IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received July 22, 2019, accepted July 29, 2019, date of publication August 1, 2019, date of current version August 29, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2932430

Identity-Based Auditing for Shared Cloud Data With Efficient and Secure Sensitive Information Hiding

YU FAN[1], YONGJIAN LIAO[1], (Member, IEEE), FAGEN LI[2], (Member, IEEE), SHIJIE ZHOU[1], AND GANGLIN ZHANG[1]

[1]School of Information and Software Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China
[2]School of Computer Science and Engineering, University of Electronic Science and Technology of China, Chengdu 610054, China

Corresponding author: Yongjian Liao (liaoyj@uestc.edu.cn)

This work was supported by the Sichuan Science and Technology Program under Grant 2018GZ0180, Grant 2018GZ0085, Grant 2017GZDZX0001, and Grant 2017GZDZX0002.

ABSTRACT The advent of cloud computing arouses the flourish of data sharing, promoting the development of research, especially in the fields of data analysis, artificial intelligence, etc. In order to address sensitive information hiding, auditing shared data efficiently and malicious manager preventing, we propose an identity-based auditing scheme for shared cloud data with a secure mechanism to hide sensitive information. This scheme provides a solution that allows users to share plaintext with researchers and keeps sensitive information invisible to the cloud and researchers at the same time. Besides, a formal security analysis is given to prove the strong security of the proposed scheme. Performance evaluation and experimental results demonstrate that our scheme is significantly more efficient over the existing scheme due to our novel mechanism for sensitive information hiding and simplifying signature algorithm. Compared to the existing approach to audit the integrity of shared data with sensitive information hiding, our scheme has desirable features and advantages as follow. Firstly, previous work has failed to construct a secure scheme to prevent malicious manager. We fill this gap and guarantee the integrity and authenticity of shared data. Secondly, our scheme constructs a novel system model to support high concurrency and massive data in the real scenario.

INDEX TERMS Cloud computing, shared cloud data, remote auditing, sensitive information hiding, malicious manager preventing.

I. INTRODUCTION

Cloud computing has tremendous computing and storage capacity, opening up a whole new world of services, platforms, and applications. Cloud computing services, such as Google Drive and Amazon's S3, have attracted a large number of individuals, companies, and public organizations to store data in the cloud rather than local fixed devices. Also, cloud services make it easier for cloud users to share data over the Internet. Therefore, convenient access methods provided by the cloud has stimulated the development of research, especially in the fields of data analysis, artificial intelligence and so on, which are based on a large amount of data and samples. Open data is a typical and inspiring example, shared primarily by commercial companies and public organizations, and used to create derivative value for researchers.

Consider the real scenario illustrated in Fig. 1, where Electronic Health Records(EHRs) [1], which are extensively used in hospitals and hold the characteristics of shared data. An EHR usually contains sensitive information about patients and hospitals. For fear that such kind of information is exposed to the cloud server and researchers, the doctor who generates the EHR should hide it before submitting to the EHR system. The administrator of the EHR system [2], [3] has a large number of privileges in the system, usually served by the hospital's manager. The administrator is more concerned about sensitive information hiding than doctors because the leak of sensitive information caused by software failure or human error would adversely affect the hospital's reputation. Thus, the administrator needs to inspect the content of EHRs and then uploads the EHRs to the cloud

The associate editor coordinating the review of this article and approving it for publication was Yinghui Zhang.

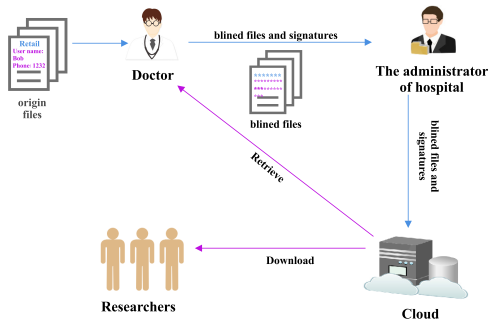


FIGURE 1. An application of shared data.

server if the sensitive information has been hidden entirely. By the above process, doctors can directly retrieve EHRs from the cloud server and recover complete EHRs, while researchers can download shared EHRs without any sensitive information.

This scenario reveals that sensitive information hiding is an essential step in data sharing. Besides, two security concerns in the scenario cannot be ignored. Firstly, the shared data in the cloud server is still at the risk of being corrupted and lost due to the existence of hardware/software failures and human errors [4], [5]. Moreover, an untrusted cloud server may conceal the fact of data lost. It is essential to efficiently audit the integrity of shared data in the cloud environment. Secondly, the administrator has so great right in the system that he can modify the shared data before uploading to the cloud server. The administrator may modify the EHR associated with a medical incident for maintaining the hospital's reputation. Anyhow, due to the malicious modification, the integrity and authenticity of shared data cannot be trusted by doctors and researchers. Therefore, it is essential to prevent a malicious administrator from tampering with data.

It can be concluded that a solution to the security concerns of data sharing as mentioned above should meet the following properties:

- **Sensitive Information Hiding.** The sensitive information in shared data cannot be exposed to the cloud server and researchers. The traditional method is to encrypt the entire file, but it does not satisfy the demand for shared data because the encrypted data cannot be used by researchers efficiently. Distributing the decryption key to the researchers seems to be helpful for them to obtain the original shared. However, in the real scenario, a data owner has no way to know the researchers who will use the shared file. As a result, it is impractical to hide sensitive information by encrypting the whole shared file. Therefore, the solution for sensitive information preventing should hide the part with sensitive information, while the other part is published.
- **Malicious Manager Preventing.** The solution can prevent a malicious manager from tampering with shared data. More specifically, it is infeasible that the signatures forged by the manager pass the verification of the cloud server and auditor.
- **Data Integrity Auditing.** In order to guarantee the integrity of shared data and prevent the dishonest operation of the cloud server, the solution should provide an efficient remote auditing mechanism, which satisfies the condition that the sensitive parts of data are hidden.

Most studies in the field of remote auditing have not dealt with sensitive information hiding. This gap was filled by Shen recently, who creatively proposed an identity-based auditing scheme for shared data with sensitive information hiding [6]. In their scheme, the manager of an organization is given the right to transform the signature to a valid one after hiding the sensitive information about the organization. That means the manager can modify data or even forge new data as he wishes and calculate corresponding signatures. Unfortunately, such malicious behavior cannot be detected by their scheme. Shen's scheme has failed to satisfy the property that prevents the malicious manager. Therefore, the problem in this paper is how to construct an efficient and secure auditing scheme for shared data, which supports sensitive information hiding and malicious manager preventing.

A. RELATED WORK

In order to enable cloud users to verify the integrity and availability of their data in the cloud server, a considerable amount of proposals have been made. The Provable Data Possession(PDP), introduced by Ateniese *et al.* [7], is a groundbreaking work in those studies. In their scheme, a file owner only consumes a low storage space to keep metadata for a file. The file is uploaded by the owner to the cloud server along with the corresponding metadata and deleted from the local. Then, the file owner or public auditor can randomly challenge the data blocks of this file in the cloud server. After receiving the challenge, the cloud server needs to generate a proof based on the file and metadata it holds. By validating this proof, the auditor can check the integrity of the file. It can be seen that the PDP model allows users or public auditors to check whether the cloud server holds an intact file without having to retrieve the entire file. The PDP model, however, has failed to address dynamic data auditing. So Ateniese *et al.* [8] further proposed a dynamic PDP scheme that implements a reliable way to modify and delete data in the cloud server, but does not support inserting data. Erway *et al.* [9] introduced a full dynamic PDP model that implements all types of data manipulation. Subsequently, in order to enhance the efficiency of full dynamic operations, a series of data structures [10]–[12] were employed to construct novel auditing schemes. More recent progress in the full dynamic model is that Sookhak *et al.* [13] proposed an auditing scheme for big data.

With the advent of PB-level data and the growing focus on data derivative value, data sharing as a fundamental application in the cloud environment has led to a large and increasing body of related work. Wang *et al.* [14] first proposed Oruta for fear that public auditors might steal the identity privacy of signers in a shared data group. In their scheme, the Homomorphic Verification Tags(HVT) based on ring

signature makes it infeasible for the public auditor to detect who is the signer through the difference among signatures. In order to address the issues in the Oruta, Wang et al. further proposed Panda [15], which can revoke user access to shared data efficiently, and Knox [16], which possesses an excellent performance in a large group setting. Yang *et al.* [17] proposed an auditing scheme with identity privacy preserving and identity traceability, enabling a group manager to determine who maliciously modified shared data. More recent work [18], [19] has focused on the shared data in the large user setting and Internet of medical things.

Such approaches, however, are limited to complex certificate management, which causes non-negligible communication and computation costs. The HVT based on identity/certificateless signature [20] is the promising solution for diminishing reliance on certificates. Smart *et al.* [21] and Wang *et al.* [22] creatively constructed a certificateless public auditing mechanism. Wang [23] introduced an identity-based PDP model that supports the public auditing in a distributed cloud environment while eliminating certificate management. The experimental results of a considerable number of work [22]–[25] demonstrate that this solution can enhance the efficiency of auditing schemes. Subsequent studies [26]–[30] have focused on combining this aspect with identity privacy preserving and user revocation in data sharing. He *et al.* [26] proposed a privacy-preserving auditing mechanism with blockless verification based on a certificateless signature they introduced. With the same goal, Yu *et al.* [27] proposed an identity-based scheme supporting privacy-preserving. Recently, Tian and Jing [31] proposed a lightweight auditing scheme which supports both anti-replay/replace attack and agent security. In additional, to solve the key exposure propose in identity-based scheme, Yu *et al.* [32] proposed a scheme with strong key-exposure resilient auditing.

In the real scenario, shared data, published by an organization, usually contains sensitive information about both individuals and the organization. This crucial point has been ignored by the abovementioned studies. Until recently, Shen *et al.* [6] stated the importance of sensitive information hiding in detail and creatively proposed a remote auditing scheme supporting such essential attribute. The introduction of this scheme also brings some potential influence to other aspect, such as decreasing the overhead of the ciphertext query [33]–[36], and verification in the untrusted cloud server [37], [38]. However, we observe that three issues are not addressed well in their scheme. Firstly, it cannot prevent a malicious manager from tampering with or forge a file. Secondly, some redundant responsibilities of the manager impede the efficiency of system operation in the practical application. Especially facing the case of massive shared data and high concurrency, insufficient bandwidth and computing power at the manager side will cause the computational and communication bottlenecks in the whole scheme. Thirdly, in their mechanism for sensitive information hiding, users must keep random values to recover the data in the future. That takes a storage burden that cannot be ignored to users. In addition, if local random values are lost due to hardware failure or human error, the data in the cloud cannot be recovered.

B. OUR CONTRIBUTION

In view of above issues, this paper proposes a novel and efficient identity-based auditing scheme, supporting sensitive information hiding and malicious manager preventing. Compared to previous work, our scheme has the following distinguishing features:

- We propose a novel mechanism of sensitive information hiding, which can prevent the malicious manager. In the mechanism, the user possesses the unique signing right to the files he owns. The manager, whose responsibilities are similar to a gateway in the computer network, only has the right to check whether the content of the file contains sensitive information. In addition, our auditing scheme can detect any malicious file operations by the manager. We also give the formal proof, guaranteeing that our mechanism achieves sensitive information hiding and malicious manager preventing.
- We propose an efficient identity-based auditing scheme for shared cloud data, constructing a novel system model to support high concurrency and massive data in the real scenario. Centralized computing tasks (e.g., hiding the organization's sensitive information), which are redundant for the manager, are distributed to users. And users directly retrieve the necessary data from the cloud server rather than send a request to the manager. These two crucial strategies reduce the probability of the advent of computational and communication bottlenecks. Besides, a novel algorithm is proposed to hide sensitive information without random values, reducing the storage costs at the user side. The identity-based cryptography is employed to eliminate the complicated certificate management.

In addition to the above work, we prove the security of our scheme that guarantees the soundness of shared data. A series of experiments are designed to compare the performance with previous work. Theoretical analysis and experimental results demonstrate that our work significantly enhances the performance and efficiency of auditing.

C. ORGANIZATION

We review some preliminaries and give the definition of system model in Section II. The definition of system components and security model are portraied in Section III. The detailed construction of our scheme is described in Section IV. Section V, Section VI, Section VII demonstrate the security and performance of the proposed scheme by theoretical analysis and numerous experiment. At last, we give a conclusion of this paper in Section VIII.

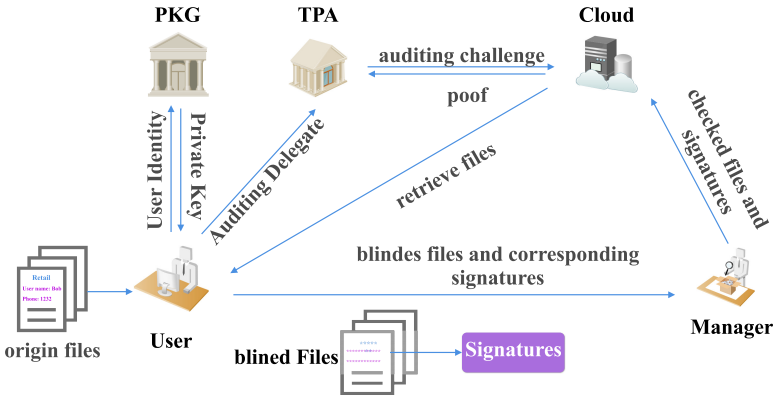


FIGURE 2. System framework.

II. PRELIMINARIES & SYSTEM MODEL

A. BILINEAR MAP

G_1 and G_T denote two multiplicative cyclic groups with the same prime order p. g is a generator of G_1. A bilinear map $e : G_1 \times G_1 \rightarrow G_T$ meets the following conditions:

- **Bilinearity**: $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$ for all $g_1, g_2 \in G_1$ and $a, b \in Z_p^*$.
- **Non-degenerate**: $e(g, g) \neq 1$, where 1 is an identity element in G_1.
- **Computational**: $e(g_1, g_2)$ can be computed efficiently for all $g_1, g_2 \in G_1$.

B. COMPUTATIONAL DIFFIE-HELLMAN (CDH) ASSUMPTION

Given the tuple (g, g^a, g^b) by the challenger, the adversary $\mathcal{A}$ intends to obtain g^{ab}, where g is a generator of multiplicative group G, and $a, b \in Z_p^*$. The adversary $\mathcal{A}$ has the advantage ϵ in solving the computational Diffie-Hellman problem if

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

Definition 1: If no adversary can solve the above problem with advantage ϵ, it can be concluded that the CDH assumption holds in G.

C. DISCRETE LOGARITHM(DL) ASSUMPTION

Given the tuple (g, g^a), the adversary $\mathcal{A}$ intends to obtain $a \in Z_p^*$, where g is a generator of a multiplicative cyclic group G. The adversary $\mathcal{A}$ has the advantage ϵ in solving discrete logarithm problem if

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

Definition 2: If no adversary can solve the above problem with advantage ϵ, it can be concluded that the DL assumption holds in G.

D. SYSTEM MODEL

Our system model, as illustrated in Fig. 2, consists of five entities: the user, the manager, the cloud, the Private Key Generator(PKG) and the Third Party Auditor(TPA). The respective duties of entities are given in detail as follows.

- *User*. The user, such as the doctor in Section I, who is an employee in an organization, generates and possesses numerous files. He should handle all data blocks with sensitive information in the files and submit them to his superior manager.
- *Manager*. The manager is in charge of gathering all files submitted by subordinate users and inspecting whether the sensitive information, especially the information about the organization, has been hidden by its owner. The manager rejects the files, which still expose sensitive information; otherwise, the manager uploads the files and corresponding signatures to the cloud server. Significantly, the manager is semi-trusted, having no right to modify any files even if he finds some sensitive information in the files.
- *Cloud*. The cloud is a semi-trusted entity. It has a tremendous storage ability, allowing the user to upload massive amounts of data and share data with the public.
- *The Third Party Auditing*. The TPA is a fully-trusted entity. It has much more ample resources and capabilities than users, offering a professional auditing service for the user.
- *The Private Key Generator*. The PKG is a fully-trusted entity. It is in charge of initializing the public parameters for the system and providing the private key for the user. It also allows the user, who possesses the private key, access the public parameters.

III. SYSTEM COMPONENTS & SECURITY MODEL

A. SYSTEM COMPONENTS

In this section, we will introduce the system component and security model of our scheme. Our scheme consists of six components: *Setup*, *PKGen*, *SigGen*, *Check*, *ProofGen*, and *ProofVerify*. The algorithms of components are given in detail as follows.

- $Setup(1^k) \rightarrow (Para, msk)$. The PKG firstly initializes the system setup by performing this probabilistic algorithm, which inputs a security parameter 1^k, and then

publishes some public parameters *Para* in the system, keeps the master secret key *msk* for the PKG.

- $PKGen(Para, msk, ID) \rightarrow (sk_{ID})$. After receiving a user's request with his identity *ID*, the PKG generates a private key sk_{ID} for the user by performing *PKGen* algorithm, which inputs the public parameters *Para*, the PKG's secret key *msk*, and $ID \in \{0,1\}^*$. Then the PKG responds the user with sk_{ID}.
- $SigGen(Para, sk_{ID}, M, Mname) \rightarrow (M', \Phi, \tau, R)$. The user validates the private key sk_{ID} received from the PKG and generates signatures corresponding to files by performing *SigGen* algorithm, which inputs the public parameters *Para*, sk_{ID}, a file M, the file's unique name *Mname*. The user obtains the blinded file M' without the sensitive information, the signatures set Φ corresponding to the M', the tag τ of the M', a verification value R.
- $Check(Para, M', \Phi, \tau) \rightarrow \{upload, reject\}$. The manager inspects the content of the blinded file M' submitted from the user by performing *Check* algorithm, which inputs the public parameters *Para*, M', the signatures set Φ, the tag τ. If no sensitive information about both individuals and organizations is found in the file, the manager *uploads* the file M' and the signatures set Φ to the cloud server; otherwise, he *rejects* this file.
- $ProofGen(Para, M', \Phi, chal) \rightarrow (P)$. A preparation that the TPA needs to do is to generate a challenge *chal*, where *chal* consists of two sets of random values. After receiving a challenge from the TPA, the cloud server returns a response with a proof P by performing *ProofGen* algorithm, which inputs the public parameters *Para*, the blinded file M', the signatures set Φ and *chal*.
- $ProofVerify(Para, chal, P) \rightarrow \{intact, not-intact\}$. The TPA validates the auditing proof P from the cloud by performing algorithm *ProofVerify*, which inputs the public parameters *Para*, the challenge *chal*, and the proof P. If P is a valid value, it can be concluded that the challenged file M' is *intact*; otherwise, M' is *not intact*.

B. SECURITY MODEL

We proceed to define the security model for our scheme against the semi-trusted cloud server (Soundness). In the security model, a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ is defined to simulate the semi-trusted cloud server; And a challenger $\mathcal{C}$ is defined to simulate the data owner. The basic game between the adversary $\mathcal{A}$ and challenger $\mathcal{C}$ is given in detail as follow.

Setup: The challenger $\mathcal{C}$ firstly initializes the system setup by performing algorithm *Setup*, which inputs a security parameter 1^k. Then $\mathcal{C}$ publishes the public parameters *Para* for the Adversary $\mathcal{A}$ and remains the master secret key *msk* secretly for himself.

Queries : the adversary $\mathcal{A}$ can present two types of adaptive queries to the challenger $\mathcal{C}$, as given below.

Private Key Queries: After receiving the request for the private key with user's identity ID_i, $\mathcal{C}$ computes a private key sk_{ID_i} by performing *PKGen* as a response to $\mathcal{A}$.

SigGen Queries: After receiving the request from $\mathcal{A}$ for the signatures of a file M' under the identity ID_i, $\mathcal{C}$ generates the private key sk_{ID_i} by performing *PKGen* algorithm and computes the signatures set by performing *SigGen* algorithm. Finally, $\mathcal{C}$ returns the set of signatures to $\mathcal{A}$.

Audit: In order to audit the data integrity, the challenger $\mathcal{C}$ adaptively challenges to the adversary $\mathcal{A}$. The Adversary $\mathcal{A}$ plays the role of a prover, who is in charge of responding to the integrity challenges from the challenger.

a) The challenger $\mathcal{C}$ generates an auditing challenge *chal* and requests $\mathcal{A}$ with *chal*.
b) After receiving the challenge *chal* from $\mathcal{C}$, $\mathcal{A}$ responds with a data integrity proof P according to knowledge it obtains in the *Queries* phase.
c) The challenger $\mathcal{C}$ performs *ProofGen* algorithm to verify the auditing *Proof* and returns the result to $\mathcal{A}$.

Note that the challenger $\mathcal{C}$ holds the blinded file M' and never send it to the adversary $\mathcal{A}$ in the security model. That implies the proof generated by $\mathcal{A}$ is invalid. If the proof P forged by $\mathcal{A}$ can pass the challenger's verification with a non-negligible probability, it can be concluded that the adversary $\mathcal{A}$ succeeds in the game.

Definition 3: Our scheme is secure against the semi-trusted cloud server if any PPT adversaries $\mathcal{A}$ can win the above game with a negligible probability in κ. That can be denoted as follow:

$$Pr[\mathcal{A}_{win}] \leq \epsilon(\kappa)$$

where κ is taken over all coin tosses between the adversary and challenger.

Malicious manager preventing and sensitive information hiding are also essential properties for our scheme. Thus, we give the security definitions as follows.

Definition 4: Our scheme is secure against the malicious manager if any invalid file M and corresponding signatures S, that have been tampered with or forged by the manager, can be verified by the cloud server with a negligible probability in κ. That is denoted as follow:

$$Pr[Verify(M, S) = Pass] \leq \epsilon(\kappa)$$

where κ is taken over all coin tosses between the manager and cloud.

Definition 5: Our scheme defends sensitive information if any PPT cloud server or researchers can crack the blinded file M', obtaining the file $\overline{M}$, which is equal to the original M, within a negligible probability ϵ. That is denoted as follow:

$$Pr[Crack(M') = M] \leq \epsilon$$

IV. CONSTRUCTION

In this section, We demonstrate a concrete construction of our scheme in detail. For preventing a potential malicious

manger, we define that the user possesses the unique signing right to the files he owns and the manager, whose duties are similar with a gateway in the network, only has the right to inspect the content of a file for the subsequent decision. That implies that any malicious modifications or forgery operations by the manager can be revealed in the procedures of the cloud's signature verification and the TAP 's auditing. We also construct a novel system model by adjusting the duties of each entity for supporting high concurrency and massive data in the real scenario. The time-consuming operations, such as signature transformation and hiding sensitive information about the organization, are mapped to the subordinate users. That makes the appearing frequency of the computational bottleneck at the manager side lower. And, with the user having the ability to download wanted files from the cloud directly, the appearing frequency of the communication bottleneck at the manager side is lower.

In our scheme, we use a portion of the user's private key to hide sensitive information instead of choosing a random value for each file. It is the new mechanism that reduces the storage cost at the user side. It also avoids the inability to recover blind files the user owns due to the unexpected loss of random values.

As the foundation of the auditing scheme, the signature algorithm affects the efficiency of data processing and integrity auditing. In view of this point, we employ Hess's efficient identity-based signature scheme [39] to sign data blocks. Inspired by Shacham and Waters scheme [40], we also fragment a file into blocks with the same size.

The details of this scheme are as follow.

A. SYSTEM SETUP: Setup(1^k)

Given a security parameter k, the setup step is typically performed once for a system as given below.

1) The PKG randomly chooses two multiplicative cyclic groups G_1 and G_T with the same prime order $p > 2^k$ and the corresponding bilinear map e: $G_1 \times G_1 \to G_T$.
2) The PKG selects an element $x \xleftarrow{R} Z_p^*$ and two group elements $g, u \xleftarrow{R} G_1$. Then the PKG computes $P_{pub} = g^x$ as the publice key and holds the master secret key $msk = x$ secretly.
3) The PKG chooses four hash functions H,H_0,H_1: $\{0,1\}^* \to G_1$ and $H_s, H_2 : \{0,1\}^* \to Z_p^*$.
4) Finally, the PKG unveils the public parameters $Para = (G_1, G_2, p, g, u, H, H_0, H_1, H_s, H_2, P_{pub})$ in the system.

B. PRIVATE KEY EXTRACTION: PKGen(Para, msk, ID)

Each users in the system firstly sends an identity to the PKG for obtaining a private key. The details are as follows.

1) When a user requests for the private key under his identity ID, the PKG generates a corresponding private key, which is $sk_{ID} = (sk'_{ID}, sk''_{ID}) = (H_0(ID \parallel 0)^x, H_1(ID \parallel 1)^x)$.
2) The user can validate the received private key sk_{ID} by computing whether

$$e(sk'_{ID}, g) \stackrel{?}{=} e(H_0(ID \parallel 0), P_{pub}) \quad (1)$$

$$e(sk''_{ID}, g) \stackrel{?}{=} e(H_1(ID \parallel 1), P_{pub}) \quad (2)$$

If above equations hold, the user accepts the private key sk_{ID}; otherwise, he rejects it and requests for a new sk_{ID} again.

C. SIGNATURE GENERATION: SigGen(Para, M, sk_{ID}, Mname)

The user firstly fragments a file M, which is to be submitted, into n-blocks, getting $M = m_1, m_2, \cdots, m_n$ where $m_i \in Z_p^*$. We define that SIS is a set of indexes for the data blocks that contain the sensitive information. The user signs each data block m_i as follows.

1) The user calculates a blinding factor with a portion of his private key, $a_i = H(Mname \parallel i)^{H_s(sk''_{ID})}$, where $i \in SIS$ and $Mname \xleftarrow{R} Z_p^*$ is a unique identification of M.
2) In order to hide sensitive information, for each block m_i whose index i is in the set SIS, the user calculates the blinded blocks $m'_i = m_i + H_2(a_i) \ mod \ p$. We give an instance of data blinding as shown in Fig. 3. It can be seen that, after blinding all above data block m_i, the user obtains the blinded file $M' = (m'_1, m'_2, ..., m'_n)$, where if $i \in SIS$, $m'_i \neq m_i$; otherwise $m'_i = m_i$.

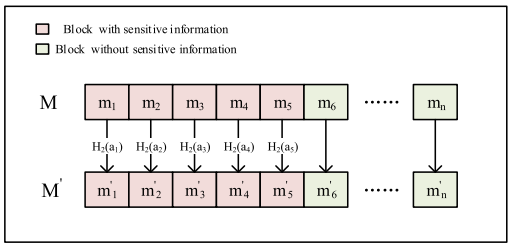


FIGURE 3. An instance of data blinding.

3) The user chooses an element $r \xleftarrow{R} Z_p^*$ and sets $R = g^r$, where R is used to verify signatures in the later algorithm. Then, the user sets the meta tag $\tau_0 = Mname \parallel n \parallel R$ and computes a file tag $\tau = \tau_0 \parallel IDS(\tau_0, ssk) \parallel spk$ for the blinded file M', where $IDS(\tau_0, ssk)$ is an identity-based signature upon the value τ_0, the sign secret key ssk is hold by user secretly and spk is the sign public key.
4) The user proceeds to generate the signature for each blocks in the blinded file M' by another portion of his private key sk'_{ID} as follow.

$$\sigma_i = H_0(ID \parallel 0)^x (H(Mname \parallel i) \cdot u^{m'_i})^r \quad (3)$$

And, the user collects all signatures into a set $\Phi = \{\sigma_i\}_{1 \le i \le n}$ for the blinded file M'. Finally, $\{M', \Phi, \tau\}$ is submitted to a manager of the organization.

D. FILE CHECK: Check(Para, M′, Φ, τ)

After receiving a file M' and necessary values submited by a user, the manager performs this algorithm as follow.

1) The manager obtains the meta tag τ_0 of M' and the sign public key spk by spliting τ. Then the manager validates τ_0 by performing IDV $(\tau_0, spk, IDS(\tau_0, ssk))$. If it passes the validation, the manager obtains the verification value R by spliting τ_0 and does the next step.
2) The manager validates each signature σ_i in the set Φ by computing whether:

$$e(\sigma_i, g) \stackrel{?}{=} e(H_0(ID \parallel 0), P_{pub}) \cdot e(H(Mname \parallel i) \cdot u^{m'_i}, R) \quad (4)$$

if the (4) does not hold, the manager rejects this file submission due to the invalid signature; otherwise, does the next step.

c) Finally, the manager inspects the content of data blocks with sensitive information. If the block m'_i in the blinded file M' is found to still contain sensitive information, the manager rejects this file submission; otherwise, the manager uploads $\{M', \Phi, \tau\}$ to the cloud. The cloud will validate the file M' and the corresponding signatures to prevent the malicious modification by the manager.

E. PROOF GENERATION: PROOFGEN(PARA, M′,Φ′,CHAL)

The procedure of proof generation consists of two phases: challenge and response. The details are given as follow.

1) **Challenge**. Before challenging a blinded file M', the TPA askes the cloud for the file tag τ and validates the meta tag τ_0 in it by the same way in *Check* algorithm. If τ_0 cannot pass the validation, the TPA reports an error exception to the file owner; otherwise, obtains the file name $Mname$, the verification value R by spliting τ_0 and then generates challenge $chal$ to audit as follows:
   a) generates a set of random indexes I, where each element of I is in $[1, n]$ and its length is c.
   b) chooses an element $v_i \stackrel{R}{\leftarrow} Z_p^*$ for each element i in the set I.
   c) requests the cloud with a challenge $chal = \{i, v_i\}$, where $i \in I$.
2) **Response.** When the TPA requests with a challenge, the cloud reads the corresponding file M', signatures set Φ it stores. Then the cloud generates a proof $P = \{\lambda, \sigma\}$ according to those resources, where $\lambda = \sum_{i\in I} m'_i v_i$ is a linear combination of data blocks, $\sigma = \prod_{i\in I} \sigma_i^{v_i}$ is an aggregate signature.

Finally, the cloud responds the TPA with the proof P.

F. PROOF VERIFICATION: PROOFVERIFY(CHAL, PARA, P)

The TPA validates the proof from the cloud by computing whether:

$$e(\sigma, g) \stackrel{?}{=} e(H_0(ID \parallel 0), P_{pub})^{\sum_{i\in I} v_i} \cdot Pe(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda}, R) \quad (5)$$

If (5) holds, the TPA reports the blinded file M' is intact to the user; otherwise, the TPA reports the file is not intact.

V. SECURITY ANALYSIS

In this section, we demonstrate that our scheme implement the properties of the completeness, auditing soundness, malicious manager preventing and sensitive information hiding. Besides, A detailed probabilistic analysis is given to ensure that our scheme achieves the detection of the cloud server's misbehavior.

A. COMPLETENESS

Completeness guarantees the correctness of our scheme if all the entities behave honestly.

Theorem 1: On the condition that the PKG performs the algorithm *PKGen* successfully, the user always accepts the private key generated by the PKG. The manager always accepts the blinded file and its corresponding signatures if they are computed by the user faultlessly. While the blinded file keeps intact, the proof computed by the cloud server can pass the validation of the TPA.

Proof: if (1), (2), (4), (5) all hold, then this theorem is proved. Since $sk'_{ID} = H_0(ID \parallel 0)^x$, it follows that,

$$\begin{aligned} e(sk'_{ID}, g) &= e(H_0(ID \parallel 0), g) \\ &= e(H_0(ID \parallel 0), P_{pub}) \end{aligned}$$

Hence, (1) holds. And the proof of (2) is same as (1). So we can say (2) also holds.

Since the signature of data block i is

$$\sigma_i = H_0(ID \parallel 0)^x (H(Mname \parallel i) \cdot u^{m'_i})^r$$

it follows that,

$$\begin{aligned} e(\sigma_i, g) &= e(H_0(ID \parallel 0)^x (H(Mname \parallel i) \cdot u^{m'_i})^r, g) \\ &= e(H_0(ID \parallel 0)^x, g) \cdot e((H(Mname \parallel i) \cdot u^{m'_i})^r, g) \\ &= e(H_0(ID \parallel 0), P_{pub}) \cdot e(H(Mname \parallel i) \cdot u^{m'_i}, R) \end{aligned}$$

Therefore, (4) holds.

Note that $\lambda = \sum_{i\in I} m'_i v_i$ and

$$\begin{aligned} \sigma &= \prod_{i\in I} \sigma_i^{v_i} = \prod_{i\in I} (H_0(ID \parallel 0)^x (H(Mname \parallel i) \cdot u^{m'_i})^r)^{v_i} \\ &= \prod_{i\in I} H_0(ID \parallel 0)^{x \cdot v_i} \cdot \prod_{i\in I} (H(Mname \parallel i) \cdot u^{m'_i})^{r \cdot v_i} \\ &= H_0(ID \parallel 0)^{x \cdot \sum_{i\in I} v_i} \cdot (\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\sum_{i\in I} m'_i v_i})^r \\ &= H_0(ID \parallel 0)^{x \cdot \sum_{i\in I} v_i} \cdot (\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda})^r \end{aligned}$$

So we have,

$$\begin{aligned} e(\sigma, g) &= e(H_0(ID \parallel 0)^{x \cdot \sum_{i\in I} v_i}, g) \\ &\quad \cdot e((\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda})^r, g) \end{aligned}$$

$$= e(H_0(ID \parallel 0)^{\sum_{i\in I} v_i},\ P_{pub}) \cdot e(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda},\ R)$$

Hence, (5) holds.

B. SOUNDNESS

Auditing soundness shows that our scheme is secure against a semi-trusted cloud server. The following theorem guarantees the soundness of our scheme as defined.

Theorem 2: If the Computational Diffie-Hellman problem and Discrete-Logarithm problem are hard in bilinear groups and the signature mechanism(*IDS*) for file signatures existentially is unforgeable, then, in the random oracle model, except with negligible probability no adversary or semi-trusted server, who is against the soundness of our scheme, ever causes the TPA to accept the proof, except by keeping intact data and responding with correct proof.

Proof: We proceed to prove the theorem in a series of security games, which is inspired by the scheme [40].

Game 0: This game's process is simply the security model in Section III.

Game 1: This game is same as Game 0, except with following difference. The challenger $\mathcal{C}$ maintains a list which records the adversary's responses in the Queries phase. In the Audit phase, The challenger $\mathcal{C}$ inspects each instance of the interactions with the adversary $\mathcal{A}$. If in any of these instances, $\mathcal{A}$ succeeds but the aggregate signature σ is not equal to $\prod_{i\in I} \sigma_i^{v_i}$ generated by $\mathcal{C}$ according to its maintained file, the challenger declares failure and aborts the game.

Analysis: Suppose that the difference in the adversary's success probabilities between Game 0 and Game 1 is non-negligible. We now demonstrate the method to biuld a simulator to solve the CDH problem. Given $g, g^{\alpha}, h \in G$ as input, where the multiplicative group $G = \langle g \rangle$ with the prime order p, this simulator intends to output h^{a}. To solve the hard problem, it interacts with the adversary $\mathcal{A}$, simulating the behavior of the challenger in Game 0, except with the differences as shown below:

- The simulator selects an element $x \overset{R}{\leftarrow} Z_p^*$ and two random values η, ξ from Z_p^*. Then, it calculates $P_{pub} = g^x$, $msk = x$, and $u = h^{\eta} g^{\xi}$.
- When asked for some signatures of data blocks, the simulator acts as follows. Firstly, it runs algorithm *PKGen* to extract a private key $sk_{ID} = (sk'_{ID},\ sk''_{ID}) = (H_0(ID \parallel 0)^x,\ H_1(ID \parallel 1)^x)$ for the user with the identity *ID*. Then, it randomly chooses a name *Mname* and an element $\omega \in Z_p^*$ for the file M'. The simulator set $r = \alpha\omega$ and the verification value $g^r = (g^{\alpha})^{\omega}$. To keeps a list of qureies and responses in a consistant way, it specifies the random oracle H as shown below.
- For each data block $m_i \in Z_P^* (i \in [1, n])$ queried by $\mathcal{A}$, the simulator selects a random value $r_i \in Z_P^*$ and specifies the random oracle H under i as

$$H(Mname \parallel i) = g^{r_i} / (h^{\eta m'_i} \cdot g^{\xi m'_i})$$

Then the simulator calculates

$$(H(Mname \parallel i) \cdot u^{m'_i})^r = (g^{r_i} / (h^{am'_i} \cdot g^{bm'_i}) \cdot (h^{\eta} g^{\xi})^{m'_i})^r = (g^{r_i})^r = R^{r_i}$$

where $R = g^r$. The simulator proceeds to sign the data block m'_i as

$$\sigma_i = H_0(ID \parallel 0)^x (H(Mname \parallel i) \cdot u^{m'_i})^r = H_0(ID \parallel 0)^x R^{r_i}$$

- The simulator interacts with the adversary $\mathcal{A}$, running *ProofGen* and *ProofVerify* algorithm. If the condition mentioned in the Game 1 occurs: the adversary $\mathcal{A}$ succeeds, while the aggregate signature σ is not equal to $\prod_{i\in I} \sigma_i^{v_i}$ generated by $\mathcal{C}$ from the file it holds, the simulator aborts this game and begins to solve the hard problem by using the records of the adversary's qureies.

Before showing how the simulator solves the CDH problem, we will deduce a few conclusions. Suppose that an honest cloud returns a correct proof $P = \{\lambda, \sigma\}$, which satisfies the verification equation that is

$$e(\sigma, g) = e(H_0(ID \parallel 0), P_{pub})^{\sum_{i\in I} v_i} \cdot e(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda}, R) \quad (6)$$

Suppose that the adversary $\mathcal{A}$ returns a forge proof $\overline{P} = \{\overline{\lambda}, \overline{\sigma}\}$ in Game 1. we know that the simulator aborted, which implies that the aggregate signature $\sigma \neq \overline{\sigma}$ and that $\overline{\sigma}$ can pass the verification equation that is

$$e(\overline{\sigma}, g) = e(H_0(ID \parallel 0), P_{pub})^{\sum_{i\in I} v_i} \cdot e(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\overline{\lambda}}, R) \quad (7)$$

where $R = g^r$. Assume that if $\overline{\lambda} = \lambda$, we can get $\overline{\sigma} = \sigma$, which contradicts the assumption in Game 1. Hence, we can draw the conclusions: it must be the case that $\overline{\sigma} \neq \sigma$ and $\Delta\lambda = \overline{\lambda} - \lambda$ is non-zero.

With this in mind, now, dividing (6) by (7), we can get

$$e(\overline{\sigma}/\sigma, g) = e(u^{\Delta\lambda}, R) = e((h^{\eta} g^{\xi})^{\Delta\lambda}, \quad ((g^{\alpha})^{\omega})).$$

Rearranging terms yields

$$e(h^{\alpha}, g)^{\eta\omega\Delta\lambda} = e(\overline{\sigma}\sigma^{-1}(g^{\alpha})^{-\xi\omega\Delta\lambda}, g).$$

It can be seen that we have obtained the solution to the CDH problem,

$$h^{\alpha} = (\overline{\sigma}\sigma^{-1}(g^{\alpha})^{-\xi\omega\Delta\lambda})^{\frac{1}{\Delta\lambda\eta\omega}},$$

as long as $\Delta\lambda\eta\omega \neq 0 \ mod \ p$. However, we have drawn a conclusion that $\Delta\lambda$ can not be zero, which deduces the probability that $\Delta\lambda\eta\omega = 0 \ mod \ p$ equals to $1/p$.

Therefore, on the condition that the difference in the adversary's success probabilities between Game 0 and Game 1 is non-negligible, a simulator can be built to solve the CDH problem by interacting with $\mathcal{A}$.

Game 2: This game is same as Game 1, except with following difference. The challenger $\mathcal{C}$ also maintains a list which records the adversary's responses in the Queries phase. In the Audit phase, The challenger $\mathcal{C}$ inspects each instance of the interactions with the adversary $\mathcal{A}$. If in any of these instances, $\mathcal{A}$ succeeds but the linear combination of data blocks λ is not equal to $\sum_{i\in I} m_i' v_i$ generated by $\mathcal{C}$ from the file it holds, the challenger declares failure and aborts the game.

Analysis: Suppose that the difference in the adversary's success probabilities between Game 1 and Game 2 is non-negligible. We now demonstrate the method to biuld a simulator to solve the DL problem. Given g, g^{α} as input, where the multiplicative group $G = \langle g \rangle$ with the prime order p, this simulator intends to output a. To solve the hard problem, it interacts with the adversary $\mathcal{A}$, simulating the behavior of the challenger in Game 1, except with the differences as shown below:

- The simulator selects an element $x \overset{R}{\leftarrow} Z_p^*$ and two random values η, ξ from Z_p^*. Then, it calculates $P_{pub} = g^x$, $msk = x$, and $u = (g^{\alpha})^{\eta} g^{\xi}$.
- The simulator interacts with the adversary $\mathcal{A}$, running algorithms *ProofGen* and *ProofVerify*. If the condition mentioned in the Game 2 occurs: the adversary $\mathcal{A}$ succeeds, while the linear combination of data blocks λ is not equal to $\sum_{i\in I} m_i' v_i$ generated by $\mathcal{C}$ from the file it holds, the simulator aborts this game and begins to solve the hard problem by using the records of the adversary's qureies.

Again, suppose that an honest cloud returns a correct proof $P = \{\lambda, \sigma\}$, which satisfies (6). And we know that the simulator aborted in Game 2, which implies that the forge proof $\overline{P} = \{\overline{\lambda}, \overline{\sigma}\}$ can pass the verification (7) and $\Delta\lambda = \overline{\lambda} - \lambda$ is not 0. From the difference between Game 1 and Game 2, we can deduce that $\overline{\sigma} = \sigma$; otherwise, the adversary cannot win in Game 2. By checking (6) and (7), we have

$$\begin{aligned} e(\sigma, g) &= e(\overline{\sigma}, g) \\ &\rightarrow e(H_0(ID \parallel 0), P_{pub})^{\sum_{i\in I} v_i} \\ &\quad \cdot e(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\lambda}, R) \\ &= e(H_0(ID \parallel 0), P_{pub})^{\sum_{i\in I} v_i} \\ &\quad \cdot e(\prod_{i\in I} H(Mname \parallel i)^{v_i} \cdot u^{\overline{\lambda}}, R) \end{aligned}$$

Hence, we get

$$u^{\lambda} = u^{\overline{\lambda}}$$

Rearranging terms yields

$$u^{\Delta\lambda} = ((g^{\alpha})^{\eta} g^{\xi})^{\Delta\lambda} = (g^{\alpha})^{\eta\Delta\lambda} \cdot g^{\xi\Delta\lambda} = 1$$

It can be seen that we have obtained the solution to the DL problem,

$$\alpha = -\xi\eta^{-1} \ mod \ p$$

as long as $\eta \neq 0 \ mod \ p$. Since η is a random value, we know the probability that $\eta = 0 \ mod \ p$ equals to $1/p$.

Therefore, on the condition that the difference in the adversary's success probabilities between Game 1 and Game 2 is non-negligible, a simulator can be built to solve the DL problem by interacting with $\mathcal{A}$.

Wrapping Up: Since the hardness of the CDH problem implies the hardness of the discrete logarithm problem [40], we accomplish proving the auditing soundness of our scheme.

C. MALICIOUS MANAGER PREVENTING

We define the following theorem to guarantee that our scheme can prevent the malicious managers.

Theorem 3: If the Computational Diffie-Hellman problem and Discrete-Logarithm problem are hard in bilinear groups, then, in the random oracle model, except with negligible probability no malicious manager, ever causes the cloud to accept the file, except by uploading the file that has not been maliciously modified and correct signatures.

Proof: The proof process of theorem 3 is similar with theorem 2. So we can say our scheme achieves the property that prevents the malicious manager.

D. SENSITIVE INFORMATION HIDING

The following theorem is defined to guarantee that it is infeasible that the cloud server and researchers obtain sensitive information.

Theorem 4: In the random oracle model, no researchers or cloud server ever recovers original data with sensitive information except with a negligible probability.

Analysis: According to the construction of the proposed scheme in Section IV, for hiding sensitive information, the user with identity ID_u calculates a blinded data m_i' for the original block m_i as follow.

$$m_i' = m_i + H_2(H(Mname \parallel i)^{H_s(sk''_{ID_u})}) \ mod \ p \tag{8}$$

If Given the m_i', the adversary $\mathcal{A}$ recovers the m_i, we declare the $\mathcal{A}$ succeeds. By analyzing the (8), three conditions that are needed for the adversary $\mathcal{A}$ to succeed are summarized as follows:

- $\mathcal{C}_1$: $\mathcal{A}$ generates the private key sk''_{ID_u} corresponding to the identity ID_u by querying the challenger $\mathcal{C}$.
- $\mathcal{C}_2$: $\mathcal{A}$ obtains the private key sk''_{ID_u} by a series of queries for the random oracle H_s.
- $\mathcal{C}_3$: $\mathcal{A}$ obtains the private key sk''_{ID_u} by a series of queries for the random oracle H_2.

If any one condition holds, $\mathcal{A}$ succeeds. Since $\mathcal{C}_1$, $\mathcal{C}_2$, $\mathcal{C}_3$ are independent events, the probability $Pr[\mathcal{A}_{succeeds}]$ is

$$\begin{aligned} Pr[\mathcal{A}_{succeeds}] &= Pr[\mathcal{C}_1] + Pr[\mathcal{C}_2] + Pr[\mathcal{C}_3] \\ &\quad - Pr[\mathcal{C}_1]Pr[\mathcal{C}_2] - Pr[\mathcal{C}_1]Pr[\mathcal{C}_3] \\ &\quad - Pr[\mathcal{C}_2]Pr[\mathcal{C}_3] + Pr[\mathcal{C}_1]Pr[\mathcal{C}_2]Pr[\mathcal{C}_3] \end{aligned}$$

Proof: We now calculate the probability $Pr[\mathcal{A}_{succeeds}]$, proving Theorem 4.

Claim 1: If CDH problem is hard in the bilinear groups, in the random oracle model, no PPT $\mathcal{A}$ ever generates the private key sk''_{ID_u} corresponding to the identity ID_u by querying the challenger $\mathcal{C}$ except with a negligible probability $Pr[\mathcal{C}_1] \leq \epsilon'$.

Proof: To prove Claim 1, we define the following game between the adversary $\mathcal{A}$ and the challenger $\mathcal{C}$.

- **Setup.** the challenger $\mathcal{C}$ runs algorithm *Setup*, publishing the public key P_{pub} for $\mathcal{A}$ and remaining the master secret key msk.
- **Queries.** When the adversary $\mathcal{A}$ askes for private keys on at most q_p identities of his choice $\{ID_1, ID_2, \cdots, ID_{q_p}\}$, $\mathcal{C}$ generates a private key $sk''_{ID_i} = H_1(ID_i \parallel 1)^{msk}$ for each request.
- **Output.** the adversary $\mathcal{A}$ generates the private key sk''_{ID_u} for the identity ID_u, where $ID_u \notin \{ID_1, ID_2, \cdots, ID_{q_p}\}$, and wins the game if the following equation holds.

$$e(sk''_{ID_u}, g) = e(H_1(ID_u \parallel 1), P_{pub}) \tag{9}$$

Suppose that $\mathcal{A}$ wins the above game with a non-negligible probability. We now demonstrate how to biuld a PPT simulator $\mathcal{S}$ to solve the CDH problem. Given $g, g^{\alpha}, h \in G$, where the multiplicative group $G = \langle g \rangle$ with the prime order p, the simulator $\mathcal{S}$ intends to generate h^a. To solve the hard problem, $\mathcal{S}$ interacts with $\mathcal{A}$ as follows:

- **Setup.** The simulator S selects an element $x \overset{R}{\leftarrow} Z_p^*$, and sets $msk = x$ and $P_{pub} = g^{\alpha} g^{x}$. And $\mathcal{S}$ publishes the public key P_{pub}.
- **Hash queries.** The adversary $\mathcal{A}$ can adaptively query the random oracle H_1. The simulate holds a list *QueryList* to record the instance $\langle ID_i, w_i, r_i \rangle$ for each query. When $\mathcal{A}$ queries the random oracle H_1 for the identity $ID_i \in \{0, 1\}^*$, $\mathcal{S}$ returns the $H_1(ID_i \parallel 1) = w_i \in G$ if ID_i is already on the *QueriesList*; otherwise, $\mathcal{S}$ generates a hash value as follow.
  a) The simulator $\mathcal{S}$ selects an element $r_i \overset{R}{\leftarrow} Z_p^*$.
  b) If the $ID_i = ID_u$, $\mathcal{S}$ sets $w_i = hg^{r_i}$; otherwise, $\mathcal{S}$ sets $w_i = g^{r_i}$.
  c) The simulator $\mathcal{S}$ inserts the instance $\langle ID_i, w_i, r_i \rangle$ to the *QueriesList* and returns the hash value:

$$H_1(ID_i \parallel 1) = w_i \in G.$$

- **Private key queries.** After receiving a private key query for the identity ID_i, the simulator generates a private as follows.
  a) The simulator $\mathcal{S}$ obtains the $H_1(ID_i \parallel 1) = w_i \in G$ by performing the algorithm of *Hash queries*.
  b) If ID_i=ID_u, $\mathcal{S}$ declares failure and aborts the game; otherwise, $\mathcal{S}$ returns a valid private key $sk''_{ID_i} = {w_i}^x$.
- **Output.** Finally, the adversary $\mathcal{A}$ generates the private key sk''_{ID_u} for the identity ID_u. If sk''_{ID_u} does not satisfies (9), $\mathcal{S}$ declares failure and aborts the game; otherwise, $\mathcal{S}$ computes the solution for the CDH problem as follow.

Since $H_1(ID_u \parallel 1) = w_i = hg^r$ and $P_{pub} = g^{\alpha} g^{x}$, where r is a random value in Z_p^*, the simulator $\mathcal{S}$ gets

$$\begin{aligned} & e(sk''_{ID_u}, g) = e(hg^r, g^{\alpha} g^{x}) \\ \rightarrow\ & e(sk''_{ID_u}, g) = e(h^{\alpha} h^{x}, g) e(g^{\alpha} g^{r+x}, g) \\ \rightarrow\ & e(sk''_{ID_u} (g^{\alpha} g^{r+x})^{-1}, g) = e(h^{\alpha} h^{x}, g) \end{aligned}$$

Thus, the simulator obtains the solution for the CDH problem,

$$h^{\alpha} = sk''_{ID_u} (h^{x} g^{\alpha} g^{r+x})^{-1}$$

Therefore, one the condition that the adversary $\mathcal{A}$ succeeds in the game, a simulator can be constructed to solve the CDH problem with a non-negligible probability. Boneh *et al.* [44] analyzed the probability that the simulator solves the hard problem in detail. Claim 1 is proved.

According to the Claim 1, we get $Pr[\mathcal{C}_1] \leq \epsilon'$. In the random oracle model, the outputs of the hash functions H_2, H_s are uniformly distributed. Hence, $Pr[\mathcal{C}_2] = Pr[\mathcal{C}_3] = 1/p$. We get

$$Pr[\mathcal{A}_{succeeds}] \leq \epsilon' + \frac{2}{p} - \frac{2\epsilon}{p} - \frac{1}{p^2} + \frac{\epsilon}{p^2}$$

Since p is a large prime number, the probability $Pr[\mathcal{A}_{succeeds}]$ that $\mathcal{A}$ recovers the original file with sensitive information is negligible. Theorem 4 is proved.

E. PROBABILISTIC ANALYSIS

As mentioned in our scheme, the TPA only verifies random blocks instead of the entire file, still achieving the detection of the cloud server's misbehavior. We now analyze the detection probability, with the same strategy in previous work [7]. Assume that the cloud server modifies t blocks out of a n-block file and the number of different blocks challenged by the TPA is c. Set X to be a discrete random variable, which is the intersection of the set of blocks challenge by the TPA and the set of blocks deleted by the cloud server. Let P_X be the probability that the TPA can detect the cloud server's deletion operation. Thus, we have:

$$\begin{aligned} P_X &= P\{X \geq 1\} = 1 - P\{X = 0\} \\ &= 1 - \frac{n-t}{n} \cdot \frac{n-1-t}{n-1} \cdot \frac{n-2-t}{n-2} \cdot \cdots \cdot \frac{n-c+1-t}{n-c+1} \end{aligned}$$

Since $\frac{n-i-1-t}{n-i-1} \leq \frac{n-i-t}{n-1}$, so we get:

$$1 - (\frac{n-t}{n})^c \leq P_X \leq 1 - (\frac{n-c+1-t}{n-c+1})^c$$

Therefore, the TPA can detect the cloud server's misbehavior with probability at least $1 - (\frac{n-t}{n})^c$. That implies that the probability increases as the number of data blocks challenged by the TPA increases.

VI. THEORETICAL ANALYSIS

A. FUNCTIONAL ANALYSIS

We show the functional comparison between our scheme and some related work [6], [15], [40]–[43]. As illustrated in

TABLE 1. Functional comparison.

Scheme	Malicious managers preventing	Sensitive information hiding	Data sharing	Direct data retrieval	Certificate freeness [41]	Public auditing
Shacham and Waters. [40]	×	×	×	—	×	√
Wang et al. [42]	×	×	×	—	×	×
Wang et al. [15]	×	×	√	√	×	√
Wang et al. [41]	×	×	×	—	√	√
Luo et al. [43]	×	×	√	√	√	√
Shen et al. [6]	×	√	√	√	×	√
Ours	√	√	√	√	√	√

TABLE 2. Comparisons on computation costs of each process with Shen's scheme.

Process	Our scheme	Shen's scheme
Data blinding	$(d_1+d_2)(H_{G_1}+2H_{Z_p^*}+A_{Z_p^*})$	$d_1(PF_{Z_p^*}+A_{Z_p^*})$
SigGen	$n(H_{G_1}+M_{G_1}+2E_{G_1})$	$n(H_{G_1}+2M_{G_1}+(l+2)E_{G_1})$
Check(sanitization)	—	$(d_1+d_2)(E_{G_1}+M_{G_1}+S_{Z_p^*})$
ProofGen	$(c-1)(M_{G_1}+A_{Z_p^*})+cE_{G_1}+cM_{Z_p^*}$	$(c-1)(M_{G_1}+A_{Z_p^*})+cE_{G_1}+cM_{Z_p^*}$
ProofVerify	$3P+M_{G_2}+(c-1)A_{Z_p^*}+E_{G_2}$ $+(c+1)E_{G_1}+cM_{G_1}+cH_{G_1}$	$4P+2M_{G_2}+2(c-1)A_{Z_p^*}+2E_{G_2}+$ $(l+c+1)E_{G_1}+(c+l)M_{G1}+cH_{G_1}$

TABLE 1, only our scheme and Shen's scheme [6] support sensitive information hiding in the field of data sharing. Furthermore, our scheme satisfies malicious manager preventing and direct data retrieval, which are absent from Shen's scheme. Also, our scheme supports the remaining properties in TABLE 1: data sharing, certificate freeness, and public auditing.

B. PERFORMANCE ANALYSIS

TABLE 2 illustrates the computation costs of our scheme and the comparison with Shen's scheme [6] in each process. In the table, H_{G_1}, M_{G_1} and E_{G_1} indicate the costs of performing one hasing, one multiplication and one exponentiation in G_1; $H_{Z_P^*}$, $A_{Z_P^*}$, $S_{Z_P^*}$ and $M+Z_P^*$ indicate the costs of performing one hasing, one addition, one subtraction and one multiplication in Z_p^*; And $PF_{Z_p^*}$ denotes the cost of generating a random value through one pseudo-random function; Similarly, E_{G_2} and M_{G_2} indicate the costs of performing one exponentiation and one multiplication in G_2; P denotes the cost of one bilinear pairing: $G_1 \times G_1 \rightarrow G_2$; n is the number of the whole file; c is the number of random blocks challenged by the TPA; d_1 is the the number of data blocks with the sensitive information about individuals, and d_2 is the sensitive information about the organization; l is the length of a user's identity in Shen's scheme [6].

From TABLE 2, it can seen that data blinding brings $(d_1+d_2)(H_{G_1}+2H_{Z_p^*}+A_{Z_p^*})$ cost to users in our scheme, and Shen's scheme is $d_1(PF_{Z_p^*}+A_{Z_p^*})$. That results from distributing the manager's computational costs to subordinate users. Consequently, combinating the process *check* (*sanitization* in Shen's scheme), Shen's scheme takes $d_1(PF_{Z_p^*}+A_{Z_p^*})+(d_1+d_2)(E_{G_1}+M_{G_1}+S_{Z_p^*})$. However, in the check process, our scheme does not take time-consuming operation, like exponentiation, and only inspects the content of data blocks. Therefore, our scheme provides a more secure and efficient mechanism for sensitive information hiding. As for auditing-realted processes, TABLE 2 illustrates that our scheme takes fewer pairings and exponentiations. Our scheme's costs of *SigGen* is $n(H_{G_1}+M_{G_1}+2E_{G_1})$, that is n multiplications and nl exponentiations fewer than the costs of Shen's scheme. Both schemes have a same cost in the process *ProofGen*. In *ProofVerify*, our scheme brings $3P+M_{G_2}+(c-1)A_{Z_p^*}+E_{G_2}+(c+1)E_{G_1}+cM_{G_1}+cH_{G_1}$ cost to the TPA, and Shen's scheme brings one pairing, $(c-1)$ additions, $(l+1)$multiplications and $(l+1)$ exponentiations more. It is widely accepted that bilinear pairing and exponentiation are more time-consuming compared to the other ones. Therefore, it can be concluded that our scheme theoretically enhances the efficiency of integrity auditing.

TABLE 3. Comparison on communication costs and storage overhead with Shen's scheme.

Scheme	Communication costs in auditing	Storage costs at the user side
Shen's scheme	$(c+1)ES_{Z_P^*}+ES_{G_1}$	$ES_{Z_p^*}$
Our scheme	$(c+1)ES_{Z_P^*}+ES_{G_1}$	Zero

Further, on the aspects of communication and storage costs, we compare with Shen's scheme as shown in TABLE 3. In the table, $ES_{Z_p^*}$ and ES_{G_1} are the sizes of an element in Z_P^* and G_1. c is the number of data blocks challenged by the TPA. When performing auditing, two scheme consume the same communication cost. And to recovery an original file,

Shen's scheme make user to keep an element in Z_p^* for the file. Our scheme eliminates such cost at the user side, as well as avoiding the inability to recover the file due to the loss of random values. In addition, our scheme support preventing malicious managers.

VII. EXPERIMENTAL ANALYSIS

We further implement prototype systems of our scheme and Shen's scheme based on Pairing-Based Cryptography (PBC) Library [45]. We program all algorithms of both schemes in C language. The running environment of prototype systems is MacOS Mojave 10.14.3 with Intel Core i5 2.7GHz CPU and 8GB 1867MHz DDR3 memory. In the implementation, we employ the symmetric the elliptic curve $y^2 = x^3 + x$, where ES_{G_1}, the size of an element in G_1, is 512 bits and $ES_{Z_p^*}$, the size of an element in Z_P^*, is 160 bits. And we set the length of user identity used by Shen's scheme to be 160 bits, which is same as their strategy. We perform our experiments on 200 sample EHRs from the Hospital of the University of Electronic Science and Technology of China(UESTC Hospital). All samples are stored in comma-separated values(CSV) format files respectively with an average size of 6KB.

A. SENSITIVE INFORMATION HIDING

Given a smaple EHR as shown in TABLE 4, it can be noted that the sensitive information is from row 1 to row 10 and the information after row 10 is needed to publish to researchers in the UESTC. Since the sensitive information includes the indispensable portions and optional ones and the size of each portions is not fixed, the size of the sensitive information is in the range of 240-300 bytes. To fix the size of it to 300 bytes, we extend the content of sensitive information with wildcard '*'. During the process of *SigGen*, each smaple is fragmented into multiple blocks with the smae szie (20 bytes). That is to say the set $SIS = \{1, 2, 3, 4, \cdots, 15\}$ is fixed.

TABLE 4. A sample EHR: San Zhang.

Hospital	UESTC Hospital
HospitalAddress	No.4 Section 2 North Jianshe Road
Fax	028-83204355
PatientName	San Zhang
PatientAddress	No.4 Section 2 North Jianshe Road
IdNumber	5011028195004041111
HomePhone	111-2222-3333
ExternalId	MR-111-1111
BirthDate	04/04/1950
Gender	Female
Race	black
Problems	DIABETES MELLITUS (ICD-250.)· · · · · ·
Medications	PRINIVIL TABS 20 MG (LISINO· · · · · ·
⋮	⋮
RespRate	16/min
BpSystolic	158 mm Hg

TABLE 5. The result of sensitive information hiding for the smaple of San Zhang.

block1	0x341efd53242693a362ae8db73c167b22fbc1b276
block2	0x7621286e4d50ac9980f3d6fd7b226d1b95c0ad88
block3	0x7a5e478a665bcadd9cf6c90084da5a1eb98c5e68
block4	0x6f584792696a89ab58d2d2ff86162bf9f4b9ac88
block5	0x7259f37b695dcd839ee5dbb941e043dcc38b7045
block6	0x3e28085e046ccaeda1e9d201711c6d1cf0849176
⋮	⋮
block15	0x6e2103592a2c96ad6cb8978d11ae0baf8b583e15
block16	BirthDate,04/04/195
block17	0\ngender, Female\nrace
⋮	⋮
block55	n\NbpSystolic,158 mm
block56	Hg

Thus, the content of the first 15 blocks is ciphertext as shown in TABLE 5, which is the result of processing the sample in TABLE 4. The result shows that the first 15 blocks with sensitive information have been hidden and other blocks can be directly used to analysis.

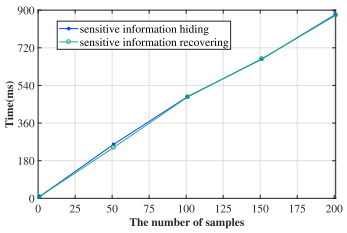


FIGURE 4. The performance of processing sensitive information.

To estimate to performance of the sensitive information hiding and recovering, we record the time cost of processing different numbers of samples EHRs(from 1 to 200) and give the result as illustrated in Fig. 4. It can be noted that the time cost of hiding 60KB sensitive information in 200 samples is 830.3ms and our mechanism hidden data at about 72KB/s on average. The curve of information recovering is almost close to that of information hiding. The reasons is that the i-th blinded block m_i' is $m_i + H_2(H(Mname \parallel i)^{H_5(sk''_{ID_u})}) \ mod \ p$, and the main time cost is to calculate hash values for both information hidding and recovering. That is to say, compared with schemes without sensitive information hiding, the time cost of processing such information is negligible for deploying in the real scenario.

B. PERFORMANCE COMPARISON

The performance of all processes are illustrated in Fig. 5, which result from processing 105 data blocks including five blocks with sensitive information. With simplifying signature algorithm, we reduce the time costs of extracting and verifying the private key to 12 ms and 14 ms, that takes second-level

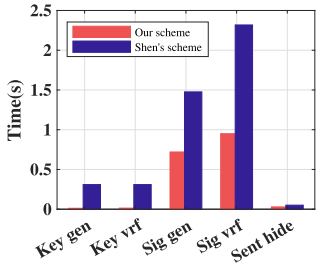


FIGURE 5. The performance of each process.

time overhead in Shen's scheme. Besides, the time costs of generating and verifying signatures are dropped to 0.72s and 0.95s. Compared to Shen's scheme, the efficiencies of *SigGen* and *Sigvry* are almost increased by 50 percent. As for sensitive information hiding, we employ a novel mechanism in our scheme to reduce the storage cost at the user side. Moreover, our scheme costs less time in implementation, as shown in Fig. 5, only taking 29 ms to hide sensitive information. And Shen's scheme takes 41 ms. With Adding up the all time costs in Fig. 5, we can get the time overhead that is taken by the user and manager to upload a 2 KB file to the cloud. Our scheme needs less than 2s. On the other hand, Shen's scheme needs more than 4s.

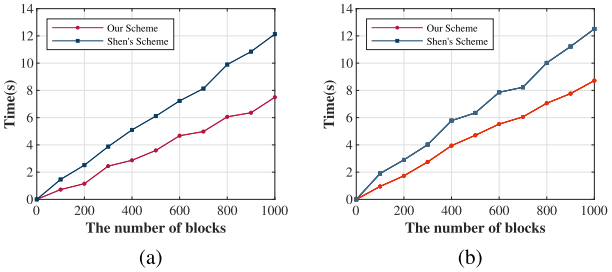


FIGURE 6. The time costs of signature generation and verification. (a) Signature generation. (b) Signature verification.

We further measure the performance of the signature generation and verification in detail, by processing the different number of blocks $n = \{100, 200, \ldots, 1000\}$. Fig. 6 shows that the operation time for signature generation and verification are proportional to the number of blocks; On processing the same number of blocks, our scheme takes less time than Shen's scheme.

In order to measure the performance of auditing, we set the different number of blocks challenged by the TPA, $c = \{100, 200, \ldots, 1000\}$. Fig. 7 illustrates the time cost of auditing process at the TPA side. And Fig. 8 illustrates the time cost of auditing at the cloud server side. Our scheme takes more ti me for generating a challenge than Shen's scheme but only needs less than 1.5 s to challenge 1000 blocks. As for the time costs for proof generation and verification, our scheme is 50 percent faster comparing with Shen's scheme. Therefore, it is can be concluded that our scheme enhances efficiency comparing with previous work. That is consistent with the theoretical analysis.

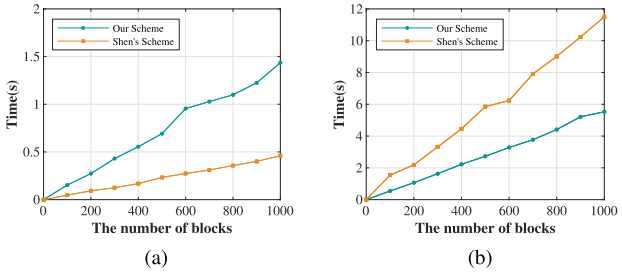


FIGURE 7. The time cost of auditing at the TPA side. (a) Challenge generation. (b) Proof verification.

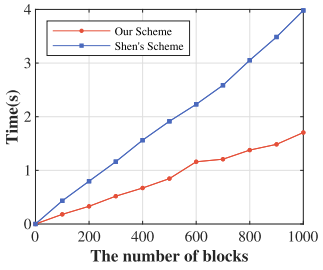


FIGURE 8. The time costs of auditing at the cloud server side.

VIII. CONCLUSION

In this paper, we investigated the auditing schemes for shared data in the cloud. We proposed a secure and efficient auditing scheme that supporting sensitive information hiding and malicious manager preventing. It allows the file-owner to share data with sensitive information and guarantees the integrity and authenticity of shared data to be fully trusted by the owner and researchers. The novel system model and mechanism for sensitive information hiding make the proposed scheme advantageous over previous work. Meanwhile, we gave the security analysis in detail to guarantee the robustness and soundness of our scheme. Theoretical analysis and experimental results demonstrate our scheme is more efficient than previous work.

REFERENCES

[1] J. L. Fernández-Alemán, I. C. Señor, P. Á. Lozoya, and A. Toval, "Security and privacy in electronic health records: A systematic literature review," *J. Biomed. Informat.*, vol. 46, no. 3, pp. 541–562, Jun. 2013.
[2] J. Sun and Y. Fang, "Cross-domain data sharing in distributed electronic health record systems," *IEEE Trans. Parallel Distrib. Syst.*, vol. 21, no. 6, pp. 754–764, Jun. 2010.
[3] J. Sun, X. Zhu, C. Zhang, and Y. Fang, "HCPP: Cryptography based secure EHR system for patient privacy and emergency healthcare," in *Proc. 31st Int. Conf. Distrib. Comput. Syst.*, Jun. 2011, pp. 373–382.
[4] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Comput.*, vol. 16, no. 1, pp. 69–73, Jan./Feb. 2012.
[5] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses," *Computer*, vol. 45, no. 1, pp. 39–45, Jan. 2012.
[6] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.
[7] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.

[8] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netow.*, Sep. 2008, p. 9.

[9] C. Erway and A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, New York, NY, USA, 2009, pp. 213–222. doi: 10.1145/1653662.1653688.

[10] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2011.

[11] Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu, "Dynamic audit services for outsourced storages in clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr./Jun. 2013.

[12] X. Chen, T. Shang, I. Kim, and J. Liu, "A remote data integrity checking scheme for big data storage," in *Proc. IEEE 2nd Int. Conf. Data Sci. Cyberspace (DSC)*, Jun. 2017, pp. 53–59.

[13] M. Sookhak, F. R. Yu, and A. Y. Zomaya, "Auditing big data storage in cloud computing using divide and conquer tables," *IEEE Trans. Parallel Distrib. Syst.*, vol. 29, no. 5, pp. 999–1012, May 2018.

[14] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," *IEEE Trans. Cloud Comput.*, vol. 2, no. 1, pp. 43–56, Jan./Mar. 2014.

[15] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Services Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

[16] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.* Berlin, Germany: Springer, 2012, pp. 507–525.

[17] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability," *J. Syst. Softw.*, vol. 113, pp. 130–139, Mar. 2016.

[18] W. Song, B. Wang, Q. Wang, Z. Peng, and W. Lou, "Tell me the truth: Practically public authentication for outsourced databases with multi-user modification," *Inf. Sci.*, vol. 387, pp. 221–237, May 2017.

[19] J. Han, Y. Li, J. Liu, and M. Zhao, "An efficient Lucas sequence-based batch auditing scheme for the Internet of medical things," *IEEE Access*, vol. 7, pp. 10077–10092, 2018.

[20] S. S. Al-Riyami and K. G. Paterson, "Certificateless public key cryptography," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2003, pp. 452–473.

[21] N. P. Smart, "Identity-based authenticated key agreement protocol based on weil pairing," *Electron. Lett.*, vol. 38, no. 13, pp. 630–632, Jun. 2002.

[22] B. Wang, B. Li, H. Li, and F. Li, "Certificateless public auditing for data integrity in the cloud," in *Proc. IEEE Conf. Commun. Netw. Secur. (CNS)*, Oct. 2013, pp. 136–144.

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

[24] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer, "Identity-based remote data possession checking in public clouds," *IET Inf. Secur.*, vol. 8, no. 2, pp. 114–121, Mar. 2014.

[25] D. He, S. Zeadally, and L. Wu, "Certificateless public auditing scheme for cloud-assisted wireless body area networks," *IEEE Syst. J.*, vol. 12, no. 1, pp. 64–73, Mar. 2018.

[26] D. He, N. Kumar, H. Wang, L. Wang, and K.-K. R. Choo, "Privacy-preserving certificateless provable data possession scheme for big data storage on cloud," *Appl. Math. Comput.*, vol. 314, pp. 31–43, Dec. 2017.

[27] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[28] L. Huang, G. Zhang, and A. Fu, "Certificateless public verification scheme with privacy-preserving and message recovery for dynamic group," in *Proc. Australas. Comput. Sci. Week Multiconf.*, Feb. 2017, Art. no. 76.

[29] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in identity-based cloud storage auditing for shared big data," *IEEE Trans. Dependable Secure Comput.*, to be published.

[30] Y. Liao, Y. Liang, A. W. Oyewole, and X. Nie, "Security analysis of a certificateless provable data possession scheme in cloud," *IEEE Access*, vol. 7, pp. 93259–93263, 2019.

[31] J. Tian and X. Jing, "A lightweight secure auditing scheme for shared data in cloud storage," *IEEE Access*, vol. 7, pp. 68071–68082, 2019.

[32] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.

[33] J. Sun, L. Ren, S. Wang, and X. Yao, "Multi-keyword searchable and data verifiable attribute-based encryption scheme for cloud storage," *IEEE Access*, vol. 7, pp. 66655–66667, 2019.

[34] Y. Liao, Y. Fan, Y. Liang, Y. Liu, and R. Mohammed, "Cryptanalysis of an identity-based encryption scheme with equality test and improvement," *IEEE Access*, vol. 7, pp. 75067–75072, 2019.

[35] Y. Liao, H. Chen, W. Huang, R. Mohammed, H. Pan, and S. Zhou, "Insecurity of an IBEET scheme and an ABEET scheme," *IEEE Access*, vol. 7, pp. 25087–25094, 2019.

[36] M. Ramadan, Y. Liao, F. Li, S. Zhou, and H. Abdalla, "IBEET-RSA: Identity-based encryption with equality test over RSA for wireless body area networks," *Mobile Netw. Appl.*, vol. 2019, pp. 1–11, 2019. doi: 10.1007/s11036-019-01215-9.

[37] Y. Liao, Y. He, F. Li, and S. Zhou, "Analysis of a mobile payment protocol with outsourced verification in cloud server and the improvement," *Comput. Standards Interfaces*, vol. 56, pp. 101–106, Feb. 2018.

[38] W. Huang, Y. Liao, S. Zhou, and H. Chen, "An efficient deniable authenticated encryption scheme for privacy protection," *IEEE Access*, vol. 7, pp. 43453–43461, 2019.

[39] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc. Int. Workshop Sel. Areas Cryptogr.* Berlin, Germany: Springer, 2002, pp. 310–324.

[40] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2008, pp. 90–107.

[41] Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 940–952, Apr. 2017.

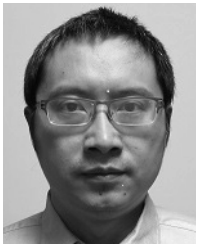
[42] H. Wang, "Proxy provable data possession in public clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 4, pp. 551–559, Oct./Dec. 2013.

[43] Y. Luo, M. Xu, K. Huang, D. Wang, and S. Fu, "Efficient auditing for shared data in the cloud with secure user revocation and computations outsourcing," *Comput. Secur.*, vol. 73, pp. 492–506, Mar. 2018.

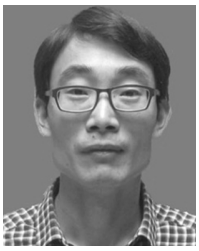
[44] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.* Berlin, Germany: Springer, 2001, pp. 514–532.

[45] B. Lynn. (Jun. 2013). *The Pairing-Based Cryptographic Library*. [Online]. Available: https://crypto.stanford.edu/pbc/

YU FAN received the B.S. degree in information engineering from the University of Electronic Science and Technology of China, in 2017, where he is currently pursuing the master's degree in cryptography. His research interests include security of cloud computing and remote auditing.



YONGJIAN LIAO received the Ph.D. degree in applied electronic science and technology from the College of Information Science and Electronic Engineering, Zhejiang University, in 2007. He is currently an Associate Professor with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His main research interests include public key cryptography and information security, in particular cryptographic protocols.



FAGEN LI received the Ph.D. degree in cryptography from Xidian University, Xi'an, China, in 2007. From 2008 to 2009, he was a Postdoctoral Fellow with Future University, Hakodate, Japan, which is supported by the Japan Society for the Promotion of Science (JSPS). He was a Research Fellow with the Institute of Mathematics for Industry, Kyushu University, Fukuoka, Japan, from 2010 to 2012. He is currently a Professor with the School of Computer Science and Engineering, University of Electronic Science and Technology of China (UESTC), Chengdu, China. He has published more than 80 papers in international journals and conferences. His recent research interests include cryptography and network security.

SHIJIE ZHOU received the Ph.D. degree in computer science and technology from the University of Electronic Science and Technology of China (UESTC), in 2004, where he is currently a Professor with the School of Information and Software Engineering. His research interests include cyber security communication and computer networks, traffic simulation, and artificial intelligence.

GANGLIN ZHANG is currently pursuing the bachelor's degree with the School of Information and Software Engineering, University of Electronic Science and Technology of China. His main research interests include cryptography and artificial intelligence security.