4-2020

Attribute-based cloud data integrity auditing for secure outsourced storage

Yong YU

Yannan LI

Bo YANG

Willy SUSILO

Guomin YANG
Singapore Management University, gmyang@smu.edu.sg

See next page for additional authors

Citation

YU, Yong; LI, Yannan; YANG, Bo; SUSILO, Willy; YANG, Guomin; and BAI, Jian. Attribute-based cloud data integrity auditing for secure outsourced storage. (2020). *IEEE Transactions on Emerging Topics in Computing*. 8, (2), 377-390.
Available at: https://ink.library.smu.edu.sg/sis_research/7289

Author

Yong YU, Yannan LI, Bo YANG, Willy SUSILO, Guomin YANG, and Jian BAI

Received 30 May 2017; revised 1 August 2017; accepted 27 August 2017.
Date of publication 4 October 2017; date of current version 9 June 2020.

Digital Object Identifier 10.1109/TETC.2017.2759329

Attribute-Based Cloud Data Integrity Auditing for Secure Outsourced Storage

YONG YU, (Member, IEEE), YANNAN LI, (Student Member, IEEE), BO YANG, (Member, IEEE), WILLY SUSILO, (Senior Member, IEEE), GUOMIN YANG, (Member, IEEE), AND JIAN BAI

Y. Yu and B. Yang are with School of Computer Science, Shaanxi Normal University, Xi'an 710062, China
Y. Li, W. Susilo, and G. Yang are with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia
Y. Yu and J. Bai are with Science and Technology on Communication Security Laboratory, Chengdu 610041, China
CORRESPONDING AUTHOR: Y. LI (liyannan2016@163.com) and B. YANG (byang@snnu.edu.cn)

ABSTRACT Outsourced storage such as cloud storage can significantly reduce the burden of data management of data owners. Despite of a long list of merits of cloud storage, it triggers many security risks at the same time. Data integrity, one of the most burning challenges in secure cloud storage, is a fundamental and pivotal element in outsourcing services. Outsourced data auditing protocols enable a verifier to efficiently check the integrity of the outsourced files without downloading the entire file from the cloud, which can dramatically reduce the communication overhead between the cloud server and the verifier. Existing protocols are mostly based on public key infrastructure or an exact identity, which lacks flexibility of key management. In this paper, we seek to address the complex key management challenge in cloud data integrity checking by introducing attribute-based cloud data auditing, where users can upload files to cloud through some customized attribute set and specify some designated auditor set to check the integrity of the outsourced data. We formalize the system model and the security model for this new primitive, and describe a concrete construction of attribute-based cloud data integrity auditing protocol. The new protocol offers desirable properties namely attribute privacy-preserving and collusion-resistance. We prove soundness of our protocol based on the computational Diffie-Hellman assumption and the discrete logarithm assumption. Finally, we develop a prototype of the protocol which demonstrates the practicality of the protocol.

INDEX TERMS Cloud storage, data integrity, attribute-based cryptography, threshold secret sharing

I. INTRODUCTION

Cloud storage, one of the most basic services of IaaS[1], is a configurable data storage model that enables data owners to store their files in the cloud without retaining a local copy, which greatly reduces data owners' storage and management burden of local files. Moreover, it is quite convenient for users to retrieve their files via terminals which have cloud access, such as mobile phones and tablet PCs. Cloud storage services have a number of significant advantages compared with traditional storage approaches, such as anytime and anywhere access, location-independent, on-demand services, flexible resources. Currently, an increasing number of individuals and enterprises are enjoying the convenience provided by cloud storage.

Cloud storage provides convenient, fast and unlimited capacity IT services to its users. However, due to the separation between data ownership and data management, cloud storage introduces some new data security challenges since data are hosted by cloud servers rather than data owners themselves. The cloud servers are not fully trusted. Any accidental data deletion by the cloud server, or worse, a physical catastrophe such as a fire or earthquake, might lead to permanent loss of users' data. This is not exaggerating the dangers to frighten people. Symantec, a well-known information security company, reported a survey and showed that 43 percent of respondents experienced cloud data loss accidents and had to recover the data from backups.[1] Thus, it is fair to claim that data integrity is the premise and basis of reliable cloud computing as well as big data analysis. If the integrity of cloud data is not ensured, the correctness of big data analysis and cloud computing cannot be guaranteed. As a

[1] https://finance.yahoo.com/news/cloud-computing-users-losing-data-205500612.html

consequence, data owners require a strong integrity guarantee of their outsourced data to make sure the cloud servers store their data correctly.

In order to address the issue mentioned above, the concept of cloud data integrity auditing was presented, which can be mainly divided into two categories, namely Proof of Retrieveability (PoR) and Provable Data Possession (PDP). PDP is a probabilistic detection protocol which employs randomly sampled data blocks rather than the entire file to perform cloud data integrity checking, which is more efficient than the deterministic auditing protocols [2], especially for large files. PoR protocols, similar to PDP, can not only detect the integrity of cloud data but also provide data retrieveability. By using error-correction coding techniques, PoR can improve the storage reliability. Both PDP protocols and PoR protocols are challenge-response protocols, where homomorphic verifiable authenticators are employed to reduce the communication and computation costs between cloud server and Third-Party Auditor (TPA) when conducting the cloud data auditing protocols.

Related Work. Deswarte *et al.* [2] put forward the concept of remote data integrity checking for the first time and presented a scheme based on RSA. Filho *et al.* [3] put forward a new protocol, which can greatly improve the date integrity auditing efficiency, that is, it costs 20 seconds for 1 MB file. Yamamoto *et al.* [4] proposed an efficient scheme by offering batch processing [5] based on the homomorphic hash function. The similar technique was employed in Sebe [6], in which they proposed a Diffie-Hellman protocol based on group Z_p but the length of each data block is limited and the storage overhead of the client is $O(n)$. Juels *et al.* [7] came up with the concept of PoR and described a concrete protocol by inserting some special blocks, named sentinels, into the original file. The cloud server is challenged by verifying some sentinels. Ateniese *et al.* [8], [9] proposed a PDP protocol based on homomorphic verifiable tag (HVT). HVT can aggregate responses of n challenged blocks into a single value, which can significantly reduce the communication cost of cloud server and TPA. Erway *et al.* [10] gave a framework supporting dynamic PDP by extending the protocol in [8], and proposed an efficient construction. Shacham and Waters [11] presented two PoR schemes using homomorphic message authentication code and BLS short signature [12]. The previous one supports private verification, while the latter one supports public verification. Recently, a variety of cloud data integrity auditing protocols with various eye-catching properties have been proposed such as supporting dynamic operations auditing [13], privacy-preserving auditing [14], [15], [16], public auditing [17], [18], and multiple copies auditing [19].

The aforementioned protocols are based on public key infrastructure (PKI), which consists of a set of roles, policies and procedures that needed to issue, manage, distribute, store and revoke digital certificates. The most commonly adopted digital certificate in our daily life is X.509 certificates, an ITU-T standard for a PKI and privilege management infrastructure. However, there are three weaknesses when involving PKI based protocols. First, the generation, management and revocation of digital certificates requires a highly complicated structure. Second, a PKI system is a tree structure and the authentication to the current CA relies on its parent CA. Thus, the root CA is a trusted center and self-signed, which is vulnerable since compromising root CA means all the related certificates should be re-issued. Third, the certificates issued by a CA may not secure enough to ensure the security of one's secret key. For example, Dell's self root certificate was reported to expose users' encrypted data to spy in 2015.[2]

In order to reduce the complexity of certificate management in PKI, identity based (ID-based) cryptology[20] was proposed by Shamir, in which the secret key binds with the user's identity. Therefore, users can communicate without exchanging digital certifications. Due to the flexibility in key management, ID-based cryptology has been widely adopted in a variety of primitives, including in cloud data integrity auditing protocols. A number of ID-based cloud data auditing protocols have been proposed such as [22]—[24]. The most commonly used identity information in existing ID-based cloud data auditing protocols is an arbitrary bit string chosen by a user, such as names, IP and E-mail, which can be viewed as a text-based recognition related to the combinations of characters and numbers. With this identity information, one can register for a private key binding to his/her identity from the private key generation center. There are three weaknesses when making use of ID-based protocols. First, identity might not be unique if identity information is not chosen properly. For example, the name "Nancy Helen" is probably not unique. Second, a user needs to "prove" to the private key generator centre that the claimed identities are indeed belong to him, which is typically verified by providing some additional documents such as one's passport or identity card. However, these supplementary documents, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TETC.2017.2759329, themselves are subject to forgery. Third, one has to keep in mind his/her identity information even sometimes an identity is too long to remember.

We seek to address the issue mentioned above by proposing an alternative named attribute-based cloud data integrity auditing. Different from the previous work that attribute-based cryptography is used to realize data sharing [25], [26] or access control [27] in a cloud environment. The notion of an attribute-based cloud data auditing protocol is a generalization of fuzzy identity-based cloud data auditing protocol [28]. In this primitive, it allows cloud users to define some attribute sets such as name, age and select a subset of those attributes to generate private keys to generate the metadata of the files which need outsourcing rather than some inherent attribute [28]. When it comes to auditing phase, the cloud users can designate a certain group of people with a set of similar attributes to execute the cloud data integrity checking.

[2] http://www.computerworld.com/article/3007981/security/what-you-need-to-know-about-dells-root-certificate-security-debacle.html

Compared with traditional cloud data integrity checking, the advantages of attribute-based data integrity auditing protocols are as follows. First, an attribute-based cloud data auditing protocol enables the data owners to specify the scope of the auditors, which avoids the situation of single-point failure in traditional protocols which has a single TPA. Second, an attribute-based cloud data auditing scheme allows users to select their attribute sets when uploading files. Generally speaking, one with n atomic attributes can enjoy 2^n combined attributes to manipulate the file. This can be implemented by an attribute-based data auditing scheme with the key size $O(n)$, rather than $O(2^n)$ if employing traditional data auditing schemes. Thus, attribute-based cloud data integrity protocols are more flexible and practical compared with the traditional proposals in many real-world scenarios.

Contributions. In this paper, we attempt to simplify the key management issue of traditional cloud data integrity auditing protocols by incorporating attribute-based cryptography. Our contributions are three-fold.

1) We propose the notion of attribute-based cloud data integrity auditing, where users can choose some arbitrary attributes to generate private keys and upload files to cloud server. Moreover, the data owners can specify the set of auditors who are able to check the integrity of the outsourced data.
2) We formalize the system model as well as the security model of this new primitive to ensure the security named *soundness* of cloud data integrity auditing.
3) We describe a concrete construction of attribute-based cloud data integrity auditing protocol. We then prove the security of the protocol under Shacham-Waters game-based proof framework [11].

Paper Organization. The rest of the paper is organized as follows. Some preliminaries are reviewed in the next section. We describe the system model and security model of attribute-based cloud data integrity auditing protocols in Section III. In Section IV, we describe the proposed construction as well as analyze its correctness and attribute privacy preserving. The security proof of the proposed protocol is shown in Section V. We report the implementation performance in Section VI. Finally, we conclude this paper in Section VII.

II. PRELIMINARIES

In this section, we review some preliminaries which will be used in the following sections, including bilinear maps, complexity assumptions, threshold secret sharing scheme and attribute-based signatures.

A. BILINEAR MAPS

Assume G_1 and G_2 are two groups of the same prime order p. g is the generator of group G_1. A bilinear pairing $e : G_1 \times G_1 \rightarrow G_2$ is a map which satisfies the following properties [30]:

Computational. $e(u, v)$ can be efficiently computed for all $u, v \in G_1$.

Bilinear. $e(u^a, v^b) = e(u, v)^{ab}$, for all $u, v \in G_1$ and all $a, b \in Z_p$.

Non-Degenerate. For the generator g in G_1, $e(g, g) \neq 1$.

B. COMPLEXITY ASSUMPTION

1) COMPUTATIONAL DIFFIE-HELLMAN (CDH) ASSUMPTION [12]

G denotes a cycle group of order p, where p is a large prime number. Given the tuple of (g, g^a, g^b), it has an ϵ advantage to solve a CDH problem if

$$Pr[\mathcal{A}(g, g^a, g^b) = g^{ab}] \geq \epsilon.$$

Definition 1: The (t, ϵ)-CDH assumption holds in G if no t-time algorithm has advantage at least ϵ in solving the CDH problem.

2) DISCRETE LOGARITHM (DL) ASSUMPTION [31]

Given the tuple (g, g^a) where $a \in Z_p$. An algorithm $\mathcal{A}$ has the advantage ϵ in solving the discrete logarithm problem if

$$Pr[\mathcal{A}(g, g^a) = a] \geq \epsilon.$$

Definition 2: The (t, ϵ)-DL assumption holds in G if no t-time algorithm has advantage at least ϵ in solving the DL problem.

C. THRESHOLD SECRET SHARING

Secret sharing is a cryptographic technique that can split and recover secret. In a threshold secret sharing scheme, a secret is divided into several segments and assigned to a group of participants. A sufficient number of participants in a qualified subset can jointly recover the secret. Specifically, in a (k, n) secret sharing scheme which includes n players $P_1, \dots, P_n$ and a dealer, the dealer divides the secret s into n pieces, and each player P_i is distributed a unique secret share $s_i (1 \leq i \leq n)$, such that

1) k or more players can reconstruct s with their shares,
2) less than k players learns nothing about the secret.

k is called the threshold value. Shamir's threshold secret sharing scheme [32] is a well-known, where a unique $k - 1$ degree polynomial $f(x)$ is employed to split the secret s into k shares

$$f(x) = a_0 + a_1 x + \cdots + a_{k-1} x^{k-1},$$

where $a_1, \dots, a_{k_1}$ are random numbers and $a_0 = s$. In the distribution phase, the dealer randomly chooses some $x_i \in Z_p$, and computes

$$s_i = f(x_i),$$

as a share of s and allocates the shares to a group of players. When reconstructing the secret, a set S of k players are selected to recover $f(x)$ as follows:

$$f(x) = \sum_{P_i \in S} \Delta_{x_i, s}(x) s_i,$$

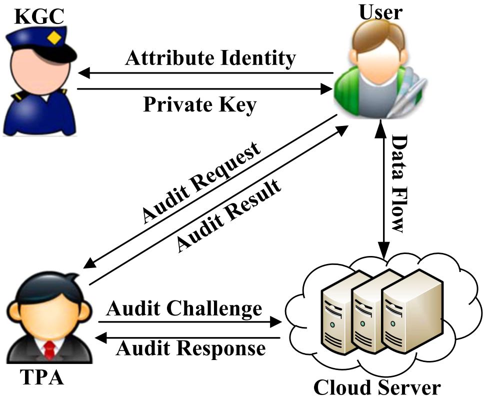


FIGURE 1. The system model of attribute-based data integrity auditing protocol.

where

$$\Delta_{x_i,s}(x) = \prod_{P_i \in S, j \neq i} \frac{x - x_j}{x_i - x_j},$$

denotes the Lagrange coefficient. We can obtain the secret s by calculating $f(0)$.

D. ATRRIBUTE-BASED SIGNATURE

An attribute-based signature (ABS) [33] involves two entities, key generation center (KGC) and a user. KGC is responsible for generating the corresponding secret key for a user with the claimed attribute set. Upon receiving secret key from KGC, a user can generate an attribute-based signature. This primitive consists of the following four algorithms.

- Setup(k): This is a probabilistic algorithm, which takes a security parameter k as input and outputs the master key MK as well as the public parameter PK.
- Extract(MK, A): This is a probabilistic algorithm which takes a master key MK and an attribute set A as input. It generates secret key SK_A for the user.
- Sign(PK, SK_A, Υ, M): This is a probabilistic algorithm which takes the public parameter PK, a secret key SK_A, a predicate Υ and a message M as input. It outputs a signature σ.
- Verify($PK, B, \Upsilon, M, \sigma$): This is a deterministic algorithm which takes the public parameter PK, an attribute set B, a predicate Υ, the message M and its alleged signature σ as input. It returns 1 or 0 to indicate the signature is valid or not.

III. SYSTEM MODEL AND SECURITY MODEL

We now present the system model and security model for attribute-based cloud data integrity auditing protocols.

A. SYSTEM MODEL

As described in Figure 1, an attribute-based cloud data integrity auditing protocol involves four entities, namely key generation centre, cloud users, cloud servers and TPA. KGC takes charge of generating users' private key according to their attribute set, and TPA is a third party designated to verify the cloud data's integrity on behalf of cloud users upon audit request. The details of an attribute-based cloud data integrity auditing protocol are as follows.

1) A cloud user forwards his/her attribute set to KGC to request his/her private key.
2) KGC generates a private key for the user with the master secret key and the user's attributes.
3) The cloud user preprocess the file by generating metadata of the file with his/her private key. The user then uploads the file together with the corresponding metadata to the cloud, and deletes the local copy of the file.
4) Upon receiving the auditing request, TPA and the cloud server execute a challenge-response protocol to verify if the stored file is intact.

B. SYSTEM COMPONENTS

More formally, an attribute-based cloud data integrity auditing protocol consists of the following six algorithms.

- Setup(1^l). This is a probabilistic algorithm run by the KGC. It takes a secure parameter l and outputs the master public key mpk as well as the master secret key msk.
- Extract(mpk, msk, A). This is a probabilistic algorithm run by the KGC. It takes the master public key mpk, the master secret key msk and a user's attribute set A as input and outputs a private key ssk_A for the user.
- MetadataGen(mpk, ssk_A, F). This is a probabilistic algorithm run by the data owner. It takes the master public key mpk, the private key ssk_A and a file F as input and outputs the file tag τ, and a set of block authenticators $\{\sigma_i\}_{1 \leq i \leq n}$ for the file blocks $\{m_i\}_{1 \leq i \leq n}$.
- Challenge(mpk, τ, B). This is a probabilistic algorithm run by the TPA. It takes the master public key mpk, the file tag τ and the auditor's attribute set B (the user can act as an auditor as well) as input. It outputs a challenge C.
- Response($mpk, F, \tau, \{\sigma_i\}_{1 \leq i \leq n}, C$). This is a probabilistic algorithm run by the cloud server. It takes the master public key mpk, the file F, the file tag τ, the block authenticators $\{\sigma_i\}_{1 \leq i \leq n}$ and the challenge C as input. It outputs a response $resp$ to prove the possession of the user's file.
- Verify($mpk, B, C, resp$). This is a deterministic algorithm run by the TPA. It takes the master public key mpk, the auditor's attribute set B, the challenge C and the response $resp$ as input and outputs an auditing result $result \in \{0, 1\}$ to show whether the stored file F is virgin.

C. SECURITY REQUIREMENTS

An attribute-based cloud data integrity auditing protocol should satisfy the following properties [11]

1) *Correctness.* Correctness states that for a valid proof, which is generated by the *Response* algorithm, the *Verify* algorithm can accept it with an overwhelming probability.
2) *Soundness.* Soundness requires that, any cheating prover, who can generate a valid proof that can pass the *Verify*

algorithm is actually storing the challenged file. In other words, there is no adversary, who does not store the file, can generate a valid proof of the challenge.

3) *Collusion resistance.* Collusion resistance indicates that a group of users can complete cloud data auditing if at least one individual has the permission to do so. In other words, if a group of users cannot generate a valid response individually, the advantage to output a valid response will not increase even all the users collude. Note that in the security model of *Soundness*, the adversary can make Extract queries to inquire the private key of selected attributes, where the overlap of the selected attributes and the set of challenge attributes must be less than d. This is resemble the collusion resistance scenario. Therefore, in the security model of *Soundness*, the adversary has the ability to perform collusion attack. Thus, the property of collusion resistance holds naturally if the property of *Soundness* holds.
4) *Attribute privacy-preserving.* Attribute privacy-preserving property denotes that, during cloud data auditing phase, TPA can not deduce the set of attributes used by users to upload the file except the d common attributes selected by cloud server. Therefore, we require that if TPA can guess the user's attribute from the response, it can also complete the deduction when only given the intersection with d attributes. This property ensures that only the intersection attributes selected by the cloud server are possibly revealed to TPA when executing the challenge-response protocol.

D. SOUNDNESS MODEL

We provide the following security model to make the notion of soundness more specifically. In essence, the security model says that there exists an extractor $Extr(mpk, attr, \tau, P')$, which takes the master public key mpk, the attribute set A, the file tag τ and the cheating prover P' as input, and can extract the original file F. The game of soundness between an adversary and a challenger is shown as follows.

Initial. The adversary declares a target attribute set, α, to be challenged upon.

Setup. The challenger runs *Setup* algorithm and gets the master public key *mpk* and the master secret key *msk*. The challenger forwards the master public key *mpk* to the adversary.

Queries. The adversary is supposed to make some queries including the Extract queries and the MetadataGen queries.

- Extract queries. The adversary can make queries on some attribute sets γ for the corresponding private keys, where $|\gamma \cap \alpha|$ should be less than d for all the queried attribute set.
- MetadataGen queries. The adversary can make queries on some file F for the file tag, the challenger runs the *Extract* algorithm to get the private keys and runs the *MetadataGen* algorithm to obtain the metadata of the file, and returns the metadata to the adversary.

ProofGen. For a specified attribute set *attr*, the file tag τ and the file F on which a MetadataGen query has been made, the adversary can make an interaction with the challenger by executing the challenge-response protocol, where the verifier acts as the challenger. The adversary is given an output of *Verify* algorithm when the protocol execution completes.

Output. Finally, the adversary outputs a challenge tag τ together with the target attribute set α which is chosen at the *Initial* stage, and the description of a prover P'.

The cheating prover P' can interact with the verifier by following the protocols with the input of the challenge tag τ returned from file F, and the challenge identity α. We say the cheating prover P' is ϵ-admissible, if it can convincingly answer an ϵ fraction of the challenges.

Definition 3: An attribute-based data integrity auditing protocol is ϵ-sound if there exists an adversary conducting the Setup *algorithm and outputs an ϵ-admissible cheating prover P' for a file F, then there exists an extraction algorithm that can recover the file F from P' with a nonnegligible probability.*

IV. A CONCRETE CONSTRUCTION

In this section, we present a concrete construction of attribute-based cloud data integrity auditing protocol inspired by attribute-based cryptographic construction [29]. We first describe the basic idea of our protocol, followed by the detailed construction.

A. BASIC IDEA

The proposed attribute-based cloud data integrity auditing protocol consists of three procedures, namely *Enroll, Store* and *Audit*. *Enroll* phase involves the cloud user and a KGC following *Setup* and *Extract* algorithm. The user chooses some attribute set and submits it to KGC. KGC checks the validity and generates the corresponding private key for the cloud user with the master secret key with *Extract* algorithm. *Store* phase involves the cloud user and the cloud server with *MetadataGen* algorithm. The user preprocesses the File F to be uploaded into F^*. Then generates the file tag and block authenticators using the private key using *MetadataGen* algorithm. After that, the cloud user uploads the metadata to the cloud server and deletes the local copy. The *Audit* phase involves an auditor(or the cloud user), cloud server and a TPA. The auditor sends his own attribute set to the TPA as an audit request and TPA runs the *Challenge-Response* protocol with cloud server to check the integrity of the file stored on the sever. TPA first generates a challenge and forwards audit request as well as the challenge set to cloud server. Upon receiving the challenge from TPA, the cloud server checks the overlap attribute set between the cloud user's and the auditor's. If the number of intersection is less than the auditing precision d, which is set by the cloud user in *Setup* phase, cloud server emits failure and returns $\perp$. Otherwise, cloud server generates a response with the challenged file F^* together with the corresponding block authenticators. To achieve user privacy-preserving, the cloud server first chooses an intersection of A and B with d elements and converts the response accordingly to prevent TPA learning the signer's attributes outside $A \cap B$, and forwards the converted response to TPA. Finally, TPA verifies the response and returns the auditing result to the user.

B. OUR CONSTRUCTION

The details of the proposed construction are as follows. Assume a user's attribute set contains at most m elements in Z_p. Without loss of generality, each element in the attribute set can be arbitrary string, and we can map it to group Z_p with a collision-resistance hash function. We denote a possible attribute set as $M = \{1, 2, \ldots, m+1\}$ for convenience. We choose a proper d as an audit precision to describe the set overlap. Let G_1 and G_2 be two multiplicative cyclic groups of the same prime order p and g be a generator of group G_1. $e : G_1 \times G_1 \rightarrow G_2$ denotes a bilinear map. (e, G_1, G_2) are public parameters. We then denote Lagrange coefficient $\Delta_{i,M}$ for $i \in Z_p$ and a set, M, of element in Z_p

$$\Delta_{i,M} = \prod_{j \in M, j \neq i} \frac{x-j}{i-j}. \tag{1}$$

Setup(1^k). Choose $g_1 = g^y, g_2, h \in G_1$ and set $A = e(g_1, g_2)$. Then, uniformly pick $t_1, \ldots, t_{m+1} \in G_1$. A function T is defined as

$$T(x) = {g_2}^{x^m} \prod_{i=1}^{m+1} {t_i}^{\Delta_{i,M}(x)}.$$

The public key is $mpk = \{g, g_1, g_2, h, t_1, \ldots, t_{m+1}\}$, and the master key is $msk = y$.

Extract(mpk, msk, A). To generate a private key for a user with an attribute set A, where $|A| = m$, KGC first chooses a random $d-1$ degree polynomial q such that $q(0) = y$. Then, for each attribute in A, KGC randomly chooses a $r_k \in Z_p$ and calculates the private key $ssk_A = (\{D_k\}_{k \in A}, \{d_k\}_{k \in A})$ as follows:

$$\begin{aligned} D_k &= g_2^{q(k)} T(k)^{r_k}, \\ d_k &= g^{r_k}. \end{aligned}$$

MetadataGen(mpk, ssk_A, F). The user (data owner) chooses a local file F, and encodes it with erasure code to get file F'; then splits F' into n blocks, each of which contains s sectors: $F^* = \{m_{ij}\}_{1 \leq i \leq n, 1 \leq j \leq s}$. Choose a $name$ from group Z_p for the file. Randomly choose s elements $u_1, \ldots, u_s \in G_1$. Let $\tau_0 = name\|n\|u_1\| \cdots \|u_s$. Then the file tag τ is τ_0 together with an attribute-based signature [33] on τ_0: $\tau = \tau_0\|Sign(\tau_0)$. Choose a random $s_k \in Z_p$ for each $k \in A$, and generate a block authenticator for the ith$((1 \leq i \leq n))$ block as follows:

$$\begin{aligned} \sigma_{1i}^{(k)} &= \left\{ D_k \cdot \left(H(name\|i) \cdot h \cdot \prod_{j=1}^{s} {u_j}^{m_{ij}} \right)^{s_k} \right\}_{k \in A} \\ \sigma_{2i}^{(k)} &= \left\{ g^{s_k} \right\}_{k \in A} \\ \sigma_{3i}^{(k)} &= \left\{ g^{r_k} \right\}_{k \in A}. \end{aligned}$$

The user uploads the file F', the corresponding *metadata* and the attribute set A to the cloud server, where the *metadata* includes the file tag together with the block authenticators $(\tau, \{\sigma_{1i}^{(k)}, \sigma_{2i}^{(k)}, \sigma_{3i}^{(k)}\}_{1 \leq i \leq n})$.

Challenge(spk, τ, B). Upon receiving the auditing request from an auditor with attribute set B, TPA picks an l-element subset I from set $[1, n]$, and chooses a random $v_i \in Z_p$ for each $i \in I$. Let C be the set $\{(i, v_i)\}_{i \in I}$, then TPA forwards the challenge C and attribute set B to the cloud server.

Response$(F, \tau, C, B, \{\sigma_{1i}^{(k)}, \sigma_{2i}^{(k)}, \sigma_{3i}^{(k)}\}_{i \in C})$. Upon receiving the challenge $C = \{(i, v_i)\}_{i \in I}$ and attribute set B from the TPA, the cloud server first checks whether $|A \cap B| \geq d$ holds. If not, the cloud server rejects the audit request and returns $\perp$. Otherwise, the cloud server chooses a set $S = |A \cap B|$ where $|S| = d$. Then, convert the corresponding block authenticators in the following way.

1) For $k \in S$, compute

$$\begin{aligned} \tilde{\sigma}_{1i}^{(k)} &= \left(\sigma_{1i}^{(k)}\right)^{1/\Delta_{k,B \setminus S}(0)} \\ \tilde{\sigma}_{2i}^{(k)} &= \left(\sigma_{2i}^{(k)}\right)^{1/\Delta_{k,B \setminus S}(0)} \\ \tilde{\sigma}_{3i}^{(k)} &= \left(\sigma_{3i}^{(k)}\right)^{1/\Delta_{k,B \setminus S}(0)}. \end{aligned}$$

2) For $k \in B \setminus S$, compute

$$\begin{aligned} \tilde{\sigma}_{1i}^{(k)} &= \left(T(k) \cdot H(name\|i) \cdot h \cdot \prod_{j=1}^{s} {u_j}^{m_{ij}} \right)^{1/\Delta_{k,B \setminus S}(0)} \\ \tilde{\sigma}_{2i}^{(k)} &= g^{1/\Delta_{k,B \setminus S}(0)} \\ \tilde{\sigma}_{3i}^{(k)} &= g^{1/\Delta_{k,B \setminus S}(0)}. \end{aligned}$$

Then, the cloud server generates response as follows:

$$\begin{aligned} \mu_j &= \sum_{i \in C} v_i m_{ij}, \\ \sigma_1^{(k)} &= \left\{ \prod_{i \in C} \tilde{\sigma}_{1i}^{(k) v_i} \right\}_{k \in B}, \\ \sigma_2^{(k)} &= \left\{ \tilde{\sigma}_{2i}^{(k)} \right\}_{i \in C, k \in B}, \\ \sigma_3^{(k)} &= \left\{ \tilde{\sigma}_{3i}^{(k)} \right\}_{i \in C, k \in B}, \end{aligned}$$

and returns $resp = (\mu_1, \ldots, \mu_s, \sigma_1^{(k)}, \sigma_2^{(k)}, \sigma_3^{(k)})$ to the TPA.

Verify$(resp, C, B)$. Upon receiving the proof from the server, the TPA verifies whether

$$\prod_{i \in C} A^{v_i} \overset{?}{=}$$
$$\left(\frac{e(\sigma_1^{(k)}, g)}{\prod_{i \in C} e\left(T(k), {\sigma_3^{(k)}}^{v_i}\right) e\left(\left(H(name\|i)h\right)^{v_i} \prod_{j=1}^{s} {u_j}^{\mu_j}, \sigma_2^{(k)}\right)} \right)^{\Delta_{k,B}(0)},$$

holds. If the equation holds, return 1; Otherwise return 0.

Correctness. The correctness of the proposed construction can be derived directly from the property of the bilinear mapping and the definition of Lagrangian coefficient (Equation (1)). We can intuitively get the following equation by the definition of the Lagrangian coefficient. For an arbitrary attribute set B and any of its subset S, we have

$$\Delta_{i,B}(x) = \Delta_{i,S}(x) \cdot \Delta_{i,B\backslash S}(x).$$

With the above property in mind, we can split the verification equation in *Verify* algorithm into two parts.

For $k \in S$, we have

$$\begin{aligned}
&\prod_{k\in B}\left(\frac{e(\sigma_1^{(k)}, g)}{\prod_{i\in C} e\left(T(k), \sigma_3^{(k)v_i}\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, \sigma_2^{(k)}\right)}\right)^{\Delta_{k,B}(0)} \\
&= \prod_{k\in B}\left(\prod_{i\in C}\frac{e(\tilde{\sigma}_1^{(k)}, g)^{v_i}}{e\left(T(k), \tilde{\sigma}_3^{(k)v_i}\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, \tilde{\sigma}_2^{(k)}\right)}\right)^{\Delta_{k,B}(0)} \\
&= \prod_{k\in S}\left(\prod_{i\in C}\frac{e(\tilde{\sigma}_1^{(k)}, g)^{v_i}}{e\left(T(k), \tilde{\sigma}_3^{(k)v_i}\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, \tilde{\sigma}_2^{(k)}\right)}\right)^{\frac{\Delta_{k,B}(0)}{\Delta_{k,B\backslash S}(0)}} \\
&= \prod_{k\in S}\prod_{i\in C}\left(\frac{e\left(H(name\|i)h\prod_{j=1}^{s} u_j^{m_{ij}}, g^{s_k}\right) e\left(T(k), g^{r_k}\right)^{v_i} e(g_2{}^{q(k)}, g)^{v_i}}{e\left(T(k), g^{r_k}\right)^{v_i} e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, g^{s_k}\right)}\right)^{\Delta_{k,S}(0)} \\
&= \prod_{k\in S}\prod_{i\in C}\left(e\left(g_2{}^{q(k)}, g\right)^{\Delta_{k,S}(0)}\right)^{v_i} = \prod_{i\in C}\left(e\left(g_2, g\right)^{\sum_{k\in S} q(k)\Delta_{k,S}(0)}\right)^{v_i} \\
&= \prod_{i\in C} A^{v_i}.
\end{aligned}$$

Similarly, for $k \in B\backslash S$, we have

$$\begin{aligned}
&\prod_{k\in B}\left(\frac{e(\sigma_1^{(k)}, g)}{\prod_{i\in C} e\left(T(k), \sigma_3^{(k)v_i}\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, \sigma_2^{(k)}\right)}\right)^{\Delta_{k,B}(0)} \\
&= \prod_{k\in B\backslash S}\left(\prod_{i\in C}\frac{e(\tilde{\sigma}_1^{(k)}, g)^{v_i}}{e\left(T(k), \tilde{\sigma}_3^{(k)v_i}\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, \tilde{\sigma}_2^{(k)}\right)}\right)^{\Delta_{k,B}(0)} \\
&= \prod_{k\in B\backslash S}\prod_{i\in C}\left(\frac{e\left(T(k)H(name\|i)h\prod_{j=1}^{s} u_j^{m_{ij}}, g\right)^{v_i}}{e\left(T(k), g\right) e\left((H(name\|i)h)^{v_i}\prod_{j=1}^{s} u_j^{\mu_j}, g\right)}\right)^{\frac{\Delta_{k,S}(0)}{\Delta_{k,B\backslash S}(0)}} \\
&= \prod_{k\in B\backslash S}\prod_{i\in C} 1^{\Delta_{k,S}(0)} = 1.
\end{aligned}$$

According to the above two equations, we can see that, for each $k \in B$, the equation in *Verify* algorithm holds.

C. ATTRIBUTE PRIVACY-PRESERVING

The proposed construction has the property of attribute privacy-preserving. It is obvious that, for a response generated by cloud server, only the d attributes in the intersection set S are valid and they do not contain any information related to other attributes in the user's attribute set A. So it reveals no information but the d common attributes selected by cloud server when the TPA checks the validity the response. Thus, the property of attribute privacy-preserving achieves.

D. COLLUSION RESISTANCE

The proposed construction has the property of collusion resistance. Intuitively, since the private keys of various users are tied to different random polynomials selected by users in Extract algorithm, an adversary fails to combine them to launch a collusion attack. A formal security proof is provided in Section V. Thus, the proposed construction can naturally achieve collusion resistance if it satisfies the property of Soundness, which will be proved in the next section.

V. SOUNDNESS PROOF

In this section, we demonstrate that the attribute-based cloud data integrity auditing protocol achieves the property of soundness.

Theorem: If the attribute-based signature scheme employed to generate file tags is existentially unforgeable and the CDH assumption holds in bilinear groups, in the random oracle model, except with negligible probability no adversary against the soundness property of our attribute-based cloud data integrity auditing protocol can make the verifier to accept a response of a challenge instance, except by generating values $\{\mu_j\}$ and $\{\sigma_1^{(k)}, \sigma_2^{(k)}, \sigma_3^{(k)}\}$ correctly, that is, they must be obtained as they are computed in the Response *algorithm in our protocol.*

We prove the above theorem through a series of games

Game 0. Game 0, as the first game, is simply the challenge game defined in security model (Section III).

Game 1. Game 1 is the same as Game 0 except one difference. The challenger keeps a list of file tags issued as the metadata of the outsourced file. If the adversary is able to generate such a file tag t that (1) is valid under the *Sign* algorithm (2) but is not a tag generated by the challenger, the challenger aborts.

Obviously, if the adversary causes the challenger to abort in Game 1 with a non-negligible probability, then we can use the adversary to forge an attribute-based signature. Otherwise, if the adversary does not let the challenger to abort, then his perspective will be the same as Game 0. Although we make this modification in Game 1, the verify and extract algorithms will never use random numbers $u_1, \ldots, u_s$ in file tags generated by entities other than the challenger. Thus, the file tag with a valid signature is either generated by the valid attribute-based signature algorithm, or produced by the adversary in his own way. However in Game 1, if the adversary is able to produce a file tag that is calculated by an effective attribute-based signature algorithm which is different from the challenger, the challenger will abort. In other words, the verify and extract algorithms will never deal with the latter situation. Therefore, we can conclude that all of the $u_1, \ldots, u_s$ are generated by the challenger.

Game 2. Game 2 is the same as Game 1, with one difference. The challenger keeps a list of responses to extract queries from the adversary. Now the challenger observes each instance of the protocol, including key extraction, tag generation, challenge-response and verification. If the adversary succeeds in any of these instances, that is, the *Verify* algorithm outputs 1, but the private key for the attribute set A in this instance is not generated by the extract algorithm, the challenger declares failure and aborts.

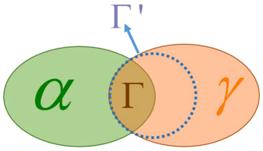


FIGURE 2. Query sets.

It is clear that the difference in the adversary's success probability between Games 2 and 1 is the probability that the adversary can forge a valid private key for an attribute set ω. With this in mind, we now illustrate that if there is a nonnegligible difference in the adversary's success probability between Games 2 and 1, we can build a simulator that can solve the Computational Diffie-Hellman problem.

The simulator is given g, g^a, g^b as input, and is supposed to output g^{ab}. The simulation runs as follows:

The adversary selects a random attribute set α to be challenged upon.

The simulator sets the public parameters as $g_1 = g^a$, $g_2 = g^b$. It then chooses a random polynomial of m degree $f(x)$ and another m degree polynomial such that

$$u(x)\begin{cases} = -x^m, & x \in \alpha \\ \neq -x^m, & others, \end{cases}$$

the simulator sets $t_i = g_2^{u(i)} g^{f(i)}$, for i from 1 to m. Then we can compute T_i in the same way as in our construction: $T(i) = g_2^{i^m} \prod_{i=1}^{m} t_i^{\Delta_{j,M(i)}} = g_2^{i^m} \prod_{i=1}^{m} g_2^{u(i)} g^{f(i)^{\Delta_{j,M(i)}}} = g_2^{i^m+u(i)} g^{f(i)}$. So we are equivalently to set $T(i)$ as follows:

$$T(i) = \begin{cases} g^{f(i)}, & x \in \alpha \\ g_2^{i^n+u(i)} g^{f(i)}, & others. \end{cases}$$

The public parameters $PP = (g, g_1, g_2, t_1, \ldots, t_{m+1},\ A = e(g_1, g_2))$, and the master secret key is a, which is outside the view of the simulator.

To answer the private key queries on identity γ, where $|\gamma \cap \alpha|$ is less than d, the simulator acts as follows. We first define three sets, as shown in Figure 2, Γ, Γ', S, where $\Gamma = \gamma \cap \alpha$, Γ' is the set satisfying $\Gamma \subseteq \Gamma' \subseteq \gamma$ and $|\Gamma'| = d - 1$, $S = \Gamma' \cup 0$ Then we set the private keys $\{D_i, d_i\}_{i \in \gamma}$ as follows:

1) For $i \in \Gamma'$, the private keys are set as

$$D_i = {g_2}^{\lambda_i} T(i)^{r_i}, d_i = g^{r_i},$$

where λ_i, r_i are randomly chosen in Z_p.

2) For $i \in \gamma - \Gamma'$, the private keys are computed as

$$D_i = \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) \cdot \left({g_1}^{\frac{-f(i)}{i^m+u(i)}} \left({g_2}^{i^m+u(i) g^{f(i)}} \right)^{{r_i}'} \right)^{\Delta_{0,S}(i)}$$

$$d_i = \left({g_1}^{\frac{1}{i^m+u(i)}} g^{{r_i}'} \right)^{\Delta_{0,S}(i)} .$$

For the simulation of $i \in \Gamma'$, it is obvious to see the correctness. When it comes to $i \in \gamma - \Gamma'$, which indicates $i \notin \alpha$, $u(i) \neq -i^m$, so $i^m + u(x)$ will be non-zero. We claim the assignment is identical to the original scheme from the adversaries view. To observe this, we set $r_i = (r_i' - \frac{a}{i^m+u(i)}) \Delta_{0.S}(i)$, then we have

$$\begin{aligned} D_i &= \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) \left({g_1}^{\frac{-f(m)}{i^m+u(i)}} \left({g_2}^{i^m+u(i)} g^{f(i)} \right)^{{r_i}'} \right)^{\Delta_{0,S}(i)} \\ &= \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) \left(g^{\frac{-af(m)}{i^m+u(i)}} \left({g_2}^{i^m+u(i)} g^{f(i)} \right)^{{r_i}'} \right)^{\Delta_{0,S}(i)} \\ &= \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) \cdot \\ &\quad \left({g_2}^{a} \left({g_2}^{i^m+u(i)} g^{f(i)} \right)^{\frac{-af(m)}{i^m+u(i)}} \left({g_2}^{i^m+u(i)} g^{f(i)} \right)^{r_i'} \right)^{\Delta_{0.S}(i)} \\ &= \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) \left({g_2}^{a} \left({g_2}^{i^m+u(i)} g^{f(i)} \right)^{{r_i}' - \frac{-af(n)}{i^m+u(i)}} \right)^{\Delta_{0.S}(i)} \\ &= \left(\prod_{j \in \Gamma'} {g_2}^{\lambda_j \Delta_{j,S}(i)} \right) {g_2}^{a \Delta_{0,S}(i)} (T(i))^{r_i} \\ &= {g_2}^{q(i)} T(i)^{r_i} \\ d_i &= \left({g_1}^{\frac{1}{i^m+u(i)}} g^{-{r_i}'} \right)^{\Delta_{0,S}(i)} = \left(g^{-(r_i' - \frac{a}{i^n+u(i)})} \right)^{\Delta_{0,S}(i)} \\ &= g^{-r_i}. \end{aligned}$$

Eventually, the adversary outputs a valid forgery of the private keys $k_\alpha = (\{{D_i}^*\}, \{{d_i}^*\})_{i \in \alpha}$ for the identity α. The simulator then can solve the CDH problem using the forgery from the adversary. First, the simulator selects a random set $\alpha^* \subseteq \alpha$, where $|\alpha^*| = d$, and computes as follows:

$$D^* = \prod_{i \in \alpha^*} \{{D_i}^*\}^{\Delta_{i,\alpha^*}(i)}$$

$$d^* = \prod_{i \in \alpha^*} \{{d_i}^*\}^{\Delta_{i,\alpha^*}(i) f(i)}.$$

Finally. the simulator outputs the solution to the instance of the CDH problem as

$$g^{ab} = D^* d^*.$$

Game 3. Game 3 is the same as Game 2, with one difference. The challenger keeps a list of responses to metadata queries from the adversary. Now the challenger observes each instance of the protocol, including key extraction, metadata generation, challenge-response and verification. If the adversary is successful in any of these instances, that is, the *Verify* algorithm outputs 1, but the adversary's aggregate authenticators are not equal to

$$\sigma_1^{(k)} = \left\{ \prod_{(i,v_i)\in C} {\sigma_{1i}^{(k)}}^{v_i} \right\}_{k\in\omega},$$

$$\sigma_2^{(k)} = \left\{ \prod_{(i,v_i)\in C} g^{-s_i} \right\}_{k\in\omega},$$

$$\sigma_3^{(k)} = \left\{ \prod_{(i,v_i)\in C} g^{-r_i} \right\}_{k\in\omega},$$

the challenger declares failure and aborts.

We analyze the difference in success probabilities between Games 3 and 2. Assume that the file which cause the aborting contains n blocks, with name *name*, has generated exponents $\{\mu_j\}$ and has sectors $\{m_{ij}\}$, and the block authenticators issued by *Metadata* generation are $\{\sigma_{1i}^{(k)}, \sigma_{2i}^{(k)}, \sigma_{3i}^{(k)}\}$. Suppose $C = \{(i, v_i)\}$ is the query that leads to the challenger's failure, and the adversary's response to that query was μ'_1, $\mu'_2, \ldots, \mu'_s$ together with ${\sigma_1^{(k)}}', {\sigma_2^{(k)}}', {\sigma_3^{(k)}}'$. The difference in success probabilities between Games 3 and 2 is forging a valid aggregate authenticator for challenge C. With this in mind, we can prove that if there is a nonnegligible difference in the adversary's success probability between Games 3 and 2, there is another algorithm that can solve the Computational Diffie-Hellman problem.

The simulator $\mathcal{C}$ is given as inputs an instance of the Computational Diffie-Hellman problem (g, g^a, g^b), and its goal is to output the value of g^{ab}. Assume the adversary makes at most $l \ll p$ metadata queries. The simulation between the simulator $\mathcal{C}$ and the adversary $\mathcal{A}$ is as follows.

Setup: $\mathcal{C}$ picks a target attribute set α^* and sets $g_1 = g^a$ and $g_2 = g^b$. Then, the simulations are as follows. $\mathcal{C}$ Select a random $k \in \{0, 1, \ldots, n\}$. Next, $\mathcal{C}$ chooses random values $x', x_1, \ldots, x_s$ from $\{0, 1, \ldots, 2l - 1\}$ and random $z', z_1, \ldots, z_s$ from Z_p. $\mathcal{C}$ also needs to find a random polynomial $f(x)$ with degree m. And define an m degree polynomial $u(x)$ such that $u(x) = -x^m$ if and only if $x \in \alpha^*$. For $1 \le k \le n + 1$, set $t_k = g_2^{u(k)} g^{f(k)}$

Under this assignment, we implicitly set $T(k)$ as

$$T(k) = g_2^{k^n} \prod_{j=1}^{n+1} (g_2^{u(j)} g^{f(j)})^{\delta_{j,N}(k)} = g_2^{k^n + u(k)} g^{f(k)}.$$

$\mathcal{C}$ publishes the public parameters of the system as

$$\Big(g, g_1, g_2, t_1, \ldots, t_{n+1}, v' = g_2^{x' - 2kl} g^{z'},$$
$$\{u_j = g_2^{x_j} g^{z_j}\}_{1\le j\le s}, A = e(g_1, g_2)\Big).$$

The simulator keeps a list of hash table. For each $i(1 \le i \le n)$, the simulator picks a random $\rho_i \in Z_p$ and sets the random oracle at i as

$$H(name\|i) = g^{\rho_i}.$$

To respond a query on attribute set α^* of a file $M = \{m_{ij}\}_{1\le i\le n, 1\le j\le s}$, for the Lth block $\{m_{Lj}\}_{1\le j\le s}$, we first define

$$F_L = -2lk + x' + \sum_{j=1}^{s} x_j m_{Lj},$$

and

$$J_L = z' + \sum_{j=1}^{s} z_j m_{Lj}.$$

If $F_L = 0 \,(\mathrm{mod}\; p)$, $\mathcal{C}$ declares failure and aborts. Otherwise, $\mathcal{C}$ chooses a set $\Theta \subset \alpha^*$, where $|\Theta| = d - 1$ and for $k \in \Theta$, defines $g^{q'(k)} = g^{\lambda'_k}$ in which λ'_k are random elements in Z_p. For $k \in \alpha^* \setminus \Theta$, $\mathcal{C}$ computes $g^{q'(k)} = \prod_{j=1}^{d-1} g^{\lambda'_j \Delta_{j,\alpha^*}(k)}$. For $k \in \alpha^*$, $\mathcal{C}$ picks random $r_k^L, s_k^L \in Z_p$ and computes

$$\sigma_{1L}^{(k)} = \left\{ (g^{q'(k)})^{-\frac{J_L + \rho_L}{F_L}} g^{\rho_L s_k^L + f(k) r_k^L} (g^{J_L} g_2^{F_L})^{s_k^L} \right\}_{k\in\alpha^*},$$

$$\sigma_{2L}^{(k)} = \left\{ g^{r_k^L} \right\}_{k\in\alpha^*},$$

$$\sigma_{3L}^{(k)} = \left\{ (g^{q'(k)})^{-1/F_L} g^{s_k^L} \right\}_{k\in\alpha^*}.$$

It is obvious to show that $\{\sigma_{1L}^{(k)}, \sigma_{2L}^{(k)}, \sigma_{3L}^{(k)}\}_{k\in\alpha^*}$ is a valid authenticator for the block L where the random value $\hat{s}_k = s_k - q'(k)/F_L$

$$\begin{aligned}
\sigma_{1L}^{(k)} &= \left(g^{q'(k)}\right)^{-\frac{J_L + \rho_L}{F_L}} g^{\rho_L s_k^L + f(k) r_k^L} \left(g^{J_L} {g_2}^{F_L}\right)^{s_k^L} \\
&= \left({g_2}^{q'(k)}\right) g^{f(k) r_k^L} \left(g^{\rho_L + J} g_2^F\right)^{s_k^L - q(k)F} \\
&= \left({g_2}^{q'(k)}\right) g^{f(k) r_k^L} \left(g^{\rho_L + J} g_2^F\right)^{\hat{s}_k^L} \\
&= {g_2}^{q'(k)} T(k)^{r_k^L} \left(H(name\|L) h \cdot \prod_{j=1}^{s} {u_j}^{m_{Lj}}\right)^{\hat{s}_k^L}.
\end{aligned}$$

The simulator $\mathcal{C}$ is able to simulate the authenticators for all the blocks in this way.

Finally, the adversary $\mathcal{A}$ outputs a authenticator forgery $S^* = \{{\sigma_{1i}^{(k)}}^*, {\sigma_{2i}^{(k)}}^*, {\sigma_{3i}^{(k)}}^*\}_{k\in\alpha^*, 1\le i\le n}$ on file F^* for the attribute set α^*.

Assume ${F_L}^* = -2lk + x' + \sum_{j=1}^{s} x_j m_{Lj}^*$ and ${J_L}^* = z' + \sum_{j=1}^{s} z_j m_{Lj}^*$. If $|\gamma \cap \alpha^*| \ge d$ or if $F^* \ne 0 \,(\mathrm{mod}\; p)$, simulator $\mathcal{C}$ aborts. Otherwise, the Lth block authenticator forgery is $\{{\sigma_{1L}^{(k)}}^*, {\sigma_{2L}^{(k)}}^*, {\sigma_{3L}^{(k)}}^*\}_{k\in\alpha^*}$

Next, the simulator $\mathcal{C}$ will interact with adversary by executing challenge-response protocols until the situation defined in Game 3 occurs, that is the adversary produces a aggregate response value as $\left\{\sigma_1^{(k)*}, \sigma_2^{(k)*}, \sigma_3^{(k)*}\right\}_{k\in\alpha^*}$, but is not equal to the expecting value. Then the simulator $\mathcal{C}$ will carry out the following procedures.

Now the challenger $\mathcal{C}$ selects a random set $\Theta^* \subset \alpha^*$ where $|\Theta^*| = d$, and computes as follows:

$${\sigma_1^{(k)}}^* = \prod_{k\in\Theta^*} \left({S_{1L}}^*\right)^{\Delta_{k,\alpha}(0)},$$

$$\sigma_2^{(k)*} = \prod_{k\in\Theta^*}\left(S_{2L}{}^*\right)^{\Delta_{k,\alpha}(0)f(k)},$$
$$\sigma_3^{(k)*} = \prod_{k\in\Theta^*}\left(S_{3L}{}^*\right)^{\Delta_{i,\alpha}(0)}.$$

Then, the simulator $\mathcal{C}$ can solve computational Diffie-Hellman problem by calculating the following equation:

$$g^{ab} = \frac{\sigma_1^{(k)*}}{\sigma_2^{(k)*}\left(\sigma_3^{(k)*}\right)^{\rho_L{}^* J^*}}.$$

Game 4. Game 4 is the same as Game 3, with one difference. The challenger keeps a list of responses to metadata queries from the adversary. Now the challenger observes each instance of the protocol, including key extraction, metadata generation, challenge-response and verification. If in any of these instances the adversary is successful, that is, the *Verify* algorithm outputs 1, but at least one of the aggregate messages μ_j such that

$$\mu_j \neq \sum_{(i,v_i)\in C} v_i m_{ij},$$

where C is the challenge selected by the verifier, the challenger declares failure and aborts.

Again, we analyze the difference in success probabilities between Games 4 and 3. Suppose the file which leads to abort has n blocks, with name *name*, has generated exponents $\{\mu_j\}$ and contains sectors $\{m_{ij}\}$, and the block authenticators issued by *Metadata* generation are $\{\sigma_{1i}^{(k)}, \sigma_{2i}^{(k)}, \sigma_{3i}^{(k)}\}$. Suppose $C = \{(i, v_i)\}$ is the query that causes the challenger's failure, and the adversary's response to that query was $\mu'_1, \ldots, \mu'_s$ together with $\sigma_1^{(k)'}, \sigma_2^{(k)'}, \sigma_3^{(k)'}$. Let the expected response be $\mu_1, \ldots, \mu_s$ and $\{\sigma_1^{(k)}, \sigma_2^{(k)}, \sigma_3^{(k)}\}$, in which

$$\mu_j = \sum_{(i,v_i)\in C} v_i m_{ij},$$
$$\sigma_1^{(k)} = \left\{\prod_{(i,v_i)\in C} \sigma_{1i}^{(k)\,v_i}\right\}_{k\in\omega},$$
$$\sigma_2^{(k)} = \{\sigma_{2i}^{(k)}\}_{i\in C, k\in\omega},$$
$$\sigma_3^{(k)} = \{\sigma_{3i}^{(k)}\}_{i\in C, k\in\omega}.$$

Game 3 already guarantees that the authenticators of all the blocks are equal, and it is only the values μ'_j and μ_j that can be different. Define $\Delta\mu_j = \mu'_j - \mu_j$ for $1 \leq j \leq s$, there is at least one $\Delta\mu_j$ whose value is not zero since at least one of the aggregate messages μ_j is not equal to the expected value.

We show that if there is a nonnegligible difference in the adversary's success probability between Games 4 and 3, there is another algorithm that can solve the the discrete logarithm problem. The simulator $\mathcal{S}$ is given $g, t \in G$, and its goal is to output x such that $t = g^x$. $\mathcal{S}$ behaves like the Game 3 challenger, with the following differences.

1) To generate the metadata of the file with n blocks $\{m_{ij}\}_{1\leq i\leq n, 1\leq j\leq s}$, $\mathcal{S}$ randomly picks two values β_j, γ_j and sets $u_j = g^{\beta_j} h^{\gamma_j}$.
2) $\mathcal{S}$ interacts with the adversary persistently until the specified condition of Game 3 takes place.

Since all the blocks have the same authenticators and both the responses are valid, then we have the following two equations:

$$\prod_S e(\sigma_1^{(k)}, g) = \prod_S \prod_{i\in C} e\left(g_2{}^{q(k)}, g^{v_i}\right) e\left(T(k)^{r_k}, g^{v_i}\right)\cdot$$
$$e\left(\left(H(name\|i)v'\right)^{v_i}, g^{s_k}\right) e\left(\prod_{j=1}^{s} u_j{}^{\mu_j}, g^{s_k}\right),$$

and

$$\prod_S e(\sigma_1^{(k)}, g) = \prod_S \prod_{i\in C} e\left(g_2{}^{q(k)}, g^{v_i}\right) e\left(T(k)^{r_k}, g^{v_i}\right)\cdot$$
$$e\left(\left(H(name\|i)v'\right)^{v_i}, g^{s_i}\right) e\left(\prod_{j=1}^{s} u_j{}^{\mu'_j}, g^{s_k}\right).$$

Therefore, we can get

$$\prod_{j=1}^{s} u_j{}^{\mu_j} = \prod_{j=1}^{s} u_j{}^{\mu'_j}.$$

Since $u_j = g^{\beta_j} t^{\gamma_j}$, we have

$$g^{\sum_{j=1}^{s}\beta_j\Delta\mu_j} t^{\sum_{j=1}^{s}\gamma_j\Delta\mu_j} = 1.$$

Because there is at least one of the $\{\Delta\mu_j\}$ is nonzero, finally, we get the solution to the given instance of discrete logarithm as follows:

$$t = g^{-\frac{\sum_{j=1}^{s}\beta_j\Delta\mu_j}{\sum_{j=1}^{s}\gamma_j\Delta\mu_j}}.$$

Wrapping up. In Game 4, the adversary can not answer a query in other ways, but generate a response by the Response algorithm in the protocol defined in Section IV. Therefore we can see that, assuming the employed attribute-based signature algorithm is unforgeable, and the Computational Diffie-Hellman problem and discrete logarithm problem are hard in bilinear groups, then there is only a negligible difference in the success probability for the adversary in Game 4 compared with Game 0. This completes the proof of this *Theorem.*

VI. IMPLEMENTATION

In this section, we report the performance of the proposed protocol. In our implementation, all the algorithms are conducted on a Win 8 64-bit laptop with Intel Core (TM) i5-4300 @ 2.49 GHz CPU and an 8 GB SSD. The projects are written in C++ language under Visual Studio 2010 compiler and we call the Miracl library [34] API to construct elliptic

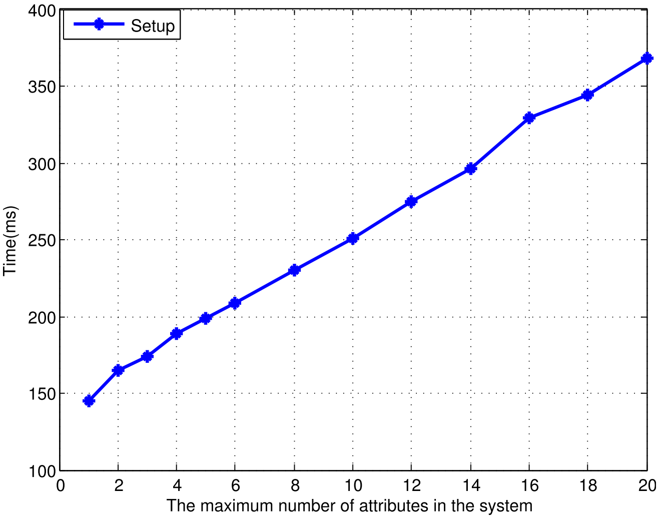


FIGURE 3. **Time consumption for *Setup* algorithm.**

curves. We choose the Cocks-Pinch curve [35] as $y^2 = x^3 - 3x + B \pmod p$, which is ideal for the security level AES-80 bits. Accordingly, p should be a 160-bit prime and $p = 3 \bmod 4$. Tate pairing with embedding degree $k = 2$ is used to implement an asymmetric bilinear map, $e : G_1 \times G_2 \rightarrow G_T$, where G_1 is a point over the base field and G_2 is a point on the quadratic twist. The implementation results will be shown from the following aspects.

In the first part, we present the time consumption of both *Setup* and *Extract* algorithms. As can be seen from Figure 3, the time cost of the *Setup* algorithm exhibits a strictly linear growth with the maximum number of attributes m in the system. This is due to the fact that the function T needs to perform m multiplications. Thus, with the increasing of m, the time cost of Setup will increase multiply as well. Figure 4 shows that the time consumption of *Extract* algorithm grows linearly with the number of attributes required for a user. The results are consistent with our empirical analysis, since the user's private key is calculated for each attribute in a user's attribute set, so the more attributes an identity includes, the longer it takes for the key extraction algorithm.

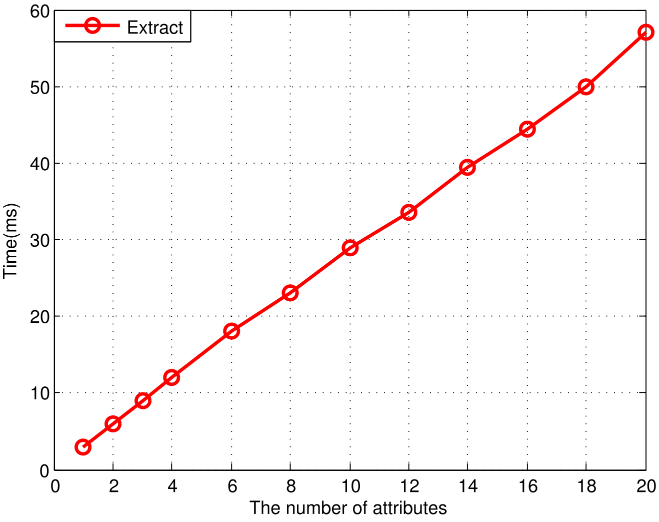


FIGURE 4. **Time consumption for *Extract* algorithm.**

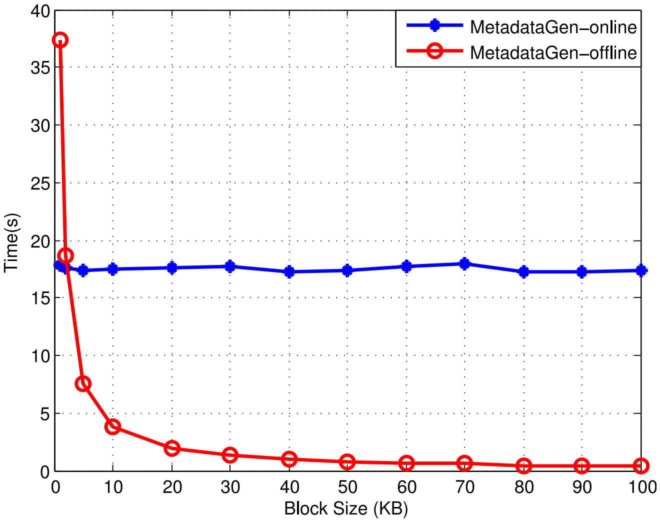


FIGURE 5. **Time consumption for *MetadataGen* algorithm of 1 MB file.**

In the second part, we test the time consumption of generating the metadata for a file. We choose a file with a fixed size of 1 MB and select the maximum number of attributes in a set to be 10, three of which to describe a user's attribute information. The block size varies from 1 KB to 100 KB with the increment of 10 KB. We divide the MetadataGen algorithm into two parts, say, online and offline phase, where the offline phase refers to the portion that can be calculated before the uploaded file is selected and the online phase is the portion that must be determined after obtaining the file. Since the off-line part changes rapidly in the range of 1-10 KB, four points are added in this interval to observe the trend of the curve. As can be seen from the results in Figure 5, the time of generating the metadata file at the online stage is a constant, about 17 seconds. This is because the online phase is mainly for the calculation $\prod_{j=1}^{s} u_j{}^{m_{ij}}$, and with a fixed sector size 160 bits, the number of sectors for 1 MB file is a constant 51,200. So the calculation of the online phase is a constant time consumption, which is independent of the block size. However, the time cost of the offline phase almost increases with the number of data blocks increasing, since there are three parts of the data block that need to be calculated for each data block. Therefore, the more blocks in a file, the more authenticators to be calculated, which takes more time consumption.

In the third part, we try to figure out the optimal block size by analyzing the tradeoff time consumption between *MetadataGen* and *Audit* algorithm for a fixed size of 1 MB file. Since the sector size is a constant, which is 160 bits, the number of sectors per data block increases as each block size increases, which results in an increase in the computational cost in the MetadataGen algorithm. However, when the block size raises, the number of data blocks will reduce(note that we challenge all the data blocks in the experiment), so it

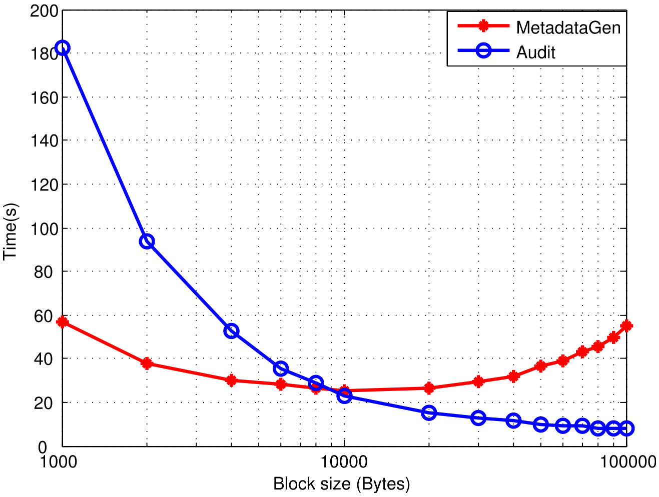


FIGURE 6. **Tradeoff between *MetadataGen* and *Audit* for 1 MB file.**

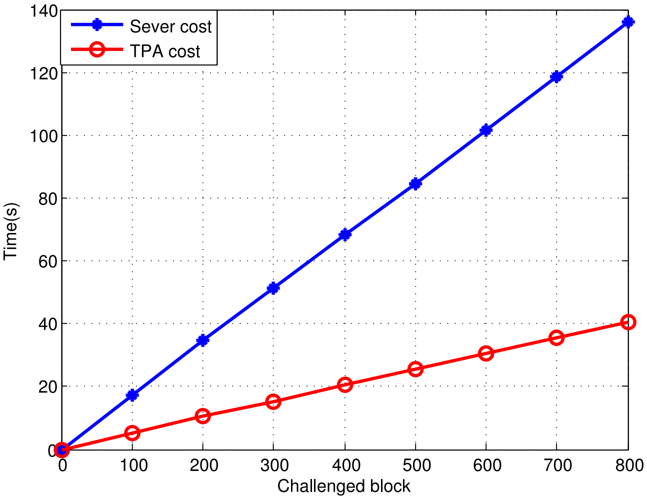


FIGURE 7. **Challenged blocks for a file of 10,000 blocks.**

will lead to less calculation in verifying $\{\sigma_1^{(k)}, \sigma_2^{(k)}, \sigma_3^{(k)}\}_{k \in A}$ in *Response* and *verify* algorithm, and thus cause less time consumption. From Figure 6, we can see that the algorithms have the best performance when the block size is between 6 to 20 KB. Thus, we chose 10 KB as the optimal block size to reduce the user's computational cost. When the block size is 10 KB, the time cost of the *MetadataGen* algorithm is 25.1 seconds.

In the fourth part, we choose a file containing 10,000 blocks, and each block is of 10 KB. We push up the number of challenged blocks from 100 to 800, with the increment of 100 for each step, to test the performance of the cloud server and TPA server. As can be seen from Figure 7, the cost of the cloud server and the TPA server grows linearly as the number of challenged blocks increases. According to Atenises *et al.* [8], if 1 percent of all blocks are damaged, only 300 and 460 blocks out of 10,000 blocks are needed to challenge that can detect the misbehaviour of the cloud server at 95 and 99 percent probability respectively. As we can see that when the number of challenged block to be 300, TPA costs 15.1 seconds, while the cloud server needs 51.3 seconds; when it comes to 460, TPA spends 23.1 seconds, while the cloud server requires 81.7 seconds.

VII. CONCLUSION

In the past few years, cloud data integrity has drawn much attention from both academia and industry. In this paper, we propose an attribute-based cloud data integrity auditing protocol, for the first time, to simplify the key management issue in traditional cloud data auditing schemes. We formalize the system model and security model for this new primitive. Subsequently, a concrete construction is presented by involving the idea of attribute-based cryptography. The proposed protocol can achieve the property of soundness, attribute privacy-preserving and collusion resistance. We prove the soundness of the protocol under Shacham-Waters game-based proof framework. The implementation illustrates the practicality and efficiency of the new proposal.

Future Work. The construction in Section IV provides a privacy-preserving guarantee that reveals nothing but the d common attributes chosen by cloud server when executing the auditing protocols. The authors are investigating a strong privacy-preserving mechanism that can ensure zero-knowledge in the auditing phase. Future work includes proposing a concrete construction that are both practical and with high efficiency.

ACKNOWLEDGMENTS

The authors would like to thank the anonymous referees sincerely for their very valuable comments. This work is supported by the National Key R&D Program of China (2017YFB0802000), the National Natural Science Foundation of China (61772326, 61572303), NSFC Research Fund for International Young Scientists (61750110528), National Cryptography Development Fund during the 13th Five-year Plan Period (MMJJ20170216), the Foundation of State Key Laboratory of Information Security (2017-MS-03) and the Fundamental Research Funds for the Central Universities (GK201702004). Y. Yu and Y. Li contributed equally to this work and should be considered as the co-first authors.

REFERENCES

[1] M. Hogan, F. Liu, A. Sokol, and J. Tong, "NIST cloud computing standards roadmap," *NIST Cloud Computing Standards Roadmap Working Group*, Nat. Inst. Standards Technol., U.S. Dept. Commerce, Gaithersburg, MD, USA, Tech. Rep. 500-291-v1.0, Jul. 2011.

[2] Y. Deswarte, J. J. Quisquater, and A. Saidane, "Remote integrity checking," *Integrity and Internal Control in Information Systems VI*. New York, NY, USA: Springer, 2004, pp. 1–11.

[3] G. Filho, D. Luiz, and P. S. L. M. Barreto, "Demonstrating data possession and uncheatable data transfer," IACR Cryptology ePrint Archive, 2006.

[4] G. Yamamoto, S. Oda, and K. Aoki, "Fast integrity for large data," in *Proc. ECRYPT Workshop Softw. Perform. Enhancement Encryption Decryption*, 2007, pp. 21–32.

[5] K. Chida and G. Yamamoto, "Batch processing of interactive proofs," in *Proc. Cryptographers Track RSA Conf.*, 2007, pp. 196–207.

[6] F. Sebe, A. Martinez-Balleste, Y. Deswarte, J. Domingo-Ferre, and J.-J. Quisquater, "Time-bounded remote file integrity checking," Laboratory Anal. Archit. Syst., Toulouse, France, Tech. Rep. 04429, 2004.

[7] A. Juels and B. S. Kaliski Jr, “PORs: Proofs of retrievability for large files,” in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.

[8] G. Ateniese, *et al.*, “Provable data possession at untrusted stores,” in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[9] G. Ateniese, S. Kamara, and J. Katz, “Proofs of storage from homomorphic identification protocols,” in *Proc. 15th Int. Conf. Theory Appl. Cryptology Inf. Secur.: Advances Cryptology*, 2009, pp. 319–333.

[10] C. C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, “Dynamic provable data possession,” *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, 2015, Art. no. 15.

[11] H. Shacham and B. Waters, “Compact proofs of retrievability,” in *Proc. 14th Int. Conf. Theory Appl. Cryptology Inf. Secur.: Advances Cryptology*, 2008, pp. 90–107.

[12] D. Boneh, B. Lynn, and H. Shacham, “Short signatures from the weil pairing,” in *Proc. 7th Int. Conf. Theory Appl. Cryptology Inf. Secur.: Advances Cryptology*, 2001, pp. 514–532.

[13] Y. Yu, J. B. Ni, M. H. Au, H. Y. Liu, H. Wang, and C. X. Xu, “Improved security of a dynamic remote data possession checking protocol for cloud storage,” *Expert Syst. Appl.*, vol. 41, no. 17, pp. 7789–7796, 2014.

[14] Y. Yu, *et al.*“Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage,” *Int. J. Inf. Secur.*, vol. 14, no. 4, pp. 307–318, 2015.

[15] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, “Privacy-preserving public auditing protocol for low performance end devices in cloud,” *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2572–2583, Nov. 2016.

[16] C. Wang, B. Zhang, K. Ren, and J. M. Roveda, “Privacy-assured outsourcing of image reconstruction service in cloud,” *IEEE Trans. Emerging Topics Comput.*, vol. 1, no. 1, pp. 166–177, Jun. 2013.

[17] C. Wang, K. Ren, W. Lou, and J. Li, “Toward publicly auditable secure cloud data storage services,” *IEEE Netw.*, vol. 24, no. 4, pp. 19–24, Jul./Aug. 2010.

[18] Y. Yu, J. B. Ni, M. H. Au, Y. Mu, B. Y. Wang, and H. Li, “Comments on a public auditing mechanism for shared cloud data service,” *IEEE Trans. Serv. Comput.*, vol. 8, no. 6, pp. 998–999, Nov./Dec. 2015.

[19] Y. Zhang, J. Ni, X. Y. Wang, and Y. Yu, “Provable multiple replication data possession with full dynamics for secure cloud storage,” *Concurrency Comput.: Practice Experience*, vol. 28, no. 4, pp. 1161–1173, 2016.

[20] A. Shamir, “Identity-based cryptosystems and signature schemes,” in *Proc. CRYPTO 84 Advances Cryptology*, 1985, pp. 47–53.

[21] J. N. Zhao, C. X. Xu, F. G. Li, and W. Z. Zhang, “Identity-based public verification with privacy-preserving for data storage security in cloud computing,” *Inst. Electron. Inf. Commun. Eng. Trans.*, vol. 96-A, no. 12, pp. 2709–2716, 2013.

[22] Y. Yu, *et al.*, “Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage,” *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[23] Y. Yu, Y. F. Zhang, Y. Mu, W. Susilo, and H. Y. Liu, “Provably secure identity based provable data possession,” in *Proc. Int. Conf. Provable Secur.*, 2015, pp. 310–325.

[24] H. Q. Wang, “Identity-based distributed provable data possession in multicloud storage,” *IEEE Trans. Serv. Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.

[25] S. Wang, K. Liang, J. K. Liu, J. Chen, J. Yu, and W. Xie, “Attribute-based data sharing scheme revisited in cloud computing,” *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 8, pp. 1661–1673, Aug. 2016.

[26] K. Liang, *et al.*, “A secure and expressive ciphertext-policy attribute-based proxy re-encryption for cloud data sharing,” *Future Generation Comput. Syst.*, vol. 52, pp. 95–108, 2015.

[27] J. K. Liu, M. H. Au, X. Huang, R. Lu, and J. Li, “Fine-grained two-factor access control for web-based cloud computing services,” *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 3, pp. 484–497, Mar. 2016.

[28] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, “Fuzzy identity-based data integrity auditing for reliable cloud storage systems,” *IEEE Trans. Depend. Secure Comput.*, doi: 10.1109/TDSC.2017.2662216.

[29] S. F. Shahandashti and R. Safavi-Naini, “Threshold attribute-based signatures and their application to anonymous credential systems,” in *Proc. 2nd Int. Conf. Cryptology Africa: Progress Cryptology*, 2009, pp. 198–216.

[30] D. Boneh and M. Franklin, “Identity-based encryption from the weil pairing,” in *Proc. 21st Annu. Int. Cryptology Conf. Advances Cryptology*, 2001, pp. 213–229.

[31] J. Katz, *Digital Signatures*. Berlin, Germany: Springer, 2010.

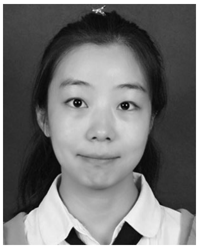
[32] A. Shamir, “How to share a secret,” *Commun. ACM*, vol. 22, no. 11, pp. 612–613, 1979.

[33] H. Maji, M. Prabhakaran, and M. Rosulek, “Attribute based signatures: Achieving attribute privacy and collusion-resistance,” 2008. [Online]. Available: http://eprint.iacr.org/2008/328

[34] [Online]. Available: https://certivox.org/display/EXT/MIRACL

[35] D. Freeman, M. Scott, and E. Teske, “A taxonomy of pairing-friendly elliptic curves,” *J. Cryptology*, vol. 23, no. 2, pp. 224–280, 2010.

YONG YU received the PhD degree in cryptography from Xidian University, in 2008. He is currently a professor of Shaanxi Normal University, China. He holds the prestigious one hundred talent Professorship of Shaanxi Province as well. He has authored more than 50 referred journal and conference papers. His research interests include cryptography and its applications, especially public encryption, digital signature, and secure cloud computing. He is an associate editor of Soft Computing. He is a member of the IEEE.



YANNAN LI received the bachelor’s and master’s degrees from the University of Electronic Science and Technology of China, in 2014 and 2017, respectively. She is currently working toward the PhD degree in the School of Computing and Information Technology, University of Wollongong, Australia. Her research interests include digital signatures and secure cloud storage. She is a student member of the IEEE.

BO YANG received the BS degree from Peking University, Beijing, China, in 1986, and the MS and PhD degrees from Xidian University, Xi’an, China, in 1993 and 1999, respectively. From 1986 to 2005, he was with Xidian University, where he was a professor of National Key Laboratory of ISN and the PhD supervisor from 2002 to 2005. He has served as a program chair of the Fourth China Conference on Information and Communications Security in 2005, a vice chair of the Conference of the Chinese Association for Cryptologic Research in 2009, and a general chair of the Fifth Joint Workshop on Information Security in 2010. He is currently a professor and PhD supervisor in the School of Computer Science, Shaanxi Normal University, Xi’an, and a special term professor of Shaanxi Province. His research interests include information theory and cryptography.

WILLY SUSILO (SM’01) received the PhD degree in computer science from the University of Wollongong, Australia. He is a professor and the head of the School of Computing and Information Technology and the director of the Institute of Cybersecurity and Cryptology (iC^2), University of Wollongong. He was previously awarded the prestigious ARC Future Fellow by the Australian Research Council (ARC) and the Researcher of the Year award in 2016 by the University of Wollongong. His main research interests include cybersecurity, cryptography, and information security. His work has been cited more than 9,000 times in Google Scholar. He is the editor-in-chief of the *Information Journal*. He has served as a program committee member in dozens of international conferences. He is currently serving as an associate editors in several international journals, including the *Elsevier Computer Standards and Interface* and the *International Journal of Information Security (IJIS, Springer)*. He has published more than 300 research papers in the area of cybersecurity and cryptology. He is a senior member of IEEE.

GUOMIN YANG received the PhD degree in computer science from the City University of Hong Kong, Hong Kong, in 2009. He is currently a senior lecturer and a DECRA fellow in the School of Computing and Information Technology, University of Wollongong, Wollongong, NSW, Australia. His current research interests include applied cryptography and network security. He is a member of the IEEE.

JIAN BAI received the MS degree in cryptography from Xidian University, in 2014. He is currently an engineer of Science and Technology on Communication Security Laboratory. His main research interests include information security and cryptography.