Identity-Based Cloud Storage Auditing for Data Sharing With Access Control of Sensitive Information

Yang Yang, Yanjiao Chen, *Senior Member, IEEE*, Fei Chen, and Jing Chen, *Member, IEEE*

***Abstract*—Remote data integrity auditing ensures the integrity of cloud storage. In practice, cloud users may not want their sensitive data to be exposed to others. Thus, it is meaningful to investigate how to realize data sharing with sensitive information hiding in cloud storage auditing. Up to now, cloud storage has been proven to achieve the sensitive information hiding property through a third-party sanitizer dedicated to sanitize user data, which leads to high outlays on purchasing and maintaining a special server. To meet this challenge, we design a novel cloud storage auditing protocol to support sensitive information hiding without the need of a third-party sanitizer. In addition, our scheme allows data owners to enable or disable other users to access their sensitive information with the help of the cloud that dose not deviate from the agreement during access control. To be specific, only after receiving the delegations from the data owner, the users can compute the valid warrants that can pass the access verification of the cloud. The proposed protocol is built on identity-based cryptography, thus avoiding the complex certificate management. We validate the advantages of the proposed protocol through massive theoretical analysis and experimental results.**

***Index Terms*—Access control, cloud storage auditing, data integrity, sensitive information hiding.**

I. Introduction

With the development of cloud storage, resource-limited data owners tend to store their abundant data on the cloud [1]. Unfortunately, user data on the cloud might be damaged because of internal mistakes and external attacks. For profit and reputation, the cloud might try its best to cover the accidents of data abnormality. Thus, data owners have to launch data integrity auditing against cloud storage from time to time. To this end, many cloud storage auditing protocols were proposed.

For data integrity auditing, cloud users store both data and its authenticators on the cloud, where the authenticators are utilized to prove the integrity of cloud storage. In practice, the outsourced data usually contains users' privacy. For example, electronic health records (EHRs) might involve patients' names and telephone numbers. If these EHRs are transferred to a disease research institute, the sensitive information of patients will be learned by the researchers. Thereafter, it is meaningful to implement cloud storage auditing for data sharing with sensitive information hiding.

At the first glance, it seems that we can easily achieve this goal through encrypting the entire data, computing the authenticators of the encrypted blocks, and outsourcing the encrypted blocks and their authenticators to the cloud. In this way, data owners can check the integrity of cloud storage. Although this method can achieve the property of sensitive information hiding in cloud storage, it will result in the failure of data sharing, i.e., other users cannot access the encrypted data. To address this problem, a possible solution is to distribute the decryption key to other users, which however makes the sensitive information no longer private. Thus, it is infeasible to realize the property of sensitive information hiding in cloud storage by encrypting the entire user data.

To meet this challenge, a decent solution for cloud storage auditing with sensitive information hiding was proposed in [2], in which a special server is introduced to sanitize sensitive blocks into uniform messy codes (e.g., multiple asterisks, question marks and so on) and transform their authenticators into valid ones. Nevertheless, this solution relies heavily on a third-party sanitizer, which will result in a large amount of outlays on purchasing and maintaining a special server. Besides, sanitizing data on a third-party server will incur additional overheads and security challenge due to the following reasons. The sanitizer needs to record the sanitizing values of the sensitive blocks for each user, and cloud users cannot directly interact with the cloud, but have to transmit and receive data via the sanitizer. Furthermore, the sanitizer might also lose data due to the same reasons as the cloud, making

Manuscript received July 14, 2021; revised September 29, 2021; accepted October 16, 2021. Date of publication October 21, 2021; date of current version June 23, 2022. The work of Yang Yang was supported by the Fundamental Research Funds for the Central Universities under Grant 2042021kf1030 and Grant 2722021BX025. The work of Yanjiao Chen was supported in part by the National Natural Science Foundation of China under Grant 61972296, and in part by the Wuhan Application Advanced Funding under Grant 2019010701011419. The work of Fei Chen was supported in part by the National Natural Science Foundation of China under Grant 61872243, and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A151501489. The work of Jing Chen was supported by the National Natural Science Foundation of China under Grant U1836202, Grant 61772383, Grant 62076187, and Grant 62172303. *(Corresponding author: Yanjiao Chen.)*

Yang Yang is with the School of Information and Security Engineering, Zhongnan University of Economics and Law, Wuhan 430073, China (e-mail: yaoyuandepiaoxue@126.com).

Yanjiao Chen is with the College of Electrical Engineering, Zhejiang University, Hangzhou 310007, Zhejiang, China (e-mail: chenyanjiao@zju.edu.cn).

Fei Chen is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: fchen@szu.edu.cn).

Jing Chen is with the Computer School, Wuhan University, Wuhan 430072, China (e-mail: chenjing@whu.edu.cn).

Digital Object Identifier 10.1109/JIOT.2021.3121678

sensitive information irrecoverable. Thereafter, it is necessary to achieve the sensitive information hiding property in cloud storage auditing protocols without the help of a third-party sanitizer.

In practice, data owners might need some other users to access their sensitive information within a certain period of time. This brings a new challenge in the design of cloud storage auditing for data sharing with sensitive information hiding. For example, a patient Alice is usually required to share her sensitive medical records with the doctor during an appointment, which is called access authorization. If Alice is referred to another doctor for some reason, she might not want the previous doctor to learn her sensitive medical records any more. This means that Alice should be able to revoke access authorization of her original doctor. Therefore, it is desirable that data owners can enable and disable other users to access their sensitive information in cloud storage auditing. However, this challenge is unexplored in previous works.

A. Contribution

To meet the above challenges, we investigate identity-based (ID-based) cloud storage auditing protocol for data sharing with sensitive information hiding. Compared with existing proposals, our contributions are as follows.

1) We realize the sensitive information hiding property in cloud storage without the help of a third-party sanitizer, while keeping the remote data integrity auditing function still in effect. The key is that the cloud decides whether to provide the actual data or the uniform messy codes for the sensitive blocks based on the authorization of users. Thanks to no involvement of third-party sanitizers, our solution has the following advantages. First, the substantial outlays of purchasing and maintaining a special server for sanitizing user data can be completely avoided. Second, the storage cost can be hugely reduced because no sanitizing values of sensitive blocks need to be stored. Third, the communication cost can be reduced by about a half since data owners can directly interact with the cloud without the relay of the sanitizer.
2) We explore how to allow data owners to enable and disable other users to access their sensitive information in cloud storage auditing. We present a novel concept, called ID-based cloud storage auditing, for data sharing with the access control of sensitive information. In our solution, users' sensitive information is shared with others in a controllable way, which is implemented by distributing well-designed delegations to authorized users. To the best of our knowledge, our protocol is the first solution with such functionalities.
3) We give a detailed proof about the correctness and the security of the proposed protocol. We also evaluate the performance of the proposed protocol through theoretical analysis and experimental results. Compared with the state-of-the-art method [2], the proposed protocol has improved storage and communication and computational costs. In addition to cloud storage auditing with sensitive information hiding, the proposed protocol can also efficiently support access control of sensitive information.

B. Related Work

To check the integrity of cloud storage, a large number of remote auditing protocols were proposed. To relieve the computation burden, users usually outsource their auditing to the third-party auditor (TPA). Ateniese *et al.* [3] first considered cloud storage auditing in a well-designed provable data possession (PDP) model, which enables that the verifiers can handle integrity verification without possessing actual data. Juels and Kaliski, Jr. [4] utilized spot checking and error correcting codes to achieve cloud storage auditing. Shacham and Waters [5] utilized pseudo-random function and BLS signature to realize cloud storage auditing.

To dynamically update user data, Ateniese *et al.* [6] initially designed a PDP protocol with partial data dynamics, which however only supports a few number of data updates. To support full data dynamics, the following protocols resorted to index arrays [7]–[10], linked lists [11]–[13], or data authenticators without indices [14]. To eliminate the damage caused by the exposure of user key, Yu *et al.* [15]–[17] designed a PDP protocols with key-exposure resistance by dynamically updating users' keys. To further improve the performance, Chen *et al.* [10] constructed a PDP protocol by using the distributed string equality checking technique. Zhang *et al.* [19] proposed a PDP protocol by using the discrete logarithm (DL) problem. These two protocols contain only basic algebraic operations, in which the most complicated operation is just exponential computation.

The above protocols [3]–[19] all depend on public key infrastructure and, thus, introduce complicated certificate management to ensure the genuineness of their public keys. To address this challenge, Wang *et al.* [20] designed an ID-based PDP protocol based on an ID-based cryptography. In such a scheme, cloud users take their identities (e.g., names and telephone numbers) as the public keys and, thus, do not resort to PKI any more. Following this seminal work, many ID-based cloud storage auditing protocols were proposed in order to fit more application scenarios. Wang *et al.* [21] designed an ID-based protocol, which supports proxy-oriented data outsourcing. Yu *et al.* [22] constructed an ID-based protocol based on asymmetric group key agreement, which achieves zero-knowledge preservation of user privacy. Wang *et al.* [23] designed an ID-based protocol to support unconditionally anonymous integrity auditing, which protects the privacy of user identity and provides an incentive for cloud user to reveal dishonest incidents. Li *et al.* [24] and Zhou *et al.* [25] constructed an ID-based protocol, which enables certificateless public auditing against escrow attacks. Li *et al.* [26] presented a fuzzy ID-based protocol, which further simplifies key management. Huang *et al.* [27] came up with an ID-based protocol, which supports incentive cloud storage auditing for nonmanager groups. Shen *et al.* [2] presented an ID-based protocol with sensitive information hiding, which achieves perfect user privacy through sanitizing the sensitive information in cloud storage.

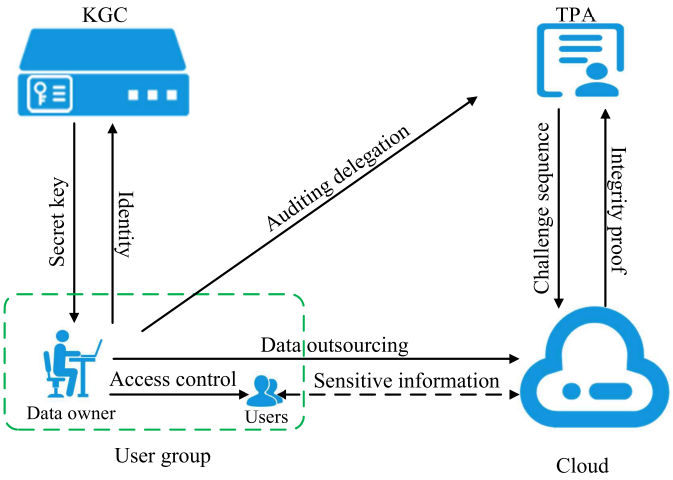

Fig. 1. System model.

Data sharing among a group of users is a widely used service in cloud storage. In practice, it is common that a cloud user joins and leaves a group. For example, Alice joins a group due to job demand, and when she is retired, she is required to be revoked from this group. Thus, how to manage user authorization is very important in cloud storage auditing. To enable data owners to control whether other users can perform the data integrity auditing against their cloud storage, Wang *et al.* [28] constructed an ID-based protocol with the control of data integrity auditing, which is built on the proxy resignature technique. Yuan and Yu [29] came up with an ID-based protocol with the control of data integrity auditing by using polynomial-based generation and proxy-based update of data authenticators. Zhang *et al.* [31] explored an ID-based protocol with the control of data integrity auditing, which is based on a new strategy for the generation and the update of user keys. Different from the above protocols with the control of data integrity auditing, this article explores how to design an ID-based protocol for data sharing with the control of sensitive information access.

C. Organization

The reminder of this article is organized as follows. Section II describes the preliminaries. Section III illustrates the proposed protocol. Section IV shows the theoretical analysis. Section V focuses on the performance evaluation. The conclusion of this article is provided in Section VI.

II. Preliminaries

This section first introduces the system and security models with respect to the ID-based cloud storage auditing, and then reviews cryptographic tools adopted in this article.

A. System Model

As depicted in Fig. 1, the system model contains four entities: 1) the cloud; 2) the users; 3) the TPA; and 4) the key generation center (KGC).

Cloud: The cloud is a powerful entity, which provides abundant storage resources to users. The cloud also controls user access through checking the validity of the received warrant.

Users: Users subscribe to the data sharing service provided by the cloud. Users may or may not intend to share their data with others. Those who provide data are called *data owners*. The data owner stores abundant data on the cloud. Through cloud storage, the data owner shares his/her data with other users under the condition that the sensitive information is available in a controllable way.

KGC: The KGC is a trustworthy entity that generates system public parameters and users' secrete keys.

TPA: The TPA is an impartial entity that checks the integrity of cloud storage.

In our system, the KGC distributes users' secret keys according to their identities. After receiving the secret key, the data owner first generates the authenticators of the data blocks in the file, and then stores them together on the cloud, where the authenticators are utilized to prove the integrity of cloud storage. For data auditing, the TPA transmits an auditing sequence to the cloud for requesting the integrity proof, and then checks the integrity of cloud storage by validating the received proof. In addition, a data owner can enable or disable other members of the same group to access his/her sensitive information with the help of the cloud. To be specific, the user sends a warrant to the cloud for data access, which is generated based on the delegation from the data owner. If this warrant is verified, the cloud returns actual data blocks; otherwise, the cloud feeds back actual data blocks for insensitive blocks and uniform messy codes for sensitive blocks. In this article, the cloud is assumed to not deviate from the agreement during access control since it is often provided by credible organizations, such as Google, Amazon, and so on.

There are two main threats to the security of the outsourced user data. First, the cloud may try its best to cover abnormal incidents of data loss for its own interests. Second, the TPA and users may elicit other users' sensitive information. Thus, our ID-based cloud storage auditing protocol has the following design goals.

1) *Correctness:* If the KGC, the cloud, the TPA and the group users honestly follow the protocol, they can always pass the correctness checking in the protocol.
2) *Auditing Soundness:* The cloud cannot successfully cheat the TPA with incorrect user data.
3) *Detectablity:* The TPA can detect the incidents of data loss with a nonnegligible probability.
4) *Sensitive Information Hiding:* The sensitive information of the data owner is prevented from the TPA and other users.

We formalize the compact framework of an ID-based cloud storage auditing protocol for data sharing with the access control of sensitive information as follows.

Definition 1: As shown in Fig. 2, an ID-based cloud storage auditing protocol for data sharing with the access control of sensitive information contains eight algorithms.

1) *Setup:* run by the KGC. Given a security parameter, the KGC produces the master secret key *MSK* and the system public key *PK*.

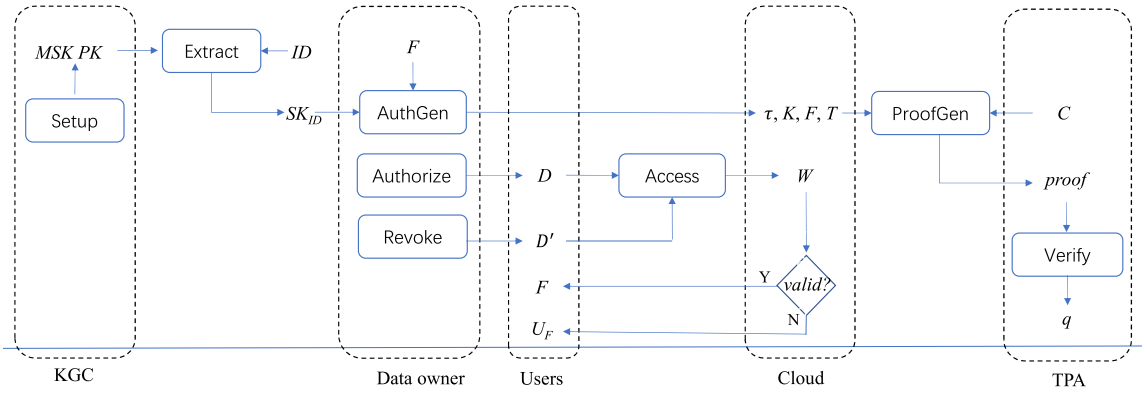

Fig. 2. Framework of an ID-based cloud storage auditing protocol for data sharing with the access control of sensitive information.

2) *Extract:* run by the KGC and the users. Given a user's identity *ID*, the KGC computes this user's secret key SK_{ID}. The user accepts the distributed secret key if it is verified; otherwise, just refuses it.
3) *AuthGen:* run by the data owner. Given a data file F, the data owner generates its tag τ, a set of its authenticators T, and a set of the indices of its sensitive blocks K.
4) *ProofGen:* run by the cloud. Given the data file, the authenticators and the auditing sequence, the cloud produces an integrity proof.
5) *Verify:* run by the TPA. Given an integrity proof, the TPA outputs $q = 1$ if the proof is verified; otherwise, the TPA outputs $q = 0$.
6) *Authorize:* run by the data owner and users. For access authorization, the data owner issues a delegation D to each user who are allowed access to the data.
7) *Revoke:* run by the data owner, the nonrevoked users and the cloud. For access revocation, the data owner updates the delegations of all nonrevoked users.
8) *Access:* run by the authorized users and the cloud. With input of the delegation, the authorized user first generates a warrant for data access and transmits it to the cloud. If the warrant is verified, the cloud returns actual data blocks in F; otherwise, the cloud feeds back actual data blocks for insensitive blocks and uniform messy codes for sensitive blocks, denoted as U_F.

B. Security Model

Our security model is built on a security game, in which the data owner or the TPA is the challenger and the malicious cloud is the adversary.

1) *Setup:* The challenger produces the master secret key and the public key using the Setup algorithm. The challenger returns the public key to the adversary.
2) *Query:* The adversary can launch the following queries to the challenger.
   a) *Querying Hash Value:* The adversary can inquire any type of hash value for an arbitrary input. The challenger computes the hash value according to the inputs of the adversary, which is then fed back.
   b) *Querying File Tag:* The adversary can request the tag of an arbitrary file. The challenger generates the tag of the input file through executing part of AuthGen, and then replies it to the adversary.
   c) *Querying User Secret Keys:* The adversary can ask for the secret key for an arbitrary identity. The challenger first produces a user secret key through Extract, and then returns it to the adversary.
   d) *Querying Data Authenticators:* The adversary can query the authenticator of an arbitrary data block, no matter whether it is sensitive or not. The challenger first computes user secret key according to Extract, then generates the authenticators of the input data blocks using AuthGen, and finally transmits these authenticators to the adversary.
   e) *Querying Integrity Proof:* The adversary can query the integrity proof of arbitrary data blocks. The challenger produces the integrity proof of the input data blocks by executing Extract, AuthGen and ProofGen in turn, which is then returned to the adversary.
3) *Challenge:* The challenger launches an auditing sequence to the adversary for requesting the integrity proof.
4) *Forgery:* The adversary forges an integrity proof as the challenge feedback to the challenger. If this proof is always accepted by the challenger, the adversary is regarded as the winner of this security game.

From the above game, we have the following security definition.

Definition 2 (Auditing Soundness): An ID-based cloud storage auditing protocol is considered to be secure if there is a knowledge extractor which can extract all the audited data when the adversary succeeds in the above security game with the challenger.

Next, we formalize the detectability of an ID-based cloud storage auditing protocol according to the probability of detecting damaged data blocks.

Definition 3 (Detectability): An ID-based cloud storage auditing protocol is detectable if the probability of the damaged data blocks being detected is non-negligible when the number of audited blocks is sufficiently large.

Finally, we present a formal definition to capture the security requirement of sensitive information hiding.

TABLE I
NOTATIONS

Notation	Description
G_1, G_2	Multiplicative cyclic groups with the prime order p
e	A bilinear map: $G_1 \times G_1 \rightarrow G_2$
$\|\|$	A string concatenation
H	A hash function
F	A data file
T	A set of authenticators of the data blocks in F
K	A set of indices of sensitive blocks in F
C	A challenge sequence
D	A delegation for the data owner to authorize user access
W	A warrant for the cloud to detect if user access is authorized

Definition 4 (Sensitive Information Hiding): An ID-based cloud storage auditing protocol guarantees sensitive information hiding if only the data owners and the permitted users can learn the sensitive information from cloud storage.

C. Mathematical Background

Bilinear Map: Assume that there are two cyclic multiplicative groups G_1 and G_2, where $G_1 = <g>$ and their orders are both a prime p. In addition, $e : G_1 \times G_1 \rightarrow G_2$ is a bilinear map, satisfying the following.

1) *Bilinearity:* $\forall \alpha, \beta \in G_1$ and $\forall x, y \in Z_p^*$, we have $e(\alpha^x, \beta^y) = e(\alpha, \beta)^{xy}$.
2) *Computability:* e can be computed within a reasonable time.
3) *Nondegeneracy:* $e(g, g) \neq 1$ is always true.

Computational Diffie–Hellman (CDH) Problem: Given $g, g^x, g^y \in G_1$ (but not $x, y \in Z_p^*$), the problem is to solve g^{xy}. The probability of addressing this problem is negligible.

Discrete Logarithm Problem: Given $g, g^x \in G_1$, the problem is to solve x. The probability of solving this problem is negligible.

Finally, we show the main notations throughout this article in Table I.

III. PROPOSED PROTOCOL

A. Overview

According to the state-of-the-art approach [2], in order to sanitize sensitive information into uniform messy codes, it is required to transmit user data to a third-party sanitizer before storing the data on the cloud. This results in outlays in purchasing and maintaining the hardware/software of the sanitizer, and large overheads. To save costs, our solution enables the cloud, instead of the sanitizer, to sanitize users' sensitive blocks into uniform messy codes. In addition, we also consider how to enable or disable a user to access the sensitive data of a specific data owner. To meet the above challenges, we propose a novel ID-based cloud storage protocol for data sharing with the access control of sensitive information, in which the data owners directly transmit their data to the cloud without the relay of a third-party sanitizer. When an unauthorized user requests a data file that contains sensitive information, the cloud feeds back actual data blocks for insensitive blocks and uniform messy codes for sensitive blocks. Moreover, the proposed protocol is designed using ID-based cryptography, which avoids complex certificate management.

In our design, the KGC distributes cloud users' secret keys according to their identities. After receiving the secret key, the cloud user sets out to verify its correctness. Before outsourcing a file to the cloud, the data owner has to compute an authenticator for each data block in the file. These authenticators are utilized to prove the integrity of cloud storage. In addition, the data owner also generates a tag to ensure the correctness of public parameters related to the outsourced file, which is transmitted to the cloud along with the data blocks and their authenticators. In each data auditing, the TPA first transmits a challenge sequence to the cloud for requesting an integrity proof, and then checks the integrity of cloud storage by judging whether the received proof is valid or not. By using the delegations of access authorization, a data owner is able to enable or disable other users to access its sensitive information. For data access, the authorized user first generates a warrant based on the received delegation, and then transmits it to the cloud. If the cloud accepts this warrant, the cloud returns the actual blocks; otherwise, the cloud feeds back the actual blocks for insensitive blocks and the uniform messy codes for sensitive blocks.

B. Detailed Design

Assume that a data owner with the identity *ID*, denoted as U_{ID}, possesses a large file F to store. The unique identifier of F is denoted as name. F can be divided into $F = (d_1, d_2, \ldots, d_n)$ according to its size. We utilize a digital signature *SSig* to ensure the correctness of verification parameters related to F, in which the signing public and secret keys are denoted as *spk* and *ssk*, respectively.

Next, we show the details of the proposed protocol.

1) *Setup*$(\lambda) \rightarrow (MSK, PK)$*:* In this algorithm, the KGC produces the master secret key *MSK* and the public key *PK*.
   a) The KGC determines two multiplicative cyclic groups G_1 and G_2 with the prime order p, a generator g of G_1, a bilinear map $e : G_1 \times G_1 \rightarrow G_2$, a random element $u \in G_1$, and a hash function $H_1 : \{0, 1\}^* \rightarrow G_1$.
   b) The KGC selects a random $x \in Z_p^*$ as the master secret key, i.e., $MSK = x$.
   c) The KGC sets the public key as $PK = (G_1, G_2, e, p, g, g^x, u, H_1)$.
2) *Extract*$(ID; MSK, PK) \rightarrow SK_{ID}$*:* In this algorithm, the KGC first distributes secret keys to corresponding users according to their identities. Users validate the correctness of the received keys.

a) Once receiving a user identity ID, the KGC generates the secret key as $SK_{ID} = H_1(ID)^x$, which is then transmitted to user U_{ID}.
b) When obtaining the secret key SK_{ID}, user U_{ID} checks its correctness by judging whether

$$e(g, SK_{ID}) = e\big(g^x, H_1(ID)\big) \tag{1}$$

holds or not. If yes, U_{ID} accepts the secret key SK_{ID}; otherwise, U_{ID} discards it and requests a new secret key by Extract again.

3) *AuthGen*$(F; SK_{ID}, PK) \rightarrow (\tau, K, T)$*:* In this algorithm, a data owner generates a list of authenticators for all blocks in its data file, and computes a tag to ensure the correctness of verification parameters related to user file. The data owner uploads the file tag, the set of indices of sensitive blocks, the data blocks, and their authenticators to the cloud.
   a) To support data integrity auditing, the data owner U_{ID} first determines a random element $r_t \in Z_p^*$, and then generates the authenticator of each data block as $t_i = SK_{ID} \cdot (H_1(\text{name}||i) \cdot u^{d_i})^{r_t}$, where $1 \le i \le n$. Let $T = (t_1, t_2, \ldots, t_n)$ denote the set of authenticators of file F.
   b) To guarantee the integrity of verification parameters, U_{ID} calculates the file tag as $\tau = \text{name}||n||g^{r_t}||g^{r_s}||SSig(\text{name}||n||g^{r_t}||g^{r_s}||ssk_{ID}) || spk_{ID}$, where $r_s \in Z_p^*$ is used to control the access of sensitive information.
   c) U_{ID} outsources $\{\tau, K\ F, T\}$ to the cloud, which are then removed from the local storage. K is the set of indices of sensitive data blocks in F.

4) *ProofGen*$(\tau, F, T, C) \rightarrow$ *proof:* In this algorithm, the cloud generates the integrity proofs after receiving the TPA's challenging sequences.
   a) The TPA first acquires the tag τ of the challenged file and then makes sure its correctness by judging whether $SSig(\text{name}||n||g^{r_t}||g^{r_s}; ssk_{ID})$ in τ is a valid-identity-based signature by spk_{ID}. If it is invalid, this means that the user file F is damaged, and the TPA reports this failure; otherwise, the TPA parses the tag τ to get the total number n of data blocks, and then produces a nonempty challenge sequence $C = \{i_1, i_2, \ldots, i_c; H_t\}$, where $i_j \in Z_n^*$, $H_t : \{0,1\}^* \rightarrow Z_p^*$ and $1 \le j \le c$. The TPA transmits the auditing sequence C to the cloud.
   b) When obtaining challenge C, the cloud first picks a secret element r_c at random, which is used to hide the sensitive information from the TPA. The cloud calculates an aggregated value of all audited blocks as $\mu = \sum_{j=1}^{c} s_j d_{i_j} + r_c$, where $s_j = H_t(j)$. The cloud aggregates the challenged authenticators into $\nu = \prod_{j=1}^{c} t_{i_j}^{s_j}$, and computes $R = e(u, g^{r_t})^{r_c}$. The cloud generates an integrity proof as $proof = \{\mu, \nu, R\}$, which is then fed back to the TPA for integrity verification.

5) *Verify*$(proof, C; PK) \rightarrow q$*:* In this algorithm, the TPA handles the integrity verification of cloud storage. This is implemented by judging whether

$$R \cdot e(\nu, g) = e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \cdot e\left(\prod_{j=1}^{c} H_1\big(\text{name}||i_j\big)^{s_j} \cdot u^{\mu}, g^{r_t}\right) \tag{2}$$

holds or not. If yes, the cloud storage is considered to be intact, indicated by $q = 1$; otherwise, the cloud storage is considered to be compromised and $q = 0$.

6) *Authorize*$(U_{ID}, U'_{ID}) \rightarrow D$*:* In this algorithm, the data owner enables other users to access his/her sensitive information.
   a) To authorize a user $U_{ID'}$ for sensitive information access, the data owner U_{ID} generates a delegation $D = (\omega, \sigma_\omega)$, where $\omega = ID||ID'||g^{r_s}||g^{r_s\prime}$ and $\sigma_\omega = H_1(\omega)^{r_s}$. Note that i) r_s and r'_s are kept private by U_{ID} and $U_{ID'}$, respectively and ii) g^{r_s} and $g^{r'_s}$ are the public parameters of U_{ID} and $U_{ID'}$, respectively. The delegation D is transmitted to $U_{ID'}$.
   b) Once receiving the delegation D, $U_{ID'}$ first checks its correctness by judging whether

$$e(\sigma_\omega, g) = e\big(H_1(\omega), g^{r_s}\big) \tag{3}$$

holds or not. If yes, $U_{ID'}$ accepts the delegation D which means that $U_{ID'}$ is authorized by U_{ID} to access its sensitive information; otherwise, $U_{ID'}$ rejects the delegation.

7) *Revoke*$(U_{ID}, U'_{ID}) \rightarrow (D, \tau)$*:* In this algorithm, the data owner revokes the access authorization of other users.
   a) To revoke the access authorization of $U_{ID'}$, the data owner U_{ID} first picks a new random element $\widehat{r_s}$ instead of r_s, and then updates the delegations of all nonrevoked users by implementing the Authorize algorithm.
   b) The cloud replaces g^{r_s} in the file tag with the new value $g^{\widehat{r_s}}$.

8) *Access*$(D) \rightarrow W$*:* In this algorithm, a user transmits a warrant to the cloud for accessing the sensitive information of a data owner. Depending on the validity of the received warrant, the cloud feeds back the actual sensitive blocks or the uniform messy codes.
   a) To access the sensitive information of U_{ID}, user $U_{ID'}$ first generates a warrant $W = (\omega, \sigma_\omega \sigma'_\omega)$ with the input of D, where $\sigma'_\omega = H_1(\omega)^{r'_s}$. Then, $U_{ID'}$ sends W to the cloud.
   b) The cloud first checks whether ID and ID' in the W are the identities of the owner of the data being accessed and the user that is interacting, and then checks whether g^{r_s} in the received warrant W and g^{r_s} in the file tag τ are equal or not, finally checks the validity of W by judging

$$e\big(\sigma_\omega \sigma'_\omega, g\big) = e(H_1(\omega), g^{r_s}) \cdot e\Big(H_1(\omega), g^{r'_s}\Big) \tag{4}$$

holds or not. If all these conditions are met, the cloud transmits the actual data blocks to $U_{ID'}$; otherwise, the cloud provides actual data blocks for

insensitive blocks and uniform messy codes for sensitive blocks.

Note that the above proposed protocol can easily achieve the sensitive information hiding from the cloud by using the random masking technique. To be specific, the data owner first blinds the sensitive data blocks into $d_i^* = d_i + \gamma_i$ before ProofGen, where $i \in K$ and $\gamma_i = H_\gamma(\text{name}||i) \in Z_p$. In such a way, the cloud can hardly learn user sensitive information due to the unknown of H_γ. On the other hand, the data owner transmits H_γ to the authorized users so that they can reconstruct sensitive data blocks. When a user is revoked, the data owner first updates the blinded sensitive data blocks using another hash function $H_{\gamma'}$, and then only shares $H_{\gamma'}$ with the user still authorized. Generally, a data file contains not too much sensitive information [2], making the data owner only need to updates a few fields.

IV. Theoretical Analysis

In this section, we provide theoretical analysis of our protocol from the perspective of correctness, soundness, detectability, and privacy.

Theorem 1 (Correctness): The correctness of our proposed is guaranteed as: 1) if the user secret key from the KGC is correct, the distributed user always accepts it; 2) if the cloud storage is intact, the integrity proof always passes the verification of the TPA; 3) if the data owner is honest, the user always accepts his/her access authorization; and 4) if the warrant from the authorized user is correct, the cloud always provides the actual data.

Proof:

1) The correctness of key distribution depends on (1), which can be easily derived as
$$e(g, SK_{ID}) = e\big(g, H_1(ID)^x\big) = e\big(g^x, H_1(ID)\big).$$
2) The auditing correctness relies on a checking equation, as shown in (2), which can be represented as
$$\begin{aligned} R \cdot e(\nu, g) &= e\big(u^{r_c}, g^{r_t}\big) \cdot e\left(\prod_{j=1}^{c} t_{i_j}^{s_j}, g\right) \\ &= e\big(u^{r_c}, g^{r_t}\big) \cdot e\left(\prod_{j=1}^{c} H_1(ID)^{x \cdot s_j}, g\right) \\ &\quad \cdot e\left(\prod_{j=1}^{c} \Big(H_1(\text{name}||i_j) \cdot u^{d_i}\Big)^{r_t s_j}, g\right) \\ &= e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \\ &\quad \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j}, g^{r_t}\right) \\ &\quad \cdot e\Big(u^{\sum_{j=1}^{c} s_j d_{i_j} + r_c}, g^{r_t}\Big) \\ &= e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \\ &\quad \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j} \cdot u^{\mu}, g^{r_t}\right). \end{aligned}$$
3) The correctness of access authorization is determined by (3), which can be proved as
$$e(\sigma_\omega, g) = e\big(H(\omega)^{r_s}, g\big) = e\big(H_1(\omega), g^{r_s}\big).$$
4) Whether the authorized users can download the sensitive blocks depends on (4), which can be deduced as
$$\begin{aligned} e\big(\sigma_\omega \sigma'_\omega, g\big) &= e\big(H_1(\omega)^{r_s}, g\big) \cdot e\Big(H_1(\omega)^{r'_s}, g\Big) \\ &= e\big(H_1(\omega), g^{r_s}\big) \cdot e\Big(H_1(\omega), g^{r'_s}\Big). \end{aligned}$$
■

Theorem 2 (Auditing Soundness): In the proposed protocol, if the challenged blocks are not intact, the cloud cannot forge a valid integrity proof that can be accepted by the TPA.

Proof: Our proof is based on the method of knowledge proof. If the cloud can forge a valid integrity proof without possessing the intact data, we can extract the intact audited data through multiple interactions between our protocol and a knowledge extractor. Next, we provide the detailed proof using the following four games.

Game 0: This game is formalized in Section II and, thus, is omitted here.

Game 1: On the basis of Game 0, this game introduces an additional assumption: the challenger keeps all its signed tags response to the adversary's queries. If the adversary can issue a valid tag which is not signed by the challenger, the challenger reports failure and then terminates.

Analysis: Assume that the challenger terminates with an overwhelming probability in Game 1. It also implies that the adversary can crack the signing secret key *ssk* of the digital signature *SSig*, which however is contradictory with the security of *SSig*. Hence, the correctness of verification parameters related to the outsourced data file can be guaranteed.

Game 2: On the basis of Game 1, this game introduces the following assumption: the challenger records all the user keys that it responds to the adversary. If the adversary can provide a valid tag which is not returned from the challenger, the challenger declares failure and then aborts.

Analysis: Assume that the challenger aborts with a non-negligible probability in Game 2. It also means that the adversary can compute the master key x from $H(ID)^x$, where *ID* is a user identity provided by the adversary. However, it contradicts the hardness of the DL problem. As a result, it is almost impossible for the adversary to compute any user's secret key due to the unknown of x, and then the security of user secret key can be guaranteed.

Game 3: On the basis of Game 2, this game introduces the following assumption: the challenger records all its generated authenticators and integrity proofs that are taken as the response to the adversary. The challenger verifies the correctness of each proof from the adversary. If the integrity proof produced by the adversary can pass the validation of the challenger, but the aggregate ν is not equal to ν' that is computed using the locally recorded data, the challenger declares failure and then terminates.

Analysis: We first build a simulator that tries to address the CDH problem, i.e., given g, g^z, and h, its goal is to compute h^z. Then, we demonstrate that if the adversary always wins Game 3 against the challenger, the simulator can address the

CDH problem with an overwhelming probability. The simulator behaves in the same way as the challenger in Game 2, except the following.

1) The simulator randomly selects an element $z \in Z_p^*$ and then computes g^z. The simulator also randomly determines two elements $a, b \in Z_p^*$ and then sets $u = g^a \cdot h^b$.
2) The simulator programs the random oracle as $H_1(\text{name}||i) = g^{e_i}/(g^{ad_i} \cdot h^{bd_i})$, where e_i is a random element in Z_p^*. According to the AuthGen algorithm, the simulator can compute the authenticator of the input block as $t_i = SK_{ID} \cdot (H_1(\text{name}||i) \cdot u^{d_i})^{r_t}$, where $SK_{ID} = H(ID)^x$ is extracted by using the Extract algorithm, and r_t is a random element in Z_p^*.
3) The simulator performs the integrity auditing through the interaction with the adversary. As formalized in this game, if the adversary passes the integrity verification, but its aggregated values ν is not as expected, this game is terminated.

On the one hand, since the proof (μ, ν, R) from the adversary is assumed to pass the integrity verification, it can make the following equation hold:

$$R \cdot e(\nu, g) = e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j} \cdot u^{\mu}, g^{r_t}\right). \quad (5)$$

On the other hand, the simulator can utilize the locally recorded blocks and authenticators to compute the aggregated values μ' and ν'.

According to the correctness of the proposed protocol, we can obtain

$$R \cdot e\big(\nu', g\big) = e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j} \cdot u^{\mu'}, g^{r_t}\right). \quad (6)$$

It is obvious that $\mu \neq \mu'$; otherwise, we will have $\nu = \nu'$, which contradicts with the assumption in this game. Let $\Delta\mu = \mu' - \mu$, which is not equal to 0. Dividing (6) by (5), we have

$$e(\nu'/\nu, g) = e\big(u^{\Delta\mu}, g^{r_t}\big). \quad (7)$$

By assuming $g^{r_t} = g^z$, we can convert (7) into

$$e\big(\nu'/\nu, g\big) = e\Big((g^a \cdot h^b)^{\Delta\mu}, g^z\Big) = e\Big((g^{za\Delta\mu} \cdot h^{zb\Delta\mu}), g\Big). \quad (8)$$

From (8), we can observe that

$$h^z = \big(\nu'/\nu \cdot g^{-za\Delta\mu}\big)^{1/(b\Delta\mu)}.$$

According to the above equation, we find that the CDH problem is unsolvable only if $b\Delta\nu = 0$. The probability that $b\Delta\nu = 0$ is at most $1/p$, which is negligible due to the large value of p. This means that the simulator is able to address the CDH problem with a high probability $1 - 1/p$, which is contradictory with the hardness assumption of the CDH problem. Thereafter, the adversary can hardly win Game 3 against the challenger. It also implies that the success probabilities of the adversary in Game 2 and Game 3 have a negligible difference.

Game 4: On the basis of Game 3, this game introduces an additional assumption: if the aggregated block μ is not equal to the expected μ' aggregated by the local blocks, the challenger reports failure and terminates.

Analysis: We first build a simulator that aims to address the DL problem. Specifically, the simulator tries to compute an element z satisfying $h = g^z$, in which g and h are given. Then, we show that if the adversary always succeeds in this game, the simulator is able to address the DL problem with a high probability. The simulator performs in the same way as the challenger in Game 3, except the following.

1) The simulator picks two random elements $a, b \in Z_p^*$ and then sets $u = g^a \cdot h^b$.
2) The simulator performs the integrity auditing task through the interaction with the adversary. As specified in Game 3, if the adversary wins, but its aggregated μ is not as expected, this game is terminated.

Due to the same reason, the simulator can also obtain (5) and (6) in this game like in Game 3. Through these two equations, we observe that

$$\begin{aligned} &e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j} \cdot u^{\mu}, g^{r_t}\right) \\ &= R \cdot e(\nu, g) = R \cdot e\big(\nu', g\big) \\ &= e\Big(H_1(ID)^{\sum_{j=1}^{c} s_j}, g^x\Big) \cdot e\left(\prod_{j=1}^{c} H_1(\text{name}||i_j)^{s_j} \cdot u^{\mu'}, g^{r_t}\right). \end{aligned} \quad (9)$$

From Game 3, we can obtain $\nu = \nu'$. According to (9), we have $u^{\mu} = u^{\mu'}$, which further implies that

$$1 = u^{\Delta\mu} = \Big(g^a \cdot h^b\Big)^{\Delta\mu} = g^{a\Delta\mu} \cdot h^{b\Delta\mu} \quad (10)$$

where $\Delta\mu = \mu' - \mu$. According to the assumption in this game, we have $\Delta\mu \neq 0$, and then (10) can be transformed into

$$h = g^{-\frac{a\Delta\mu}{b\Delta\mu}} = g^{-\frac{a}{b}}$$

which is the solution of the DL problem, unless b is equal to 0. The probability of $b = 0$ is only $1/p$, which is negligible. This means that the simulator can address the DL problem with a high probability $1 - 1/p$, which is contradictory to the hardness assumption of the DL problem. Therefore, the adversary cannot succeed in Game 4 against the challenger with an overwhelming probability. It also implies that the success probabilities of the adversary in Game 3 and Game 4 have a negligible difference.

Based on the above discussion, the difference among these games is negligible. Thus, there exists a knowledge

extractor that can successfully reconstruct the audited blocks. Specifically, the extractor launches c different challenges against the same blocks $d_{i_j}(j = 1, 2, \ldots, c)$. With the input of the received integrity proofs, the extractor can first construct c independently linear equations with respect to $d_{i_j}(j = 1, 2, \ldots, c)$, and then extract $d_{i_j}(j = 1, 2, \ldots, c)$ by using these equations. To sum up, if the integrity proof can be accepted by the TPA, the cloud must truly possess user data. ■

Theorem 3 (Detectability): In the proposed protocol, if a user file on the cloud has n blocks, but m of them are damaged, the probability of detecting this abnormality is no less than $1-(n-m/n)^c$, where c is the number of challenge elements.

Proof: The damaged blocks can be detected only in the case that at least one of them is challenged by the TPA. Let m_c denote the number of damaged blocks that are challenged, and P_c denote the probability of detecting the data corruption. Then, we can have

$$P_c = P\{m_c \geq 1\} = 1 - \prod_{i=0}^{c-1} \frac{n-m-i}{n-i} \geq 1 - \left(\frac{n-m}{n}\right)^c.$$

If c is sufficiently large, P_c will be nonnegligible. The proof is completed. For example, if 100 of 1000 blocks are corrupted and ten blocks are challenged in each data auditing, then the probability of detecting this abnormality is at least 95.8%, which is satisfactory as expected. ■

Theorem 4 (Sensitive Information Hiding):

1) The TPA cannot learn users' sensitive information from the integrity proofs in the phase of data auditing;
2) Unauthorized users cannot obtain the data owners' sensitive information from cloud storage in the phase of data access.

Proof:

1) From the received integrity proof (μ, ν, R), the TPA can have the following observations.
   a) Due to the randomness of r_c, user blocks can hardly be reconstructed from $\mu = \sum_{j=1}^{c} s_j d_{i_j} + r_c$. Note that even though the TPA can repeatedly launch multiple challenge sequences against the same user blocks, still it can hardly solve these blocks. This is because r_c is regenerated in each challenge.
   b) Because of the hardness of the CDH problem, it is almost impossible to solve r_c from $R = e(u, g^{r_t})^{r_c}$.
   c) Since the values of $H(ID)^x$ and r_t are unknown, user blocks can also hardly be decoded from $\nu = \prod_{j=1}^{c} t_{i_j}^{s_j} = H(ID)^x \cdot (H_1(\text{name}||i) \cdot u^{d_i})^{r_t}$.

   Thus, users' sensitive information can be prevented from the TPA.
2) Obviously, unauthorized users cannot recover the data owners' sensitive blocks from the downloaded messy codes. In addition, unauthorized users also cannot directly download the sensitive blocks of the data owners through forging a warrant that can pass the verification of the cloud, due to the following reasons.
   a) To fulfill the first two requirements of the cloud, unauthorized user has to embed the identities and g^{r_s} in the right place of the warrant, where g^{r_s} is the public parameter of the targeted data owner.
   b) For the last requirement of the cloud, unauthorized user has to construct a valid warrant satisfying (4). However, if unauthorized users succeed in doing so, it means that r_s can be solved from g^{r_s}, which is contradictory to the hardness assumption of the CDH problem. That is, to say, the unauthorized user cannot cheat the cloud with the forged warrants.

Thereafter, users' sensitive information can also be protected from unauthorized users. ■

V. Performance Evaluation

In this section, we show the theoretical comparison between the proposed protocol and the previous [2] from the perspectives of storage, communication and computational costs. We also conduct intensive experiments to validate the advantages of our protocol.

A. Theoretical Comparison

We introduce some new notations for describing the operations in our protocol. Let $\mathcal{P}$ denote a bilinear pairing, $\mathcal{M}_1$ and $\mathcal{M}_2$ denote a multiplication in G_1 and G_2, $\mathcal{E}_1$ and $\mathcal{E}_2$ denote an exponentiation in G_1 and G_2, $\mathcal{A}_z$ denote an addition in Z_p^*, $\mathcal{S}_z$ denote a subtraction in Z_p^*, $\mathcal{M}_z$ denote a multiplication in Z_p^*, $\mathcal{H}$ denote a hash computation for different hash functions, k denote the number of sensitive data blocks, l denote the length of the file identifier, N_A and N_c denote the number of data access and integrity auditing, and U_A denote the number of authorized users. In addition, the length of an element in Z_n^*, Z_p^*, G_1, and G_2 is represented as $|n|$, $|p|$, $|g_1|$ and $|g_2|$, respectively.

Storage Cost: In our protocol, the storage cost is mainly determined by the outsourced data blocks and their authenticators. Thus, we concentrate on the storage cost incurred by these data. The size of the outsourced data blocks and their authenticators is both $n \cdot |p|$ bits. Thereafter, the total storage cost is about $2n \cdot |p|$ bits.

Communication Cost: According to the proposed protocol, the communication cost contains two parts, offline and online. The offline communication cost is mainly caused by outsourcing data blocks and their authenticators to the cloud, whose size is $2n \cdot |p|$ bits. The online communication cost is mainly incurred by data access and integrity auditing. For each access, the authorized user downloads data from the cloud, which incurs $n \cdot |p|$ communication cost. For each auditing, the TPA first transmits a challenge sequence $C = \{i_1, i_2, \ldots, i_c; H_t\}$ to the cloud, whose size is $c \cdot |n| + |p|$ bits; the cloud then feeds back an integrity proof $\{\mu, \nu, R\}$, whose size is $|p|+|g_1|+|g_2|$ bits. As a consequence, the total online communication cost is about $nN_A \cdot |p| + N_c \cdot (c \cdot |n| + 2 \cdot |p| + |g_1| + |g_2|)$ bits.

Computational Cost: In comparison to the other algorithms, the computational cost of system setup and key extraction is negligible and, thus, is omitted here. Before data outsourcing, the client has to compute the authenticators of file blocks, which requires $2n\mathcal{M}_1+2n\mathcal{E}_1+n\mathcal{H}$ computational cost. During data auditing, the TPA first generates a challenge sequence, which incurs a negligible computational cost; then, the cloud

TABLE II
PERFORMANCE COMPARISON BETWEEN THE PROPOSED PROTOCOL AND THE STATE-OF-THE-ART METHOD [2]

		The proposed protocol	The state-of-the-art [2]
Storage cost		$2n \cdot \|p\|$	$(2n+k) \cdot \|p\|$
Communication cost	Offline phase	$2n \cdot \|p\|$	$4n \cdot \|p\|$
	Online phase	$nN_A \cdot \|p\| + Nc \cdot (c \cdot \|n\| + 2 \cdot \|p\| + \|g_1\| + \|g_2\|)$	$2nN_A \cdot \|p\| + Nc \cdot (c \cdot \|n\| + 2 \cdot \|p\| + \|g_1\|)$
Computation cost	Authenticator generation	$2n\mathcal{M}_1 + 2n\mathcal{E}_1 + n\mathcal{H}$	$k(\mathcal{A}_z + \mathcal{S}_z) + (2n+k)\mathcal{M}_1 + (2n+k)\mathcal{E}_1 + n\mathcal{H}$
	Proof generation	$P + c\mathcal{A}_z + (c-1)\mathcal{M}_1 + c\mathcal{M}_z + c\mathcal{E}_1 + \mathcal{E}_2 + c\mathcal{H}$	$(c-1)\mathcal{A}_Z + (c-1)\mathcal{M}_1 + c\mathcal{M}_z + c\mathcal{E}_1 + c\mathcal{H}$
	Integrity verification	$3\mathcal{P} + (c-1)A_z + (c-1)\mathcal{M}_1 + 2\mathcal{M}_2 + (c+3)\mathcal{E}_1 + (c+1)\mathcal{H}$	$4\mathcal{P} + (c-1)\mathcal{A}_z + (c+l)\mathcal{M}_1 + 3\mathcal{M}_2 + (c+l+1)\mathcal{E}_1 + 2\mathcal{E}_2 + c\mathcal{H}$
	Access authorization	$U_A \cdot (2P + \mathcal{E}_1 + 2\mathcal{H})$	–
	Access revocation	$U_A \cdot (2P + \mathcal{E}_1 + 2\mathcal{H})$	–

calculates an integrity proof $\{\mu, \nu, R\}$, which needs $P + c\mathcal{A}_z + (c-1)\mathcal{M}_1 + c\mathcal{M}_z + c\mathcal{E}_1 + \mathcal{E}_2 + c\mathcal{H}$ computational cost; finally, the TPA checks the integrity of cloud storage, which requires $3\mathcal{P} + (c-1)A_z + (c-1)\mathcal{M}_1 + 2\mathcal{M}_2 + (c+3)\mathcal{E}_1 + (c+1)\mathcal{H}$ computational cost. To authorize a user, the data owner first generates a delegation, which incurs $\mathcal{E}_1 + \mathcal{H}$ computational cost; then the user to be authorized checks the correctness of the received delegation, which needs $2P + \mathcal{H}$ computational cost. Thus, the total time cost of access authorization is $U_A \cdot (2P + \mathcal{E}_1 + 2\mathcal{H})$. To revoke a user, the data owner updates the delegations of all nonrevoked users, which needs $U_A \cdot (2P + \mathcal{E}_1 + 2\mathcal{H})$ computational cost.

Comparison With the State-of-the-Art Method [2]: The recent work [2] is regarded as one of the most efficient protocols in the field of cloud storage auditing with sensitive information hiding. As shown in Table II, we have the following observations.

1) The proposed protocol has a lower storage cost than [2]. This is due to the fact that the sanitizer in [2] needs to store the sanitizing values of sensitive blocks, which is avoided in our proposed protocol.
2) The proposed protocol has a lower communication cost than [2]. The main reason is that during data outsourcing and data access, users' data in our protocol can be directly transferred between the users and the cloud without the relay of the sanitizer in [2].
3) Compared with [2], our proposed protocol improves the computational cost. During authenticator generation, our protocol has the computational cost $O(n)$, which is lower than [2] whose computational cost is $O(n+k)$. In proof generation and integrity verification, the complexity of these two protocols is both $O(c)$.

According to the above discussions, the proposed protocol can achieve sensitive information hiding in remote data auditing without the help of a sanitizer, which brings the following advantages. First, the expensive outlays in purchasing and maintaining a third-party server is avoided. Second, the storage, communication and computational costs are reduced. Furthermore, the proposed protocol allows data owners to enable or disable other users to access their sensitive data blocks, which can be applied to more application scenarios.

B. Experimental Results

In this section, we provide experimental results to validate the advantages of our protocol, implemented by using the development tool Eclipse with the JAVA Pairing-Based Cryptography Library [32]. The workstation is constructed on i5-7200U CPU and 8-GB RAM. The size of user files is set to 20 KB, consisting of 1000 blocks. The elliptic curve is chosen as $y^2 = x^3 + x$ with $|p| = 160$ bits and $|g| = 512$ bits. The default number of data blocks, the challenged blocks, the sensitive blocks, the data access and the integrity auditing is set to 1000, 200, 20, 1, and 1, respectively. In addition, each result is averaged by 100 runs.

Storage Cost: The storage cost is mainly determined by the size of user blocks and their authenticators. As depicted in Fig. 3(a), we can find that the storage costs of these two protocols both increase with the number of sensitive blocks as expected. The storage cost of the proposed protocol is much lower than [2], because [2] needs to store the sanitized values of sensitive blocks in addition to the indices of sensitive blocks.

Communication Cost: The communication cost is dominated by interactions among different entities in the protocols. As depicted in Fig. 3(b), the communication costs of these two protocols are both proportional to the number of data blocks, because the larger number of data blocks, the larger amount of data transmission during data outsourcing. The offline communication cost of the proposed protocol is about half of that of [2]. This is due to the fact that users in [2] have to transmit the data to the sanitizer at first and then the sanitizer forwards the sanitized data to the cloud, while users in the proposed protocol directly send the data to the cloud. The online communication cost of our protocol is also about half of that of [2]. This is because that the

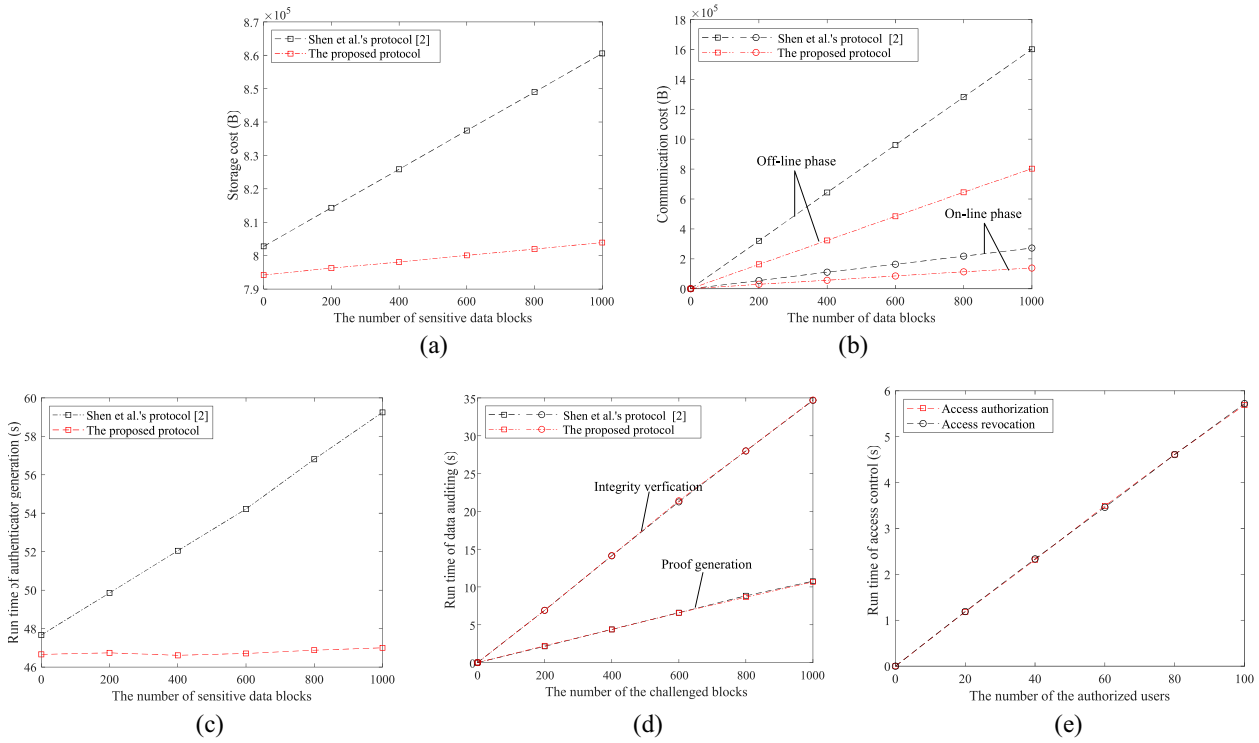

Fig. 3. Performance comparison between the proposed protocol and the state-of-the-art method [2]. (a) Storage cost. (b) Communication cost. (c) Computational cost of authenticator generation. (d) Computational cost of data auditing. (e) Computational cost of access control.

online communication cost is dominated by data access, especially in the application of cloud storage with large-scale user files.

Computational Cost: The computational cost is the highest during data outsourcing and integrity auditing. For data outsourcing, we provide the computational cost of authenticator generation in Fig. 3(c), which accounts for the most proportion in data outsourcing. It can be found that when the number of sensitive blocks varies from 0 to 1000, the computational cost of authentication generation is almost constant in the proposed protocol, while the computational cost increases in [2]. This is due to the fact that the recomputation on the authenticators of sensitive data blocks in [2] is eliminated in our protocol. The gap of the computational cost between these two protocols is proportional to the number of sensitive blocks in term of authentication generation, which is because that the more sensitive data blocks, the more data authenticators need to be recalculated in [2]. For data auditing, we focus on the computational costs of proof generation and integrity verification, as shown in Fig. 3(d), which takes up the overwhelming majority in data auditing. We can observe that no matter in proof generation or in integrity verification, the computational costs of these two protocols increase with the number of challenged blocks due to aggregation operations performed on the challenged data. These two protocols have similar computational costs in proof generation and integrity verification as expected.

Furthermore, we provide the computational cost of access control in Fig. 3(e). It can be observed that the computational costs of access authorization and revocation both increase with the number of authorized users. This is because authorized users all need to verify the correctness of delegations from the data owner. The proposed protocol can efficiently support access authorization and revocation. It takes little time to distribute the delegations to all authorized users. Note that [2] dose not support access control of sensitive information.

VI. Conclusion

In this article, we presented an ID-based cloud storage auditing protocol for data sharing with access control of sensitive information, which allows data owners to enable or disable other users to access their sensitive information. Our solution realizes the property of sensitive information hiding without the need of a third-party sanitizer, which can avoid extra outlays on purchasing and maintaining a server for sanitizing user data. To achieve this, the data owner first distributes delegations to other users for access authorization, and then the authorized users transmit warrants to the cloud, generated by using the received delegation. If this warrant is verified successfully, the cloud provides the actual sensitive blocks; otherwise, the cloud feeds back uniform messy codes. In addition, data owners in our solution can control the access of their sensitive information by updating the delegations of their authorized users. We validated the security and efficiency of the proposed protocol through comprehensive theoretical analysis and experimental results. In future work, it is interesting to realize cloud storage auditing with decentralized access control of sensitive information in order to avoid the credibility problem caused by centralized access control of the cloud or the sanitizer in [2].

REFERENCES

[1] *Top Threats to Cloud Computing: Egregious Eleven Deep Dive*, Cloud Security Alliance, Seattle, WA, USA, pp. 1–30, 2020.

[2] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling ID-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.

[3] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 1–25.

[4] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. ACM Conf. Comput. Commun. Security*, 2007, pp. 584–597.

[5] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Security*, 2008, pp. 90–107.

[6] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. Int. Conf. Security Privacy Commun. Netw.*, 2008, pp. 1–10.

[7] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. ACM Trans. Inf. Syst. Security*, vol. 17, 2015, pp. 1–29.

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

[9] K. Yang and X. Jia, "An efficient and secure dynamic auditing protocol for data storage in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 24, no. 9, pp. 1717–1726, Sep. 2013.

[10] F. Chen, T. Xiang, Y. Yang, and S. Chow, "Secure cloud storage meets with secure network coding," *IEEE Trans. Comput.*, vol. 12, no. 10, pp. 1936–1948, Oct. 2017.

[11] H. Tian *et al.*, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Trans. Services Comput.*, vol. 10, no. 5, pp. 701–714, Sep./Oct. 2017.

[12] J. Shen, J. Shen, X. Chen, X. Huang, and W. Susilo, "An efficient public auditing protocol with novel dynamic structure for cloud data," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

[13] H. Yan, J. Li, J. Han, and Y. Zhang, "A novel efficient remote data possession checking protocol in cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 1, pp. 78–88, Jan. 2017.

[14] C. Hahn, H. Kwon, D. Kim, and J. Hur, "Enabling fast public auditing and data dynamics in cloud services," *IEEE Trans. Services Comput.*, early access, Oct. 14, 2020, doi: 10.1109/TSC.2020.3030947.

[15] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.

[16] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.

[17] 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.

[18] F. Chen, T. Xiang, Y. Yang, C. Wang, and S. Zhang, "Secure cloud storage hits distributed string equality checking: More efficient, conceptually simpler, and provably secure," in *Proc. IEEE Conf. Comput. Commun.*, 2015, pp. 2389–2937.

[19] J. Zhang, Y. Yang, Y. Chen, and F. Chen, "A secure cloud storage system based on discrete logarithm problem," in *Proc. IEEE/ACM Int. Symp. Qual. Service*, 2017, pp. 1–10.

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

[21] H. Wang, D. He, and S. Tang, "ID-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.

[22] Y. Yu *et al.*, "ID-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.

[23] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous ID-based public provable data possession," *IEEE Trans. Service Comput.*, vol. 12, no. 5, pp. 824–835, Sep./Oct. 2019.

[24] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Service Comput.*, vol. 14, no. 1, pp. 71–81, Jan./Feb. 2021, doi: 10.1109/TSC.2018.2789893.

[25] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, "Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics," *IEEE Trans. Depend. Secure Comput.*, early access, Aug. 4, 2020, doi: 10.1109/TDSC.2020.3013927.

[26] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. Choo, "Fuzzy ID-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Depend. Secure Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2019.

[27] L. Huang *et al.*, "IPANM: Incentive public auditing scheme for non-manager groups in clouds," *IEEE Trans. Depend. Secure Comput.*, early access, Jun. 25, 2020, doi: 10.1109/TDSC.2020.3004827.

[28] 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.

[29] J. Yuan and S. Yu, "Public integrity auditing for dynamic data sharing with multiuser modification," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 8, pp. 1717–1726, Aug. 2015.

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

[31] Y. Zhang, J. Yu, R. Hao, C. Wang, and K. Ren, "Enabling efficient user revocation in ID-based cloud storage auditing for shared big data," *IEEE Trans. Depend. Secure Comput.*, vol. 17, no. 3, pp. 608–619, May/Jun. 2020.

[32] D. Angelo and I. Vincenzo. *jPBC: Java Pairing Based Cryptography*. Accessed: 2013. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/

Yang Yang received the Ph.D. degree in computer architecture from Wuhan University, Wuhan, China, in 2018.

He is currently an Assistant Professor with the Zhongnan University of Economics and Law, Wuhan. His research interests include cloud computing security and wireless physical communication.

Yanjiao Chen (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from Hong Kong University of Science and Technology, Hong Kong, in 2015.

She is currently a Bariren Researcher with Zhejiang University, Hangzhou, China. Her research interests include computer networks, wireless system security, cloud computing, and network economy.

Fei Chen received the Ph.D. degree in computer science and engineering from the Chinese University of Hong Kong, Hong Kong, in 2014.

He joined the College of Computer Science and Engineering, Shenzhen University, Shenzhen, China, as a Lecturer, in 2015. His research interests include information and network security and data protection and privacy.

Jing Chen (Member, IEEE) received the Ph.D. degree in computer science from Huazhong University of Science and Technology, Wuhan, China, in 2008.

He is currently a Full Professor with the Computer School, Wuhan University, Wuhan. He has published more than 80 research papers in many international journals and conferences, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON MOBILE COMPUTING, INFOCOM, SECON, and TrustCom. His research interests are in cloud security and network security.