Efficient Identity-Based Data Integrity Auditing with Key-Exposure Resistance for Cloud Storage

Wenting Shen, Jia Yu, Ming Yang, and Jiankun Hu, *Senior Member*, *IEEE*

***Abstract*—The key exposure is a serious threat for the security of data integrity auditing. Once the user's private key for auditing is exposed, most of the existing data integrity auditing schemes would inevitably become unable to work. To deal with this problem, we construct a novel and efficient identity-based data integrity auditing scheme with key-exposure resilience for cloud storage. This is achieved by designing a novel key update technique, which is fully compatible with BLS signature used in identity-based data integrity auditing. In our design, the Third Party Auditor (TPA) is responsible for generating update information. The user can update his private key based on the private key in one previous time period and the update information from the TPA. Furthermore, the proposed scheme supports real lazy update, which greatly improves the efficiency and the feasibility of key update. Meanwhile, the proposed scheme relies on identity-based cryptography, which makes certificate management easy. The security proof and the performance analysis demonstrate that the proposed scheme achieves desirable security and efficiency.**

***Index Terms*—Cloud storage, data integrity auditing, identity-based cryptography, key-exposure resistance, key update.**

I. Introduction

CLOUD storage services are being gradually accepted in recent years for its great advantages, such as low cost, flexibility and on-demand service. As a result, more and more enterprises and individuals are choosing cloud platform to maintain and store their data. Although cloud storage service is convenient for users, security risks should not be neglected [1]. One of the biggest concerns is the integrity of the outsourced data due to the inevitable operation errors or software/hardware failures in the cloud [2]. To ensure the integrity of the cloud data, a great deal of cloud data integrity auditing schemes have been proposed [3–6]. Generally, to release the burden of users, a trusted Third-Party Auditor (TPA) is delegated to undertake the data integrity auditing task [7–9].

In most of existing data integrity auditing schemes [10–14], a pair of public key and private key needs to be generated for the user. The public key is used to verify the validity of the proof generated by the cloud. The private key is only utilized to calculate the authenticators for data blocks. The authenticators are used to verify whether the cloud correctly stores the user's data in the phase of data auditing. These schemes are based on traditional Public Key Infrastructure (PKI) setting, which uses a digital certificate to ensure the authenticity of the user's public key. As a result, it incurs the considerable overheads since certificate generation, certificate revocation and certificate renewal are complicated and time-consuming. A feasible solution for simplifying the certificate management is identity-based cryptography. In the identity-based cryptography, the user's private key is generated by a trusted Private Key Generator (PKG) based on the user's identity [15]. The public key is replaced with the user's identity information (e.g. user name, E-mail address, and employee number), which removes the use of certificate [16, 17]. Based on the identity-based cryptography, Wang et al. [18] constructed the first identity-based data integrity auditing scheme. Following that, Wang et al. [19] designed an identity-based proxy-oriented data integrity auditing scheme, in which the authenticators are calculated with the help of the proxy. Zhang et al. [20] constructed an identity-based shared data integrity auditing scheme with efficient user revocation. The above identity-based data integrity auditing schemes all use BLS signature to construct the authenticators of data blocks for supporting the data integrity auditing.

The key exposure is a serious security issue for data integrity auditing. Once the user's private key for data integrity auditing is exposed to the cloud, the cloud is able to discard the data rarely accessed to save the storage space, or hide the incidents of data loss to maintain his reputation by forging the valid authenticators with the user's private key. Consequently, it will make the data integrity auditing unable to work correctly any more. To deal with this problem, Yu et al. [21–23] and Xu et al. [24] designed the data integrity auditing schemes with key-exposure resilient in the PKI settings. They make use of different key update techniques to update the user's private key. However, all these key update techniques are incompatible with BLS signature. It means that these key update techniques cannot be directly applied to the existing identity-based data integrity auditing schemes [18, 20, 25–27].

Contribution. The contributions are summarized as follows:

This research is supported by National Natural Science Foundation of China (62102211), Shandong Provincial Natural Science Foundation, China (ZR2021QF018), The Open Research Fund from Shandong Key Laboratory of Computer Network (SDKLCN-2020-02). (*Corresponding author: J. Yu*)

W. Shen is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China, with Shandong Key Laboratory of Computer Networks, Jinan 250014, China. E-mail: shenwentingmath@163.com.

J. Yu is with the College of Computer Science and Technology, Qingdao University, Qingdao 266071, China, with State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China. E-mail: qduyujia@gmail.com.

M. Yang is with Qilu University of Technology (Shandong Academy of Sciences), Jinan, 250041, China, with Shandong Computer Science Center, Shandong Key Laboratory of Computer Networks, Jinan, 250041, China. E-mail: yangm@sdas.org.

J. Hu is with the Chair Professor and Research Director of Cyber Security Lab, School of Engineering and IT, University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA), Canberra, Australia. E-mail: J.Hu@adfa.edu.au.

We propose a novel identity-based data integrity auditing scheme with key-exposure resilience for cloud storage. In the proposed scheme, we design a novel key update technique, which is fully compatible with BLS signature in identity-based data integrity auditing. The time-consuming computational operation for key updates is performed by the Third Party Auditor (TPA). Specifically, the TPA calculates the update information with its secret key in each time period, and sends the update information to the user. The user can check the validity of the update information and efficiently update his private key based on the private key in one previous time period and the update information from the TPA. Even if the malicious cloud obtains the user's private key in a certain time period, the auditing tasks in other time periods are still able to be well performed. When the user's private key is updated frequently, the user only needs to perform lightweight computations to calculate the private key for the current time period. In addition, our scheme supports real lazy update, i.e., the user updates his private key only when he uploads the file to the cloud rather than updating his private key in each time period. It greatly improves the efficiency and the feasibility of key update. Our scheme relies on the identity-based cryptography, which eliminates the complicated certificate management in traditional PKI systems. We prove the security of the proposed scheme and give a comprehensive performance analysis.

A. Related Work

In 2007, "Provable Data Possession" (PDP) was introduced by Ateniese et al. [28], which allows the cloud to convince the verifiers that it stores the data correctly. Juels and Kaliski [29] presented "Proof of Retrievability" (PoR) and constructed a publicly verifiable PoR scheme. In this scheme, data are encoded by error correcting codes and several "sentinels" are embedded in the file. The verifier is capable of verifying the correctness of data by checking whether the sentinels at some specific positions exist or not. In 2008, Shacham and Waters [30] constructed two improved PoR schemes by utilizing BLS signature and pseudorandom function respectively.

To support data dynamics, the first provable data possession scheme with partial data dynamics was proposed by Ateniese et al. [31]. After that, plenty of data integrity auditing schemes focus on supporting full data dynamics [32–35]. To achieve data privacy protection, Wang et al. [36] proposed a data integrity checking scheme, in which the cloud utilizes a random value to produce auditing proof. Based on zero-knowledge proof of discrete logarithm, Li et al. [37] designed a cloud storage checking scheme supporting data privacy protection. Ding [38] employs edge server to help users to calculate authenticators, which alleviates the user's computation burden. Based on online/offline signature, Li et al. [3] presented a lightweight data integrity checking scheme for the user with low computation capability. Wang et al. [39] took the problem of user identity privacy into account and proposed a shared data integrity checking scheme with identity privacy protection by employing ring signature. To improve the cloud's storage efficiency, Xu et al. [26] proposed a blockchain-enabled deduplicatable data integrity auditing scheme. Shen et al. [40] designed a public auditing scheme with efficient data ownership transfer. Zhou et al. [41] proposed a multicopy data integrity auditing scheme, in which the improved Merkle hash tree is used to achieve multicopy dynamic operations.

The problem of complicated certificate management in the data integrity auditing has been researched. In [18], Wang et al. designed a cloud storage auditing scheme, which is built on identity-based cryptography. Zhang et al. [20] considered the problem of user revocation and presented a shared data integrity auditing scheme with efficient user revocation by updating non-revoked group users' private keys. Wang et al. [9] constructed an identity-based data integrity checking scheme, which achieves both unconditional anonymity and incentive. Zhang et al. [16] designed a data integrity auditing scheme with conditional identity privacy protection. Wang et al. [42] proposed an identity-based data outsourcing scheme with comprehensive auditing, in which the proxy generates data authenticators on behalf of the user. Based on the identity-based cryptography, Shen et al. [27] designed a shared data integrity checking scheme, in which the sensitive information is sanitized by the sanitizer.

All aforementioned schemes are designed on the assumption that the user's private key for data integrity auditing is absolutely secure and cannot be exposed. Yu et al. [21] firstly discussed the problem of key exposure and presented a data integrity auditing scheme with key-exposure resilience by updating the user's private key. Subsequently, Yu et al. [22] proposed a data integrity checking scheme with verifiable outsourcing of key updates. In this scheme, the task of updating private key is executed by the TPA. In [23], a strong key-exposure resilient data integrity auditing scheme is proposed, in which the cloud cannot obtain the user's private key in unexposed time periods. Xu et al. [24] designed an intrusion-resilient data integrity checking scheme, which mitigates the risk of key exposure. In this scheme, a full binary tree is utilized to update private key. Other key-exposure resilient data integrity auditing schemes [43], [44] have been proposed. Nonetheless, these schemes either introduce an additional key update server or are based on time-consuming lattice cryptography.

B. Organization

In Section II, we first introduce notions and preliminaries used in the paper. After illustrating the system model and security model (Section III), we give the identity-based data integrity auditing scheme with key-exposure resistance in Section IV. The security analysis and the performance analysis are discussed in Section V and Section VI respectively. Finally, in Section VII, the conclusion is made.

II. Notations and Preliminaries

In this section, we briefly review bilinear maps, discrete logarithm (DL) problem and computational Diffie-Hellman (CDH) problem.

A. Notations

We present some notations used in this paper in Table I.

TABLE I
NOTATIONS

Notation	Meaning
p	One large prime
G, G_T	Multiplicative cyclic groups with prime order p
e	A bilinear pairing map $e : G \times G \rightarrow G_T$
g, u	Two generators of group G
H_1, H_2	Two cryptographic hash function, $H_1 : \{0,1\}^* \rightarrow G$ and $H_2 : G \times \{0,1\}^* \rightarrow Z_p^*$
l	The security parameter
n	The number of data blocks of the file F
ID	The user's identity
$SK_{TPA,ID}$	The TPA's secret key corresponding to the user U_{ID}
$sk_{ID,0}$	The initial private key of the user U_{ID}
N	The maxinum lifetime of all cloud files
t	The threshold value
$x^*_{ID,j}$	The update information of the user U_{ID} in the time period j
$sk_{ID,j}$	The private key of the user U_{ID} in the time period j
$F = \{m_1, m_2, ..., m_n\}$	The cloud file F composed by blocks $m_i (i \in [1, n])$
f_{id}	The file identifier
$\Phi = \{\beta_1, \beta_2, ..., \beta_n\}$	The authenticator set
τ	The file tag
S	The challenged block set
c	The number of challenged blocks
$Chal = \{i, v_i\}_{i \in S}$	The auditing challenge
$Proof = \{j, \mu, \eta\}$	The auditing proof

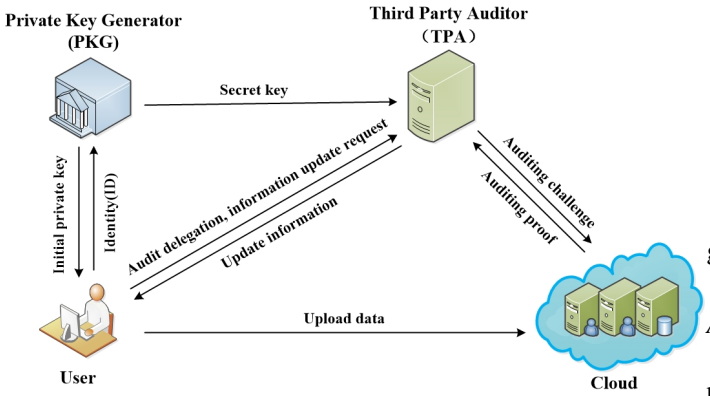

Fig. 1. System model

B. Preliminaries

1) **Bilinear Maps**

Let G and G_T be two multiplicative cyclic groups of prime order p. A bilinear map e is a map $e : G \times G \rightarrow G_T$ which satisfies [45]:

a) *Bilinearity*: for all $g_1, g_2 \in G$ and $a, b \in Z_p^*$, $e\left(g_1^a, g_2^b\right) = e\left(g_1, g_2\right)^{ab}$.

b) *Non-degeneracy*: $e\left(g, g\right) \neq 1$, where g is a generator of G.

c) *Computability*: there is an efficient algorithm to calculate map $e : G \times G \rightarrow G_T$.

2) **Discrete Logarithm (DL) Problem**

Given g, $g^x \in G$, where $x \in Z_p^*$ is unknown, generate x. The DL assumption in G holds if it is computationally infeasible to solve the DL problem in G [46].

3) **Computational Diffie-Hellman (CDH) Problem**

Given g, g^x and $g^y \in G$, where $x, y \in Z_p^*$ are unknown, generate $g^{xy} \in G$. The CDH assumption in G holds if it is computationally infeasible to solve the CDH problem in G [47].

III. System Model and Security Model

In this section, we present the system model, the design goal, the underlying algorithms and the security model.

A. System Model

As illustrated in Fig.1, the system model of an identity-based data integrity auditing scheme with key-exposure resistance comprises four types of entities: the user, the cloud, the Third Party Auditor (TPA) and the Private Key Generator (PKG).

1) User: The user has a large number of files to upload to the cloud.
2) Cloud: The cloud has enormous storage space and computation resources, and provides data storage services for the user.
3) TPA: The TPA is an entity with abundant computation resources. It takes charge of two important tasks. One is to verify whether the cloud data is stored intactly. The other is to generate the update information used to update the private keys for the user in different time periods.
4) PKG: The PKG is responsible for generating system public parameters, the user's initial private key and the TPA's secret key.

At the beginning of each time period, the TPA computes and sends the update information to the user, where the update information is used to update the user's private key for data integrity auditing. The user calculates the private key for the

current time period based on the update information and the private key in one previous time period, and then generates data authenticators with the private key in the current time period. The user sends the data blocks and the corresponding authenticators to the cloud.

The TPA is delegated by the user to complete the task of data integrity auditing. Firstly, a challenge is outputted by the TPA. Then the TPA sends the challenge to the cloud. Upon receiving the challenge, the cloud returns a corresponding proof to the TPA. The TPA verifies whether the cloud data are kept unchanged or not by checking the validity of proof.

In this paper, we mainly focus on solving the problem of key exposure in data integrity auditing, and do not consider the problem of data privacy protection. In addition, we also do not consider the threat that the TPA is curious about the user's private key used to generate the authenticators. Actually, all existing data integrity auditing schemes with key-exposure resistance do not consider this threat. The detail explanation is described in *Remark 1*.

Remark 1: The basic assumption of most data integrity auditing schemes is that the TPA honestly returns a correct auditing result to the user. The user's private key is only used to generate the authenticators for realizing data integrity verification from the TPA. If the TPA is able to obtain an exposed private key and uncover the user's private keys, it will not forge the authenticators with these private keys to pass its own verification, which is fully unnecessary and unreasonable. In other words, the user's private keys are useless for the TPA. Therefore, the TPA has no incentive to obtain the user's private key. As a result, similar to the schemes [22], [23], we select the TPA to execute the tasks of key update information generation and data integrity auditing.

B. Design Goals

To achieve key-exposure resistance in identity-based data integrity auditing, the following goals should be achieved:

1) *The correctness*:
   a) *The initial private key correctness*: to guarantee that the initial private key can pass the user's checking only if the PKG produces the initial private key honestly.
   b) *The update information correctness*: to guarantee that the update information can pass the user's checking only if the TPA produces the update information correctly.
   c) *Auditing correctness*: to assure that the proof can pass the TPA's checking only if the cloud executes the auditing task honestly.
2) *Key-exposure resistance*: to ensure that even if the private keys in *t* time periods are exposed, it does not affect the security of the private keys in other time periods.
3) *Auditing soundness*: to guarantee that if the cloud does not actually keep the user's entire data, it cannot pass the the TPA's checking.
4) *Efficient key update*: to ensure low overhead of private key update.

C. Underlying Algorithms

Definition 1: An identity-based data integrity auditing scheme with key-exposure resistance for cloud storage includes the following eight algorithms:

1) $Setup(1^l)$: The setup algorithm is executed by the PKG. Input the security parameter l, it outputs the master secret key x_0, the system public parameters $param$. The PKG keeps the master secret key in secret.
2) $Extract(x_0, ID, param)$: The extraction algorithm is completed by the PKG. With the master secret key x_0, the user's identity ID and the system public parameters $param$ as input, it generates the user's initial private key $sk_{ID,0}$ and the TPA's secret key $SK_{TPA,ID}$ corresponding to the user U_{ID}. The user U_{ID} can check whether $sk_{ID,0}$ is valid or not and accepts it as his initial private key only if it passes the checking.
3) $UpIGen(param, j, j-1, SK_{TPA,ID})$: The update information algorithm is handled by the TPA. With the public parameters $param$, the current time period j, the previous time period $j-1$ and the TPA's secret key $SK_{TPA,ID}$ as input, it outputs the update information $x^*_{ID,j}$ used to update the user U_{ID}'s private key in the time period j. The user can check the validity of $x^*_{ID,j}$.
4) $KeyUpdate(param, j, ID, x^*_{ID,j}, sk_{ID,j-1})$: The key update algorithm is carried out by the user U_{ID}. With the public parameters $param$, the current time period j, the user identity ID, the corresponding update information $x^*_{ID,j}$ and the private key $sk_{ID,j-1}$ in the previous time period $j-1$ as input, it generates the private key $sk_{ID,j}$ in the current time period j.
5) $AuthGen(param, F, j, sk_{ID,j}, f_{id}, ssk)$: The authenticator generation algorithm is completed by the user U_{ID}. With the public parameters $param$, the current time period j, the file F, the user U_{ID}'s private key $sk_{ID,j}$, the file identifier f_{id} and the user's signing private key ssk as input, it generates an authenticator set Φ and a file tag τ.
6) $Challenge(param, c, ID, f_{id}, j)$: The challenge generation algorithm is run by the TPA. With the public parameters $param$, the number of challenged blocks c, the user identity ID, the file identifier f_{id} and the current time period j as input, it outputs the challenge $chal$ for the file f_{id} of the user U_{ID} in the current time period j.
7) $ProofGen(param, j, \tau, F, \Phi, chal)$: The proof generation algorithm is handle by the cloud. With the public parameters $param$, the current time period j, the file tag τ, the file F, the authenticator set Φ and the auditing challenge $chal$ as input, it generates an auditing proof $proof$.
8) $ProofVerify(chal, param, proof, \tau, j, ID, f_{id})$: The proof verification algorithm is executed by the TPA. With the challenge $chal$, the system public parameters $param$, the proof $proof$, the file tag τ, the current time period j, the user identity ID, and the file identifier f_{id} as input, and outputs "1" if $proof$ is a valid proof; or "0", otherwise.

D. Security Model

Our security model considers key-exposure resistance and audtiting soundness. In this security model, the user plays the role of the challenger and the malicious cloud is viewed as the adversary. Assume the adversary is not able to query the private keys of the same user in more than t time periods. We define a game between the challenger and the adversary to present how the adversary attacks the securtiy of an identity-based data integrity auditing scheme with key-exposure resistance. This game consists of the following phases:

1) **Setup phase**. The challenger executes the *Setup* algorithm to obtain the master secret key x_0 and the system public parameters $param$, then forwards $param$ to the adversary and holds x_0. Set time period $j = 0$.
2) **Query phase**. In this phase, the adversary makes the following two queries to the challenger.
   a) *Private key Queries*: The adversary can issue the query of the private key for any identity ID in time period j. Moreover, for the same user U_{ID}, the adversary cannot query his private keys in more than t time periods. The challenger executes the $Extract$, $UpMGen$ and $KeyUpdate$ algorithms to calculate the private key $sk_{ID,j}$ in time period j, and sends $sk_{ID,j}$ to the adversary.
   b) *Authenticator Queries*: The adversary can make queries for the data authenticators of the file F under the identity ID in time period j. The challenger calculates the corresponding authenticators for the file F by running the $AuthGen$ algorithm, and returns these authenticators to the adversary. The adversary stores the file F and the corresponding authenticators. Set time period $j = j + 1$.

   At the end of each time period, the adversary is able to choose to stay in query phase or move on to the next phase.
3) **Challenge phase**. The challenger chooses a time period j^*, a file identifier ${f_{id}}^*$ and a user identity ID^*. In time period j^*, ID^* must does not appear in $Private\ key$ queries. The challenger sends a challenge $chal = \{i, v_i\}_{i \in S}$ to the adversary, where $S \in \{\gamma_1, \gamma_2, ..., \gamma_c\}$ ($\gamma_\alpha \in [1, n]$, $\alpha \in [1, c]$ and $c \in [1, n]$).
4) **Forgery phase**. The adversary produces a data integrity proof $proof$ corresponding to the challenge *chal* in time period j^*, and sends *proof* to the challenger. If $ProofVerify(chal, param, proof, \tau, j^*, ID^*, {f_{id}}^*) =$ "1", then the adversary succeeds in the above game.

This security model describes that the adversary cannot query the private keys of the same user in more than t time periods and also cannot query the private key of the challenged user in the challenged time period, but can query the data authenticators for any file in each time period. If the adversary does not correctly store the user U_{ID^*}'s all challenged data blocks for a time period in which the private key is not exposed, and cannot guess all bad blocks, he is unable to forge a valid proof to pass the verification of challenger. The adversary aims at passing the challenger's verification by outputting a valid proof for the challenged data blocks in the time period j^*. In the time period j^*, the user U_{ID^*}'s private key is not exposed. The following definition shows that there exists a knowledge extractor that can extract all challenged data blocks whenever the above adversary is capable of generating a valid proof $proof$ in the time period j^*.

Definition 2: We say an identity-based data integrity auditing scheme with key-exposure resistance is secure if the following condition holds: whenever an adversary in the above described game can pass the challenger's verification by outputting a valid proof *proof* with non-negligible probability, there is a knowledge extractor that can extract the challenged data blocks with non-negligible probability.

Definition 3: An identity-based data integrity auditing scheme with key-exposure resistance is $(\rho, \delta)(0 < \rho, \delta < 1)$ detectable if ρ fractions of the whole file are corrupted by the cloud, the probability that these corrupted data blocks are detected is no less than δ.

IV. The Proposed Scheme

A. Construction of Our Proposal

The cloud file F is divided into n data blocks, which is denoted as $F = (m_1, m_2, ..., m_n)(m_i \in Z_p^*)$. N is the maximum lifetime of all cloud files. In previous identity-based data integrity auditing schemes [20],[27], an identity-based signature Sig is employed to ensure the validity of the file identifier. Similarly, a similar identity-based signature Sig is utilized in the proposed scheme to guarantee the validity of the verification value, the file identifier f_{id} and the time period. Assume the user has held the signing private key ssk corresponding to the signature Sig. Such an assumption makes the description of the proposed scheme more clear and simple. Fig. 2 shows the workflow of private key extraction. The workflow of update information generation, key update, authenticator generation, and data integrity auditing are shown in Fig.3.

The detailed algorithms are presented below.

1) **Setup**(1^l)

   The PKG produces the master secret key, the TPA's secret key and the system public parameters.
   a) The PKG picks two multiplicative cyclic groups G and G_T of prime order p, two random generators g and u of G. The PKG also selects two different cryptographic hash functions $H_1 : \{0,1\}^* \rightarrow G$ and $H_2 : G \times \{0,1\}^* \rightarrow Z_p^*$ and a bilinear map $e : G \times G \rightarrow G_T$.
   b) The PKG picks a random value $x_0 \in Z_p^*$ as the master secret key and computes $Y_0 = g^{x_0}$ as the master public key.
   c) The PKG publishes system parameters $param = (G, G_T, p, u, g, e, H_1, H_2, Y_0)$.
2) **Extract**$(x_0, ID, param)$

   The PKG calculates and sends the initial private key to the user U_{ID}. The user U_{ID} is able to check whether the initial private key he received is valid or not. Furthermore, the PKG computes the TPA's secret key corresponding to the user U_{ID}.

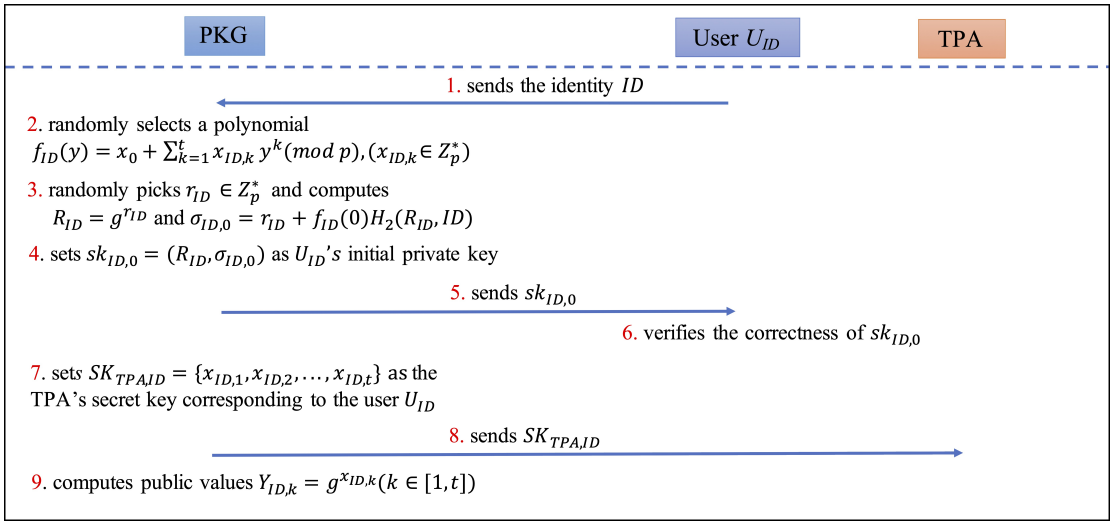

Fig. 2. The workflow of private key extraction

a) For the user U_{ID}, the PKG randomly selects a polynomial $f_{ID}(y) = x_0 + \sum_{k=1}^{t} x_{ID,k} y^k \pmod p (x_{ID,k} \in Z_p^*)$, where t is the threshold on the number of a user's private keys that are allowed to be compromised. Generally speaking, t is much smaller than N.

b) The PKG randomly picks $r_{ID} \in Z_p^*$, and computes $R_{ID} = g^{r_{ID}}$ and $\sigma_{ID,0} = r_{ID} + f_{ID}(0)H_2(R_{ID}, ID) \pmod p$ according to the user's identity ID, where $f_{ID}(0) = x_0$. And then the PKG sets $sk_{ID,0} = (R_{ID}, \sigma_{ID,0})$ as the user U_{ID}'s initial private key, and sends it to the user U_{ID}.

c) The user U_{ID} accepts the initial private key $sk_{ID,0}$ if $sk_{ID,0}$ can pass the verification of the following equation.

$$g^{\sigma_{ID,0}} = R_{ID} \cdot Y_0^{H_2(R_{ID}, ID)} \tag{1}$$

d) The PKG sets $SK_{TPA,ID} = \{x_{ID,1}, x_{ID,2}, ..., x_{ID,t}\}$ as the TPA's secret key corresponding to the user U_{ID}. The PKG sends $SK_{TPA,ID}$ to the TPA and computes public values $Y_{ID,k} = g^{x_{ID,k}} (k \in [1,t])$.

3) **UpIGen**$(param, j, j-1, SK_{TPA,ID})$

At the beginning of the time period j $(1 \le j \le N)$, the TPA computes and sends the update information to the user U_{ID}. The update information is used to update the user U_{ID}'s private key. The user U_{ID} checks the validity of the update information.

a) At the beginning of the time period j $(1 \le j \le N)$, the TPA computes the update information $x_{ID,j}^* = \sum_{k=1}^{t} x_{ID,k}(j^k - (j-1)^k) (1 \le j \le N)$ for the time period j with the secret key $SK_{TPA,ID} = \{x_{ID,1}, x_{ID,2}, ..., x_{ID,t}\}$ corresponding to the user U_{ID}. The update information $x_{ID,j}^*$ is used to update the user U_{ID}'s private key in the time period j. Note that $x_{ID,j}^* = f_{ID}(j) - f_{ID}(j-1)$. And then the TPA sends the update information $x_{ID,j}^*$ to the user U_{ID}.

b) The user U_{ID} checks whether the update information $x_{ID,j}^*$ is valid by the following equation

$$g^{x_{ID,j}^*} = \prod_{k=1}^{t} Y_{ID,k}^{\,j^k - (j-1)^k} \tag{2}$$

If the equation (2) holds, it means that the update information generated by the TPA is valid. Then, the user U_{ID} executes the following *KeyUpdate* algorithm.

4) **KeyUpdate**$(param, j, ID, x_{ID,j}^*, sk_{ID,j-1})$

The user U_{ID} calculates the private key in the current time period using the update information he received in *UpIGen* algorithm and the private key in one previous time period.

The user U_{ID} computes $\sigma_{ID,j} = \sigma_{ID,j-1} + x_{ID,j}^* \cdot H_2(R_{ID}, ID)$ using the update information $x_{ID,j}^*$ and the private key $\sigma_{ID,j-1}$ in one previous time period $j-1$, and sets $sk_{ID,j} = (R_{ID}, \sigma_{ID,j})$ as the private key in the current time period j. Note that $\sigma_{ID,j} = r_{ID} + f_{ID}(j) \cdot H_2(R_{ID}, ID)$.

Remark 2: Our scheme supports real lazy update. The user updates his private key only when he uploads the file to the cloud. It means that the user sends a key update request to the TPA for obtaining the update information only when there is a file that needs to be stored to the cloud. Assume that the user U_{ID} uploads the file in the time period l, and does not upload the file in the time periods from j to $l-1$. The user U_{ID} only needs to update his private key in the time period l with one single step rather than do multiple updates from the time period j to the time period l.

Specifically, at the beginning of the time period l, the user U_{ID} sends a key update request to the TPA for obtaining the update information to update his private key. Upon receiving the update request, the TPA executes the *UpIGen* algorithm to generate the update information $x_{ID,j,l}^* = \sum_{k=1}^{t} x_{ID,k}(l^k - j^k)$, then delivers it to the user U_{ID}. The user U_{ID} can verify the correctness of $x_{ID,j,l}^*$ by the equation (2) in *UpIGen* algorithm. If $x_{ID,j,l}^*$ passes the verification, the user computes $\sigma_{ID,l} =$

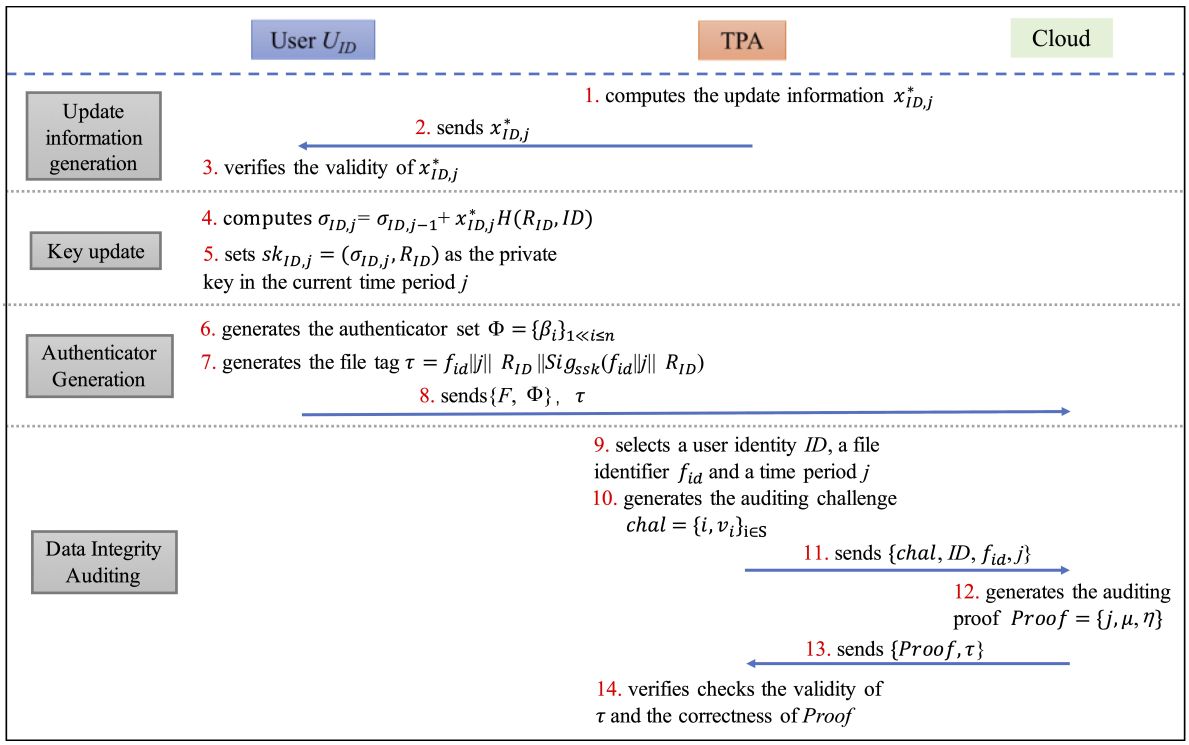

Fig. 3. The workflow of update information generation, key update, authenticator generation, and data integrity auditing

$\sigma_{ID,j} + x^*_{ID,j,l} \cdot H_2(R_{ID}, ID)$ based on the update information $x^*_{ID,j,l}$ and the private key $\sigma_{ID,j}$ in the previous time period j. Note that $\sigma_{ID,l} = \sigma_{ID,j} + (f_{ID}(l) - f_{ID}(j)) \cdot H_2(R_{ID}, ID) = r_{ID} + f_{ID}(l) \cdot H_2(R_{ID}, ID)$. The private key $sk_{ID,l}$ in the time period l is set as $sk_{ID,l} = (\sigma_{ID,l}, R_{ID})$.

Remark 3: In our scheme, r_{ID} can be the same for the same user in different time periods, for the following reasons. The user U_{ID}'s private key in the time period j is $sk_{ID,j} = (\sigma_{ID,j}, R_{ID})$, where $\sigma_{ID,j} = \sigma_{ID,j-1} + x^*_{ID,j} \cdot H_2(R_{ID}, ID) = r_{ID} + f_{ID}(j) \cdot H_2(R_{ID}, ID)$. $f_{ID}(j) = x_0 + \sum_{k=1}^{t} x_{ID,k} j^k \pmod p (x_{ID,k} \in Z_p^*)$ is the value computed by the PKG based on the PKG's master secret key x_0, the TPA's secret key $SK_{TPA,ID} = \{x_{ID,1}, x_{ID,2}, ..., x_{ID,t}\}$ corresponding to the user U_{ID} and the time period j. $f_{ID}(j)$ changes in every time period. The adversary cannot compute $f_{ID}(j)$ because he cannot obtain the PKG's master secret key x_0 and the TPA's secret key $SK_{TPA,ID}$. Similarly, the adversary cannot compute $f_{ID}(j) (j \in [1, N])$ in the whole time periods. In other words, the adversary cannot obtain r_{ID} even if he is able to obtain the user U_{ID}'s private keys in t time periods. In the phase of key update, the adversary cannot deduce the unexposed private keys without r_{ID} and $f_{ID}(j) (j \in [1, N])$ even with up to t exposed private keys. As a result, the key update is secure even if r_{ID} is the same for the same user in different time periods.

Remark 4: In our scheme, the user's private key in the current time period is generated based on the private key in one previous time period and the update information from the TPA. If the TPA obtains the user's private key in a certain time period by compromising the hardware token that stores the user's private key, then it can inevitably uncover the user's private key in other time periods since it also knows the update information. It seems that there is no feasible technique to solve the problem that the TPA can uncover the user's private keys after obtaining an exposed private key. Actually, this is not a real threat in data integrity auditing with key-exposure resistance.

5) **AuthGen**$(param, F, j, sk_{ID,j}, f_{id}, ssk)$

The user U_{ID} computes the authenticators for the file F with the private key in the current time period, and calculates the file tag used to guarantee the validity of the file identifier, the time period and the verification value. Then, the user U_{ID} uploads the file F along with the authenticator set and the file tag to the cloud.

a) For each block $m_i \in Z_p^* (i \in [1, n])$ of the file F, the user U_{ID} computes the authenticator β_i using the private key $sk_{ID,j}$ in the current time period j as follows: $\beta_i = (H_1(f_{id}||i||j) \cdot u^{m_i})^{\sigma_{ID,j}}$, where $f_{id} \in Z_p^*$ is the file identifier. Let $\Phi = \{\beta_i\}_{1\leq i\leq n}$ be the set of authenticators in the time period j.

b) The user U_{ID} generates the file tag $\tau = f_{id}||j||R_{ID}||Sig_{ssk}(f_{id}||j||R_{ID})$ with the signing private key ssk.

c) The user U_{ID} uploads $\{F, \Phi\}$ along with the file tag τ to the cloud.

6) **Challenge**$(param, c, ID, f_{id}, j)$:

The TPA selects a user identity ID, a file identifier f_{id} and a time period j. Then the TPA outputs an auditing challenge $chal = \{i, v_i\}_{i\in S}$, where $S \in [1, n]$ is a subset

including c elements and v_i is a random value in Z_p^*. The TPA sends $\{chal, ID, f_{id}, j\}$ to the cloud.

7) **ProofGen**$(param, j, \tau, F, \Phi, chal)$
Upon receiving $chal$, the cloud outputs an auditing proof $proof = \{j, \mu, \eta\}$ to the TPA, where $\mu = \sum_{i\in S} m_i v_i$ and $\eta = \prod_{i\in S} \beta_i^{v_i}$. The cloud sends the proof $proof$ along with the file tag τ to the TPA.
8) **ProofVerify**$(chal, param, proof, \tau, j, ID, f_{id})$
After receiving $proof$, the TPA checks the validity of file tag τ by verifying whether $Sig_{ssk}(f_{id}||j||R_{ID})$ is a valid signature or not. If the file tag τ is valid, the TPA retrieves the file identifier f_{id}, the time period j and the verification value R_{ID}. Then, the TPA checks the validity of the proof $proof$ via the following equation:

$$e(\eta, g) = e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\mu}, R_{ID} \cdot (Y_0 \prod_{k=1}^{t} {Y_{ID,k}}^{j^k})^{H_2(R_{ID},ID)}) \quad (3)$$

The TPA outputs "1" if the equation (3) holds; otherwise, outputs "0".

B. Discussion

In most of identity-based key-exposure resilient schemes [48], [49], [50], [51], a physically-secure helper is introduced to generate key update information. The user can update his private key with the key update information. In identity-based cryptography [15],[19], [25], the PKG is responsible for generating the user's initial private key.

To achieve the high-security level, both the helper and the PKG can replace the TPA to perform the key update information generation task in the proposed scheme. However, introducing the helper will increase the complexity of our system. If the PKG is in charge of generating key update information, he must always be online. In general, the PKG goes offline after generating the initial private key for the user. Thus, we still select the TPA to generate key update information for the user.

V. Security Analysis

The following analysis indicates that the proposed scheme is secure from the perspective of correctness, key-exposure resistance, detectability and key update security.

Theorem 1: (**Correctness**). A valid identity-based data integrity auditing scheme with key-exposure resistance meets the following properties:

1) (Initial private key correctness) The initial private key can pass the checking if the PKG produces the initial private key honestly.
2) (Update information correctness) The update information can pass the checking if the TPA produces the update information correctly.
3) (Auditing correctness) The proof can pass the checking if the cloud executes the auditing task honestly.

Proof.

1) Given the initial private key $sk_{ID,0} = (R_{ID}, \sigma_{ID,0})$ generated by the PKG, the validity of $sk_{ID,0}$ can be checked by the user under the equation (1). The validity of the initial private key in equation (1) is presented as follows:

$$\begin{aligned} g^{\sigma_{ID,0}} &= g^{r_{ID}+f_{ID}(0)H_2(R_{ID},ID)} \\ &= g^{r_{ID}} g^{x_0 H_2(R_{ID},ID)} \\ &= R_{ID} \cdot {Y_0}^{H_2(R_{ID},ID)} \end{aligned}$$

2) Given the update information $x_{ID,j}^*$ from the TPA, the user can check the correctness of $x_{ID,j}^*$ based on the verification equation (2). The validity of the update information in equation (2) is presented as follows:

$$\begin{aligned} g^{x_{ID,j}^*} &= g^{\sum_{k=1}^{t} x_{ID,k}(j^k-(j-1)^k)} \\ &= \prod_{k=1}^{t} g^{x_{ID,k}(j^k-(j-1)^k)} \\ &= \prod_{k=1}^{t} {Y_{ID,k}}^{j^k-(j-1)^k} \end{aligned}$$

3) Given the proof $proof = \{j, \mu, \eta\}$ produced by the cloud, the validity of $proof$ can be checked by the TPA using the equation (3). The validity of the proof in equation (3) is elaborated as follows:

$$\begin{aligned} e(\eta, g) =& e(\prod_{i\in S} {\beta_i}^{v_i}, g) \\ =& e(\prod_{i\in S} (H_1(f_{id}||i||j) \cdot u^{m_i})^{\sigma_{ID,j}\cdot v_i}, g) \\ =& e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\sum_{i\in S} m_i \cdot v_i}, g^{\sigma_{ID,j}}) \\ =& e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\mu}, \\ & g^{r_{ID}+f_{ID}(j)\cdot H_2(R_{ID},ID)}) \\ =& e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\mu}, \\ & g^{r_{ID}} \cdot g^{(x_0+\sum_{k=1}^{t} x_{ID,k}\cdot j^k)\cdot H_2(R_{ID},ID)} \\ =& e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\mu}, \\ & R_{ID} \cdot (g^{x_0} \cdot \prod_{k=1}^{t} g^{x_{ID,k}\cdot j^k})^{H_2(R_{ID},ID)}) \\ =& e(\prod_{i\in S} H_1(f_{id}||i||j)^{v_i} \cdot u^{\mu}, \\ & R_{ID} \cdot (Y_0 \prod_{k=1}^{t} {Y_{ID,k}}^{j^k})^{H_2(R_{ID},ID)}) \end{aligned}$$

Theorem 2: (**Security**) Assume the signature scheme employed for file tag is secure and the CDH problem in G is hard. Whenever an adversary in our security model can pass the challenger's verification by outputting a valid proof *proof* with non-negligible probability, there is a knowledge extractor that can extract the challenged data blocks with non-negligible probability.

Proof. If the proof produced by the cloud can pass the verification, a knowledge extractor can be constructed to extract the whole challenged data blocks. We define the following games to complete our proof.

Game 0. Game 0 is the game defined in Section III-D.

Game 1. Game 1 is equivalent to Game 0, except that the challenger keeps a list which consists of all the signed tags. If the adversary submits one tag, the challenger claims failure when this tag is a valid Sig signature but not signed by the challenger.

Analysis. If the challenger declares failure and aborts in Game 1 with non-negligible probability, a valid Sig signature can be easily forged by the adversary. This is in contradiction with the assumption that Sig is an unforgeable identity-based signature. Thus, the time period j^*, the verification value R_{ID^*} and the file identifier ${f_{id}}^*$ in the interactions with the adversary are all valid and produced by the challenger.

Game 2. Game 2 is equivalent to Game 1, except that the challenger maintains some records used to respond to the adversary's queries in local list. If the adversary can output a valid proof to pass the checking, while the adversary's aggregate authenticator is not equal to the expected $\prod_{i\in S}{\beta_i}^{v_i}$, the adversary succeeds, then the challenger will abort.

Analysis. Suppose the adversary produces a forged proof $\{j^*, \mu', \eta'\}$. This proof is able to satisfy the following equation,

$$\begin{aligned} e(\eta', g) = e(&\prod\nolimits_{i\in S} H_1({f_{id}}^*||i||j^*)^{v_i}\cdot u^{\mu'},\\ &R_{ID^*}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)}) \end{aligned} \tag{4}$$

Suppose $proof = \{j^*, \mu, \eta\}$ is a valid proof produced by the honest prover. $proof$ satisfies the following verification equation,

$$\begin{aligned} e(\eta, g) = e(&\prod\nolimits_{i\in S} H_1({f_{id}}^*|||i||j^*)^{v_i}\cdot u^{\mu},\\ &R_{ID^*}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)}) \end{aligned} \tag{5}$$

Obviously, $\mu \neq \mu'$, otherwise $\eta = \eta'$, which contradicts the above assumption. Define $\Delta\mu = \mu' - \mu$. We design a simulator to solve the CDH problem if the adversary succeeds with a non-negligible probability.

Given $g, g^\alpha, h \in G_1$, the aim of simulator is to calculate h^α. The simulator randomly picks $a, b \in Z_p^*$, and sets $u = g^a h^b$. Meanwhile, set $Y_0 = g^\alpha$ and select $\alpha_{ID^*,1}, \alpha_{ID^*,2}, \ldots, \alpha_{ID^*,t}, r_{ID^*} \in Z_p^*$ at random. Set $Y_{ID^*,k} = g^{\alpha_{ID^*,k}} (k \in [1,t])$ and $R_{ID^*} = g^{r_{ID^*}}$. Publish $Y_{ID^*,1}, Y_{ID^*,2}, \ldots, Y_{ID^*,t}$ and R_{ID^*}. For each $i \in [1,n]$ in the challenge, a random value $r_i \in Z_p^*$ is chosen by the simulator, then the simulator programs the random oracle at i as $H_1({f_{id}}^*||i||j^*) = g^{r_i}/(g^{am_i}\cdot h^{bm_i})$.

The simulator is able to calculate the data authenticator β_i, because we get

$$\begin{aligned} H_1({f_{id}}^*||i||j^*)\cdot u^{m_i} &= g^{r_i}/(g^{am_i}\cdot h^{bm_i})\cdot u^{m_i}\\ &= (g^{r_i}/(g^{am_i}\cdot h^{bm_i}))\cdot(g^{am_i}\cdot h^{bm_i})\\ &= g^{r_i} \end{aligned}$$

Thus, the simulator computes β_i as follows:

$$\begin{aligned} \beta_i &= (H_1({f_{id}}^*||i||j^*)\cdot u^{m_i})^{\sigma_{ID^*,j}}\\ &= (g^{r_i})^{\sigma_{ID^*,j}} = (g^{\sigma_{ID^*,j}})^{r_i}\\ &= (g^{r_{ID^*}+f_{ID^*}(j)H_2(R_{ID^*},ID^*)})^{r_i}\\ &= (g^{r_{ID^*}}\cdot g^{(\alpha+\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)\cdot H_2(R_{ID^*},ID^*)})^{r_i}\\ &= (R_{ID^*}\cdot(Y_0\cdot\prod\nolimits_{k=1}^{t} g^{\alpha_{ID^*,k}j^k})^{H_2(R_{ID^*},ID^*)})^{r_i}\\ &= (R_{ID^*}\cdot(Y_0\cdot\prod\nolimits_{k=1}^{t} {Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})^{r_i} \end{aligned}$$

Dividing equation (4) by equation (5), we obtain

$$\begin{aligned} &e(\eta'/\eta, g)\\ &= e(u^{\Delta\mu}, R_{ID^*}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})\\ &= e((g^a h^b)^{\Delta\mu}, R_{ID^*}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})\\ &= e(g^{a\Delta\mu}, g^{r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})\\ &\quad\cdot e(h^{b\Delta\mu}, g^{r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})\\ &= e(g^{a\Delta\mu r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)a\Delta\mu}, g)\\ &\quad\cdot e(h^{b\Delta\mu}, g^{r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)})\\ &= e(g^{a\Delta\mu r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)\cdot a\Delta\mu}, g)\\ &\quad\cdot e(h^{b\Delta\mu}, {Y_0}^{H_2(R_{ID^*},ID^*)})\\ &\quad\cdot e(h^{b\Delta\mu}, g^{r_{ID^*}}g^{(\sum_{k=1}^{t}\alpha_{ID,k}j^k)\cdot H_2(R_{ID^*},ID^*)})\\ &= e(g^{a\Delta\mu r_{ID^*}}(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)a\Delta\mu}, g)\\ &\quad\cdot e(h, Y_0)^{b\Delta\mu\cdot H_2(R_{ID^*},ID^*)}\\ &\quad\cdot e(h^{b\Delta\mu\cdot(r_{ID^*}+(\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)\cdot H_2(R_{ID^*},ID^*))}, g)\\ &= e(g^{a\Delta\mu r_{ID^*}}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)a\Delta\mu}\cdot\\ &\qquad h^{b\Delta\mu\cdot(r_{ID^*}+(\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)H_2(R_{ID^*},ID^*))}, g)\\ &\quad\cdot e(h, Y_0)^{b\Delta\mu\cdot H_2(R_{ID^*},ID^*)}\\ &= e(g^{a\Delta\mu r_{ID^*}}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{H_2(R_{ID^*},ID^*)a\Delta\mu}\cdot\\ &\qquad h^{b\Delta\mu\cdot(r_{ID^*}+(\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)H_2(R_{ID^*},ID^*))}, g)\\ &\quad\cdot e(h, g^\alpha)^{b\Delta\mu\cdot H_2(R_{ID^*},ID^*)} \end{aligned}$$

It further implies

$$\begin{aligned} &e(\eta'\cdot\eta^{-1}\cdot g^{-a\Delta\mu r_{ID^*}}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{-H_2(R_{ID^*},ID^*)a\Delta\mu}\cdot\\ &\quad h^{-b\Delta\mu\cdot(r_{ID^*}+(\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)\cdot H_2(R_{ID^*},ID^*))}, g)\\ &= e(h^\alpha, g)^{b\Delta\mu\cdot H_2(R_{ID^*},ID^*)} \end{aligned}$$

Therefore, we get a solution of solving the CDH problem as follows

$$\begin{aligned} h^\alpha =&(\eta'\cdot\eta^{-1}\cdot g^{-a\Delta\mu r_{ID^*}}\cdot(Y_0\prod\nolimits_{k=1}^{t}{Y_{ID^*,k}}^{j^k})^{-H_2(R_{ID^*},ID^*)a\Delta\mu}\cdot\\ &h^{-b\Delta\mu\cdot(r_{ID^*}+(\sum_{k=1}^{t}\alpha_{ID^*,k}j^k)H_2(R_{ID^*},ID^*))})^{1/(b\Delta\mu\cdot H_2(R_{ID^*},ID^*))} \end{aligned}$$

as long as the denominator $b\Delta\mu\cdot H_2(R_{ID^*},ID^*) \neq 0 \bmod p$. The probability that $b\Delta\mu\cdot H_2(R_{ID^*},ID^*) \neq 0 \bmod p$ is $1-1/p$, which is non-negligible. Therefore, it is contradiction with the assumption that the CDH problem in G is hard.

It means that if the difference between the adversary's probabilities of success in Game 1 and Game 2 is non-negligible, a simulator can be constructed to solve the CDH problem.

Game 3. Game 3 is equivalent to Game 2, except that the challenger maintains each interaction result with the adversary. If the adversary can output a valid proof to pass the checking, while the adversary's aggregated data block is different from the expected μ, then the challenger will abort.

Analysis. Suppose $proof = \{j^*, \mu, \eta\}$ is a valid proof outputted by the honest prover. $Proof$ can pass the checking of the following verification equation

$$e(\eta, g) = e(\prod\nolimits_{i\in S} H_1(f_{id^*}||i||j^*)^{v_i} \cdot u^{\mu}, R_{ID^*} \cdot (Y_0 \prod\nolimits_{k=1}^{t} Y_{ID^*,k}{}^{j^k})^{H_2(R_{ID^*}, ID^*)}).$$

Suppose the adversary outputs a forged proof $\{j^*, \mu', \eta'\}$. The following verification equation holds since the forgery is successful,

$$e(\eta', g) = e(\prod\nolimits_{i\in S} H_1(f_{id^*}||i||j^*)^{v_i} \cdot u^{\mu'}, R_{ID^*} \cdot (Y_0 \prod\nolimits_{k=1}^{t} Y_{ID^*,k}{}^{j^k})^{H_2(R_{ID^*}, ID^*)}).$$

We know $\eta' = \eta$ from the Game 2. Define $\Delta\mu = \mu' - \mu$. A simulator can be constructed to solve the DL problem.

Input $g, h \in G_1$ to the simulator. The aim of simulator is to calculate a value α which satisfies $h = g^{\alpha}$. The simulator randomly picks $a, b \in Z_p^*$, and sets $u = g^a h^b$. From the above two verification equations, we get

$$\begin{aligned} &e(\prod\nolimits_{i\in S} H_1(f_{id^*}||i||j^*)^{v_i} \cdot u^{\mu}, R_{ID^*} \\ &\quad \cdot (Y_0 \prod\nolimits_{k=1}^{t} Y_{ID^*,k}{}^{j^k})^{H_2(R_{ID^*}, ID^*)}) \\ =&e(\eta, g) = e(\eta', g) \\ =&e(\prod\nolimits_{i\in S} H_1(f_{id^*}||i||j^*)^{v_i} \cdot u^{\mu'}, R_{ID^*} \\ &\quad \cdot (Y_0 \prod\nolimits_{k=1}^{t} Y_{ID^*,k}{}^{j^k})^{H_2(R_{ID^*}, ID^*)}) \end{aligned}$$

Therefore, we obtain that $u^{\mu} = u^{\mu'}$, and further implies $1 = u^{\Delta\mu} = (g^a h^b)^{\Delta\mu} = g^{a\Delta\mu} \cdot h^{b\Delta\mu}$. Furthermore, we get $\Delta\mu \neq 0 \bmod p$, otherwise, we obtain $\mu' = \mu \bmod p$. This is contradiction with the aforementioned assumption. Thus, we find a solution to solve the DL problem as follows

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

as long as the denominator $a \neq 0 \bmod p$. The probability that $a \neq 0 \bmod p$ is $1 - 1/p$, which is non-negligible. Therefore, it is contradiction with the assumption that the DL problem in G is hard.

It means that if the difference between the adversary's probabilities of success in Game 2 and Game 3 is non-negligible, a simulator can be constructed to solve the DL problem.

There is only negligible difference probability between these games. Note that the hardness of the CDH problem implies the hardness of the discrete logarithm problem. If the signature scheme employed for file tag is secure and the CDH problem in G is hard, the challenger will reject except when the adversary generates a valid proof.

In the end, a knowledge extractor is constructed to extract all challenged data blocks $m_i (i \in S, |i| = c)$ by employing independent coefficients $v_i (i \in S, |I| = c)$ to generate proof on the same blocks $m_i (i \in S, |i| = c)$ for c times. The extractor can obtain c independently linear equations in the variables $m_i (i \in S, |i| = c)$. By solving these equations, the extractor can extract $m_i (i \in S, |i| = c)$ easily.

Theorem 3: (**The detectability**): Assume n data blocks are stored in the cloud and θ data blocks are corrupted in the proposed scheme. If c data blocks are challenged by the TPA, the cloud's misbehavior can be detected with the probability at least $1 - \left(\frac{n-\theta}{n}\right)^c$.

Proof. The cloud's misbehaviour can be detected if and only if at least one of the data blocks challenged by the TPA matches the corrupted data blocks. Let a discrete random variable Y be the number of challenged data blocks that matches the corrupted data blocks. We employ P_Y to denote the probability of detecting the cloud's misbehaviour. Thus, we have

$$\begin{aligned} P_Y =& P\{Y \geq 1\} \\ =& 1 - P\{Y = 0\} \\ =& 1 - \frac{n-\theta}{n} \times \frac{n-1-\theta}{n-1} \times ... \times \frac{n-c+1-\theta}{n-c+1} \end{aligned}$$

We have $P_Y \leq 1 - (\frac{n-\theta}{n})^c$. Therefore, the conclusion is that the proposed scheme is able to find the cloud's misbehavior with probability at least $1 - \left(\frac{n-\theta}{n}\right)^c$.

VI. Performance Evaluation

A. Functionalities Comparison

In Table II, we illustrate the functionalities comparison of our scheme with several related schemes [13, 18, 23, 30]. All of schemes in [13, 18, 30] can not resist key exposure. The scheme [23] can resist key exposure by updating the user's private key. However, it relies on public key infrastructure (PKI) which needs to execute the complicated certificate management. Besides, it cannot support private key lazy update. Compared with schemes [13, 18, 23, 30], our scheme satisfies the following four properties: certificate management simplification, public verifiability, key-exposure resistance and lazy update.

B. Performance Analysis

The following notations are defined to denote the computation costs of different operations in our scheme. The computation cost of one exponentiation operation in G is denoted as Exp_G. The computation cost of one pairing operation is denoted as $Pair$. Compared with Exp_G and $Pair$, the computation cost of other operations like addition, multiplication and hashing on G and the operations on Z_p^* are negligible.

(1) **Computation cost.** The comparison of computation cost for different entities is illustrated in table III. The PKG costs $(t+1)Exp_G$ to generate public values in the phase of setup, where t is the threshold value, and costs Exp_G to generate the user's initial private key in the phase of extraction. On the user side, the computation cost mainly comes from the process of authenticator generation. The overhead used to update private key can be ignored. In the process of authenticator generation, the computation cost is $2nExp_G$ on the user side, where n is the number of data blocks in the file. In our scheme, we do not consider the computation cost of the verification process on the user side because this process is optional. On the TPA side, the TPA needs to compute the update information and check the validity of the proof. However, the update information

TABLE II
FUNCTIONALITIES COMPARISON WITH EXISTING RELATED SCHEMES

Schemes	Certificate management simplification	Public verifiability	Key-Exposure resistance	Lazy update
Shacham et al. [30]	No	Yes	No	No
Fu et al.[13]	No	Yes	No	No
Wang et al.[18]	Yes	Yes	No	No
Yu et al.[23]	No	Yes	Yes	No
Ours	Yes	Yes	Yes	Yes

TABLE III
THE COMPARISON OF COMPUTATION COST FOR DIFFERENT ENTITIES OF OUR SCHEME AND WANG ET AL. SCHEME [18]

Scheme	**PKG**	**User**	**TPA**	**Cloud**
Wang et al. scheme [18]	Exp_G	$2nExp_G$	$2Pair + (c+2)Exp_G$	$cExp_G$
Our scheme	$(t+2)Exp_G$	$2nExp_G$	$2Pair + (c+4)Exp_G$	$cExp_G$

generation contributes negligible computation cost. In the process of auditing proof verification, the computation cost is $2Pair + (c+4)Exp_G$. On the cloud side, the computation cost for generating the proof is $cExp_G$.

To present the computation advantages of our scheme, we also give the computation cost of the classic scheme [18] in table III. We select the scheme [18] as a benchmark, because it is the representative state-of-the-art in identity-based data integrity auditing. As shown in table III, the PKG costs Exp_G to generate the private key for the user in scheme [18]. The computation cost of the user is $2nExp_G$, which is used to produce the authenticators. On the TPA side, the computation cost used to verify the correctness of auditing proof is $2Pair+(c+2)Exp_G$. The computation cost of outputting the proof on the cloud side is $cExp_G$.

As a result, the computation costs of the user and the cloud in our scheme are the same as that in the scheme [18]. To achieve the extra key-exposure resilience, the computation cost of the PKG and the TPA in our scheme add acceptable overhead than that in the scheme [18].

TABLE IV
THE COMPARISON OF COMMUNICATION COST OF OUR SCHEME AND WANG ET AL. SCHEME [18]

Scheme	Challenge	Proof
Wang et al. scheme [18]	$c \cdot (\|n\| + \|p\|)$	$\|p\| + \|q\|$
Our scheme	$c \cdot (\|n\| + \|p\|)$	$2\|p\| + \|q\|$

(2) **Communication cost.** The challenge and the proof in the auditing phase are the dominant factors for the communication costs of our scheme and the scheme [18]. As shown in table IV, the communication cost of the scheme [18] is almost the same as that of our scheme. In our scheme and the scheme [18], the size of the challenge is $c \cdot (|n| + |p|)$ bits, where $|n|$ and $|p|$ are the size of elements in $[1, n]$ and Z_p^* respectively. The size of the proof is $|p|+|q|$ bits in the scheme [18], where $|q|$ is the size of element in G. In our scheme, the size of the proof is $2|p| + |q|$ bits, which requires more communication costs than the scheme [18].

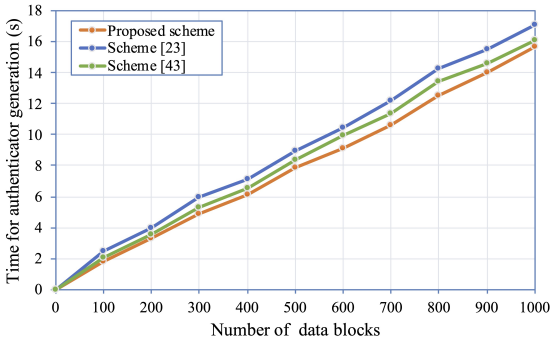

Fig. 4. Computation cost in the phase of authenticator generation

C. Experimental Results

We utilize C programming language with Pairing-Based Cryptography (PBC) Library [52] and the GNU Multiple Precision Arithmetic (GMP) [53] to code all algorithms. All simulation experiments are implemented on 64-bit Linux system with an Intel Core i5-6200 with 2.3GHz processor and 8GB memory. In our setting, the size of an element in Z_p^* is set as 160 bits, and the base field size is set as 512 bits.

To evaluate the experimental performance of our scheme, we choose the schemes [23], [43] as the benchmarks since they are well known as two most efficient data integrity auditing schemes with key-exposure resistance which are based on PKI cryptosystem and identity-based cryptosystem, respectively.

1) **Authenticator generation phase.** We evaluate the performance of authenticator generation and the results are shown in Fig. 4. We choose different numbers of data blocks with incremental numbers from 100 to 1,000 with 100 interval. Fig. 4 indicates that the computation costs of authenticator generation in our scheme and the schemes [23], [43] are linearly increase with the number of data blocks. Our scheme requires less computation cost in the

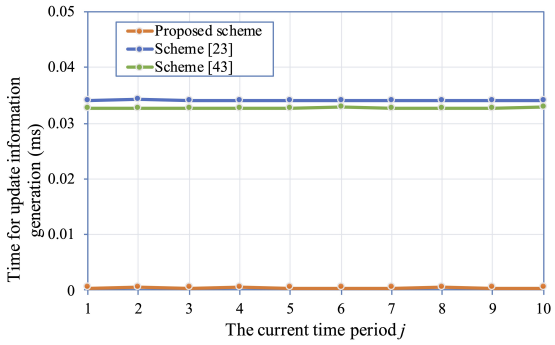

Fig. 5. Computation cost of update information generation

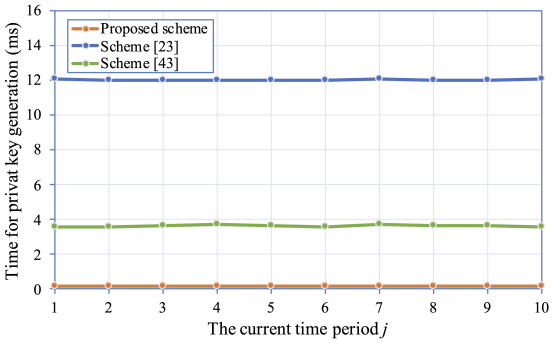

Fig. 6. Computation cost of private key update

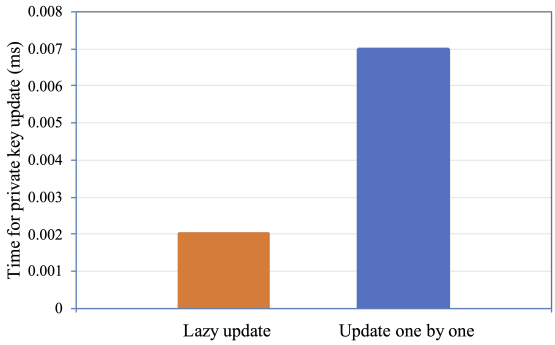

Fig. 7. The time of different private key update strategies

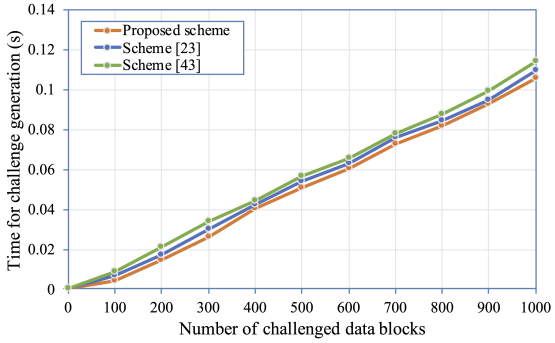

Fig. 8. Computation cost of challenge generation

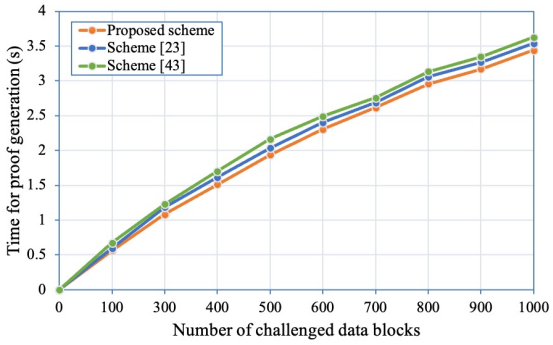

Fig. 9. Computation cost of proof generation

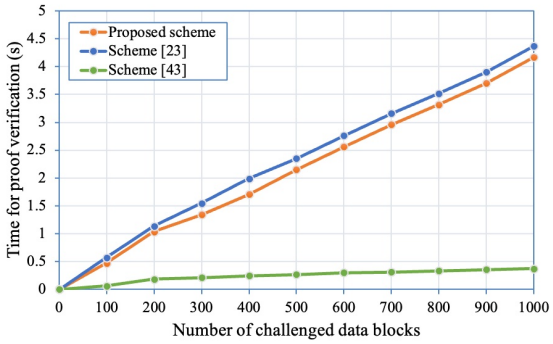

Fig. 10. Computation cost of proof verification

authenticator generation phase compared to the schemes [23], [43].

2) **Private key update phase.** In the phase of private key update, firstly the TPA generates the update information for the user, then the user updates his private key according to the update information. Fig. 5 and Fig. 6 show that, in our scheme and the schemes [23], [43], the computation costs of update information generation and private key generation in each time period are almost the same. However, the computation cost of our scheme is much smaller than that of the schemes [23], [43].

   To compare the computation costs of different private key update strategies in our scheme, we do the following two experiments. One is to update the private key directly from the time period 1 to the time period 7, which is called lazy update. The other one is to update the private key one by one from the time period 1 to the time period 7. As shown in Fig. 7, in lazy update, the user can directly generate the private key in the time period 7, which costs 0.002ms. However, in one by one update, the user has to calculate the private keys in all time periods from the time period 1 to the time period 7, which costs 0.007ms. Thus, lazy update achieves high efficiency compared with the one by one update.

3) **Auditing phase.** We evaluate the computation costs of three different processes in auditing phase with different numbers of challenged data blocks. In our experiment, we challenge different data blocks with incremental numbers

from 100 to 1000 with 100 interval. The Fig. 8, Fig. 9 and Fig. 10 illustrate that the computation costs of three processes are proportional to the number of challenged data blocks. The computation costs of challenge generation and proof generation in our scheme and the schemes [23], [43] are approximately equivalent. In the phase of the proof verification, our scheme and the scheme [23] require more computation costs on the cloud side compared with the scheme [43].

In data integrity auditing, both the TPA and the cloud have abundant computation resources. Therefore, the computational efficiency on the user side is our main concern in our scheme. The experiment results show that our scheme has better efficiency on the user side and performs well in the phase of private key update. It means that the user only needs to perform lightweight computations to update his private key.

VII. Conclusion

In this paper, we explore how to address the key exposure problem in identity-based data integrity auditing. We propose a novel and efficient identity-based data integrity auditing scheme with key-exposure resilience for cloud storage. In this scheme, even if the malicious cloud obtains the user's private key in a certain time period, the auditing task of other time periods are still able to work. The security proof and the performance analysis show that the proposed scheme is secure and efficient.

References

[1] K. Ren, C. Wang, and Q. Wang, "Security challenges for the public cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, Jan 2012.

[2] D. Song, E. Shi, I. Fischer, and U. Shankar, "Cloud data protection for the masses," *Computer*, vol. 45, no. 1, pp. 39–45, 2012.

[3] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 11, pp. 2572–2583, Nov 2016.

[4] F. Chen, F. Meng, T. Xiang, H. Dai, J. Li, and J. Qin, "Towards usable cloud storage auditing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 11, pp. 2605–2617, 2020.

[5] Y. Su, Y. Li, K. Zhang, and B. Yang, "A privacy-preserving public integrity check scheme for outsourced ehrs," *Information Sciences*, vol. 542, pp. 112 – 130, 2021.

[6] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, "Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics," *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020.

[7] X. Zhang, J. Zhao, C. Xu, H. Li, H. Wang, and Y. Zhang, "CIPPPA: Conditional identity privacy-preserving public auditing for cloud-based WBANs against malicious auditors," *IEEE Transactions on Cloud Computing*, pp. 1–14, 2019.

[8] Y. Zhang, C. Xu, X. Lin, and X. S. Shen, "Blockchain-based public integrity verification for cloud storage against procrastinating auditors," *IEEE Transactions on Cloud Computing*, pp. 1–15, 2019.

[9] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Transactions on Services Computing*, vol. 12, no. 5, pp. 824–835, Sep. 2019.

[10] H. Wang, H. Qin, M. Zhao, X. Wei, H. Shen, and W. Susilo, "Blockchain-based fair payment smart contract for public cloud storage auditing," *Information Sciences*, vol. 519, pp. 348 – 362, 2020.

[11] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *2010 Proceedings IEEE INFOCOM*, March 2010, pp. 1–9.

[12] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and R. Kotagiri, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, Sept 2014.

[13] A. Fu, S. Yu, Y. Zhang, H. Wang, and C. Huang, "NPP: A new privacy-aware public auditing scheme for cloud data sharing with group users," *IEEE Transactions on Big Data*, pp. 1–1, 2017.

[14] W. Shen, J. Qin, J. Yu, R. Hao, J. Hu, and J. Ma, "Data integrity auditing without private key storage for secure cloud storage," *IEEE Transactions on Cloud Computing*, pp. 1–15, 2019.

[15] B. Dan and M. Franklin, "Identity-based encryption from the weil pairing," *SIAM journal on computing*, vol. 32, no. 3, pp. 586–615, 2003.

[16] 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 Transactions on Information Forensics and Security*, vol. 12, no. 4, pp. 767–778, April 2017.

[17] X. Zhang, J. Zhao, C. Xu, H. Wang, and Y. Zhang, "DOPIV: Post-quantum secure identity-based data outsourcing with public integrity verification in cloud storage," *IEEE Transactions on Services Computing*, pp. 1–13, 2019.

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

[19] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1165–1176, June 2016.

[20] 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 Transactions on Dependable and Secure Computing*, vol. 17, no. 3, pp. 608–619, 2020.

[21] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling

cloud storage auditing with key-exposure resistance," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 6, pp. 1167–1179, 2015.

[22] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Transactions on Information Forensics and Security*, vol. 11, no. 6, pp. 1362–1375, June 2016.

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

[24] Y. Xu, R. Ding, J. Cui, and H. Zhong, "Intrusion-resilient public auditing protocol for data storage in cloud computing," in *Information Security and Privacy. ACISP 2018, Australasian Conference on Information Security and Privacy*, 2018, pp. 399–416.

[25] J. Zhang and Q. Dong, "Efficient id-based public auditing for the outsourced data in cloud storage," *Information Sciences*, vol. 343-344, pp. 1 – 14, 2016.

[26] Z. Xu, D. He, P. Vijayakumar, B. Gupta, and J. Shen, "Certificateless public auditing scheme with data privacy and dynamics in group user model of cloud-assisted medical wsns," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–1, 2021.

[27] 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 Transactions on Information Forensics and Security*, vol. 14, no. 2, pp. 331–346, Feb 2019.

[28] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07, 2007, pp. 598–609.

[29] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07, 2007, pp. 584–597.

[30] H. Shacham and B. Waters, "Compact proofs of retrievability," in *International conference on the theory and application of cryptology and information security*. Springer, 2008, pp. 90–107.

[31] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, 2008, pp. 1–10.

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

[33] H. Tian, Y. Chen, C. C. Chang, H. Jiang, Y. Huang, Y. Chen, and J. Liu, "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, Sept 2017.

[34] A. Yang, J. Xu, J. Weng, J. Zhou, and D. S. Wong, "Lightweight and privacy-preserving delegatable proofs of storage with data dynamics in cloud storage," *IEEE Transactions on Cloud Computing*, pp. 1–14, DOI:10.1109/TCC.2018.2851256, 2018.

[35] W. Guo, H. Zhang, S. Qin, F. Gao, Z. Jin, W. Li, and Q. Wen, "Outsourced dynamic provable data possession with batch update for secure cloud storage," *Future Generation Computer Systems*, vol. 95, pp. 309–322, 2019.

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

[37] Y. Li, Y. Yu, B. Yang, G. Min, and H. Wu, "Privacy preserving cloud data auditing with efficient key update," *Future Generation Computer Systems*, vol. 78, pp. 789–798, 2018.

[38] R. Ding, H. Zhong, J. Ma, X. Liu, and J. Ning, "Lightweight privacy-preserving identity-based verifiable iot-based health storage system," *IEEE Internet of Things Journal*, vol. 6, no. 5, pp. 8393–8405, 2019.

[39] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *2012 IEEE Fifth International Conference on Cloud Computing*, June 2012, pp. 295–302.

[40] J. Shen, F. Guo, X. Chen, and W. Susilo, "Secure cloud auditing with efficient ownership transfer," in *European Symposium on Research in Computer Security*. Springer, 2020, pp. 611–631.

[41] L. Zhou, A. Fu, Y. Mu, H. Wang, and Y. Sun, "Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system," *Information ences*, vol. 545, pp. 254–9276, 2021.

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

[43] S. M. V. Nithya and V. R. Uthariaraj, "Identity-based public auditing scheme for cloud storage with strong key-exposure resilience," *Security and Communication Networks*, pp. 1–13, 2020.

[44] X. Zhang, H. Wang, and C. Xu, "Identity-based key-exposure resilient cloud storage public auditing scheme from lattices," *Information Sciences*, vol. 472, pp. 223 – 234, 2019.

[45] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," *Journal of cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[46] K. S. McCurley, "The discrete logarithm problem," in *Proc. of Symp. in Applied Math*, vol. 42. USA, 1990, pp. 49–74.

[47] F. Bao, R. H. Deng, and H. Zhu, "Variations of diffie-hellman problem," in *International conference on information and communications security*. Springer, 2003, pp. 301–312.

[48] C.-L. Hsu and H.-Y. Lin, "New identity-based key-insulated convertible multi-authenticated encryption scheme," *Journal of Network and Computer Applications*, vol. 34, no. 5, pp. 1724–1731, 2011.

[49] Y. S. Rao and R. Dutta, "Bandwidth-efficient attribute-based key-insulated signatures with message recovery," *Information Sciences*, vol. 369, pp. 648–673, 2016.
[50] Y. Watanabe and J. Shikata, "Identity-based hierarchical key-insulated encryption without random oracles," in *Public-Key Cryptography–PKC 2016*. Springer, 2016, pp. 255–279.
[51] S. Alornyo, Y. Zhao, G. Zhu, and H. Xiong, "Identity based key-insulated encryption with outsourced equality test." *Int. J. Netw. Secur.*, vol. 22, no. 2, pp. 257–264, 2020.
[52] B. Lynn, "The pairing-based cryptographic library," https://crypto.stanford.edu/pbc/, 2015.
[53] "The gnu multiple precision arithmetic library (gmp)," http://gmplib.org/.

Wenting Shen is an associate professor of the College of Computer Science and Technology at Qingdao University. She received the M.S. and B.S. degrees in college of Computer Science and Technology from Qingdao University, China, in 2017 and 2014, respectively. She received Ph. D. degree in School of Mathematics from Shandong University, in 2020. Her research interests include cloud security and big data security.

Jia Yu is a professor of the College of Computer Science and Technology at Qingdao University. He received the M.S. and B.S. degrees in School of Computer Science and Technology from Shandong University in 2003 and 2000, respectively. He received Ph. D. degree in Institute of Network Security from Shandong University, in 2006. He was a visiting professor with the Department of Computer Science and Engineering, the State University of New York at Buffalo, from Nov. 2013 to Nov. 2014. His research interests include cloud computing security, key evolving cryptography, digital signature, and network security.

Ming Yang received the B.S. and M.S. degrees from the School of Information Science and Engineering, Shandong University, in 2004 and 2007, and the Ph.D. degree from the School of Electronic Engineering, Beijing University of Posts and Telecommunications, in 2010. He is currently an Assistant Professor with Qilu University of Technology (Shandong Academy of Sciences), Shandong Computer Science Center, Shandong Provincial Key Laboratory of Computer Networks. His research interests include cloud computing security, big data security and network security.

Jiankun Hu receive the Ph.D. degree in Control Engineering from Harbin Institute of Technology, China in 1993 and the master's degree in Computer Science and Software Engineering from Monash University, Australia in 2000. He was a Research Fellow in Delft University of the Netherlands, from 1997 to 1998, and Melbourne University, Australia, from 1998 to 1999. His main research interest is in the field of cyber security, including biometrics security, where he has published many papers in high-quality conferences and journals including IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI). He has served at the editorial board of up to 7 international journals and served as Security Symposium Chair of IEEE flagship conferences of IEEE ICC and IEEE Globecom. He has obtained 7 ARC (Australian Research Council) Grants and is now serving on the prestigious Panel of Mathematics, Information and Computing Sciences (MIC), ARC ERA (The Excellence in Research for Australia) Evaluation Committee.