ACM, 2008, pp. 141–152.
[7] R.S. Barga, L.A. Digiampietri, Automatic capture and efficient storage of e-Science experiment provenance, Concurrency and Computation: Practice and Experience 20 (5) (2008) 419–429.
[8] R. Hasan, R. Sion, M. Winslett, Introducing secure provenance: problems and challenges, in: Proceedings of ACM workshop on Storage Security and Survivabilit, StorageSS'07, pp. 13–18.
[9] C.A. Lynch, When documents deceive: trust and provenance as new factors for information retrieval in a tangled Web, Journal of the American Society for Information Science and Technology 52 (1) (2001) 12–17.
[10] Susan Hohenberger, Anna Lysyanskaya, How to securely outsource cryptographic computations, in: TCC'05, in: LNCS, vol. 3378, 2005, pp. 264–282.
[11] Giuseppe Ateniese, Susan Hohenberger, Proxy re-signatures: new definitions, algorithms, and applications, in: CCS'05, 2005, pp. 310–319.
[12] D.M. Gorden, A survey of fast exponentiation methods, Journal of Algorithms 27 (1) (1998) 129–146.
[13] Y.L. Simmhan, B. Plale, D. Gannon, A survey of data provenance in e-Science, SIGMOD Record 34 (3) (2005) 31–36.
[14] I.T. Foster, J. Vockler, M. Wilde, Y. Zhao, Chimera. A virtual data system for representing, querying, and automating data derivation, in: SSDBM'02, IEEE Computer Society, 2002, pp. 37–46.
[15] R. Hasan, Z. Anwar, W. Yurcik, L. Brumbaugh, R. Campbell, A survey of peer-to-peer storage techniques for distributed file systems, in: ITCC'05, IEEE Computer Society, 2005, pp. 205–213.
[16] Vipul Goyal, Omkant Pandey, Amit Sahai, Brent Waters, Attribute-based encryption for fine-grained access control of encrypted data, in: CCS'06, ACM, 2006, pp. 89–98.
[17] B. Waters, Ciphertext-policy attribute-based encryption: an expressive, efficient, and provably secure realization, in: Proc. of PKC'11, in: LNCS, vol. 6571, Springer, Berlin, Heidelberg, 2011, pp. 53–70.
[18] Amit Sahai, Brent Waters, Fuzzy identity-based encryption, in: EUROCRYPT'05, in: LNCS, vol. 3494, Springer, 2005, pp. 457–473.
[19] H. Maji, M. Prabhakaran, M. Rosulek, Attribute based signatures: achieving attribute privacy and collusion-resistance, 2008. Available at: http://eprint.iacr.org/2008/328.
[20] Jin Li, Kwangjo Kim, Hidden attribute-based signatures without anonymity revocation, Information Sciences 180 (9) (2010) 1681–1689. Elsevier.
[21] Jin Li, Man Ho Au, Willy Susilo, Dongqing Xie, Kui Ren, Attribute-based signature and its applications, in: Proceeding of the 5th ACM Symposium on Information, Computer and Communications Security, ASIACCS'10, ACM, 2010, pp. 60–69.
[22] Jin Li, Xu Ma, Dongqing Xie, Improving privacy in multi-authority attribute-based signature, in: ChinaCrypt, 2011.
[23] D. Chaum, E. van Heyst, Group signatures, in: Eurocrypt'91, in: LNCS, vol. 547, Springer-Verlag, 1991, pp. 257–265.
[24] Dan Boneh, Xavier Boyen, Hovav Shacham, Short Group Signature. CRYPTO 2004.
[25] Amos Fiat, Moni Naor, Broadcast encryption, in: CRYPTO'93, in: LNCS, vol. 773, Springer-Verlag, 1993, pp. 480–491.
[26] Jin Li, Qiong Huang, Xiaofeng Chen, Sherman S. Chow, Multi-authority ciphertext-policy attribute-based encryption with accountability, in: ASIACCS'11, ACM, 2011.
[27] P. Feldman, A practical scheme for non-interactive verifiable secret sharing, in: Proc. 28th FOCS, 1987, pp. 427–437.
[28] M. Green, S. Hohenberger, B. Waters, Outsourcing the decryption of ABE ciphertexts, in: Proc. of SEC'11, USENIX Association, Berkeley, 2011.
[29] Xiaofeng Chen, Jin Li, Jianfeng Ma, Qiang Tang, Wenjing Lou, New algorithms for secure outsourcing of modular exponentiations, in: ESORICS, in: LNCS, vol. 7459, Springer, 2012, pp. 541–556.

Jin Li received his B.S. (2002) and M.S. (2004) from Southwest University and Sun Yat-sen University, respectively, both in Mathematics. He obtained his Ph.D. degree in information security from Sun Yat-sen University in 2007. Currently, he is working at Guangzhou University. His research interests include applied cryptography and security in cloud computing (secure outsourcing computation and cloud storage).

Xiaofeng Chen received his B.S. and M.S. degrees in Mathematics from Northwest University, China. He obtained his Ph.D. degree in Cryptography from Xidian University in 2003. Currently, he is working at Xidian University as a professor. His research interests include applied cryptography and cloud security.

Qiong Huang received his B.S. and M.S. degrees from Fudan University in 2003 and 2006, respectively, and he obtained his Ph.D. degree from City University of Hong Kong in 2010. He is currently working at the College of Informatics, South China Agricultural University. His research interests include cryptography and information security, in particular, cryptographic protocols design and analysis.

Duncan S. Wong received his B.Eng. degree in Electrical and Electronic Engineering with first class honors from the University of Hong Kong in 1994, his M.Phil. degree in Information Engineering from the Chinese University of Hong Kong in 1998, and his Ph.D. degree in Computer Science from Northeastern University, Boston, MA, USA, in 2002. After graduation, he was a visiting assistant professor at the Chinese University of Hong Kong for one year before joining City University of Hong Kong in September 2003. He is now an associate professor in the Department of Computer Science.