Hindawi
Security and Communication Networks
Volume 2019, Article ID 8275074, 15 pages
https://doi.org/10.1155/2019/8275074

Research Article

A Lightweight Identity-Based Cloud Storage Auditing Supporting Proxy Update and Workload-Based Payment

Wenting Shen,[1] Jing Qin ,[1,2] and Jixin Ma[3]

[1]*School of Mathematics, Shandong University, Shandong 250100, China*
[2]*State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China*
[3]*School of Computing and Mathematical Science, University of Greenwich, London, UK*

Correspondence should be addressed to Jing Qin; qinjing@sdu.edu.cn

Received 25 October 2018; Accepted 2 January 2019; Published 3 March 2019

Academic Editor: David Nuñez

Cloud storage auditing allows the users to store their data to the cloud with a guarantee that the data integrity can be efficiently checked. In order to release the user from the burden of generating data signatures, the proxy with a valid warrant is introduced to help the user process data in lightweight cloud storage auditing schemes. However, the proxy might be revoked or the proxy's warrant might expire. These problems are common and essential in real-world applications, but they are not considered and solved in existing lightweight cloud storage auditing schemes. In this paper, we propose a lightweight identity-based cloud storage auditing scheme supporting proxy update, which not only reduces the user's computation overhead but also makes the revoked proxy or the expired proxy unable to process data on behalf of the user any more. The signatures generated by the revoked proxy or the expired proxy can still be used to verify data integrity. Furthermore, our scheme also supports workload-based payment for the proxy. The security proof and the performance analysis indicate that our scheme is secure and efficient.

1. Introduction

Cloud storage enables users to store their data in the cloud without retaining a copy locally. It greatly reduces the burden of data storage management and maintenance and avoids the capital expenditure on software/hardware on the user side. Although cloud storage provides a lot of appealing benefits for users, it also incurs a number of security challenges [1]. Once the users upload their data to the cloud, they will lose the physical control of their data since they no longer keep their data locally. Thus, it is natural for users to worry whether their data stored in the cloud is intact or not due to the inevitable human errors or software/hardware bugs in the cloud. In order to guarantee the data integrity and reduce the user's computation and communication burden, some public cloud storage auditing schemes [2–4] have been proposed to allow a public verifier, such as the Third Party Auditor (TPA), to perform periodical data integrity auditing tasks on behalf of users.

In most existing public cloud storage auditing schemes, the user relieves the computation burden for verifying data integrity by introducing the TPA, but he still needs to perform heavy computation for generating data signatures before uploading data to the cloud. These data signatures are used to check the integrity of cloud data. In order to deal with the above problem, several lightweight cloud storage auditing schemes [5–8] have been proposed. Wang et al. [5] proposed a proxy-oriented data integrity auditing scheme. In this scheme, the proxy helps the user to generate data signatures, which obviously alleviates the user's computation burden. Shen et al. [6] constructed a lightweight cloud storage auditing scheme, which introduces the Third Party Medium (TPM) to generate data signatures and verify the data integrity for users. These schemes introduced a proxy with powerful computation capabilities to execute time-consuming operation on behalf of users. Nevertheless, there are two critical problems not well solved in all existing lightweight cloud storage auditing schemes.

Firstly, these lightweight cloud storage auditing schemes cannot support proxy update. In real-world applications, in order to designate a proxy, the user needs to issue a warrant with a valid time period to this proxy. The cloud will verify whether the proxy is valid based on this warrant. Furthermore, the proxy might be revoked before the expiration of this valid time period. For instance, a user with low computation capabilities delegates a proxy to help him process data. The proxy can be a legal organization which has significant computation resources. The user and the proxy sign a contract with a specific valid time period. Once the contract expires, the expired proxy should not be able to process data on behalf of the user any more. Another situation is that the user wants to change the proxy before the expiration of the contract's valid time period due to the misbehaviour of the proxy. This revoked proxy also should not be able to process data for the user any more even if his warrant is still in a valid time period. Thus, how to design a lightweight cloud storage auditing supporting proxy update is worthwhile.

Secondly, existing lightweight cloud storage auditing schemes do not consider the mechanism of paying for the proxy based on the workload. In the lightweight cloud storage auditing scenario, the user delegates a proxy to upload files to the cloud on his behalf. However, the number of these uploaded files might be greatly different in different time period. Obviously, it is unfair for the proxy if we pay for the proxy according to service time. It is more reasonable to pay for the proxy according to how many files he uploads to the cloud. Therefore, it is necessary to design an effective mechanism to pay for the proxy based on the workload in cloud storage auditing.

Contribution. In order to deal with the above problems, we propose a novel lightweight identity-based cloud storage auditing scheme. In this scheme, we introduce a proxy to help the user generate data signatures, which remarkably releases the users' burden on computation. Different from the previous lightweight cloud storage auditing schemes, our proposed scheme supports proxy update. In the detailed scheme, the user issues a warrant to the designated proxy. The proxy with the valid warrant can process data on behalf of the user. In order to realize effective proxy update, the valid time period and the proxy identity are embedded into the warrant and the cloud keeps a public revocation list. It makes the revoked proxy or the expired proxy unable to process and upload data to the cloud on behalf of the user any more. When the proxy is revoked or the proxy's warrant expires, the signatures generated by this proxy can still be used to check data integrity according to this proxy identity, the corresponding time period, and some verification values. We also design an effective mechanism to achieve workload-based payment for the proxy. Our scheme relies on identity-based cryptography, which simplifies certificate management. We finally prove the security of the scheme and evaluate the performance of the scheme by concrete implementations.

1.1. Related Work. Ateniese et al. [2] firstly proposed the notion of Provable Data Possession (PDP), in which the techniques of homomorphic authenticators and random sample are utilized to verify the integrity of data in the cloud. To achieve the retrievability and the integrity guarantee of the cloud data, Juels and Kaliski [9] proposed the concept of Proof of Retrievability (PoR) and designed a concrete scheme by employing the techniques of the spot-checking and the error-correcting codes. Later, many variants of PDP and PoR schemes [10–13] were constructed to deal with different problems in cloud storage auditing.

In order to protect data privacy, Wang et al. [13] proposed a cloud storage auditing scheme with data privacy preserving by using a random masking technique. Li et al. [14] presented a privacy preserving remote data integrity auditing scheme based on the zero-knowledge proof. To support data dynamic operations, Liu et al. [15] designed a dynamic data integrity auditing scheme with efficient fine-grained updates based on the Merkle hash tree. Tian et al. [16] proposed a cloud storage auditing scheme supporting data dynamics with the employment of dynamic hash table. By utilizing key update techniques [17], Yu et al. [18] solved the problem of key exposure in cloud storage auditing. The privacy preserving of authentication in cloud storage auditing was considered in [19]. The problem of users' identity privacy in shared data auditing was taken into account in [20, 21].

To eliminate the complex certificate management in PKI setting, Wang et al. [22] constructed the first identity-based remote data integrity auditing scheme. Based on identity-based cryptosystem, Yu et al. [23] designed a perfect data privacy-preserving cloud storage auditing scheme, which is able to achieve zero knowledge privacy against a third party auditor. Li et al. [24] proposed a fuzzy identity-based cloud storage auditing scheme, in which a set of descriptive attributes is used as a user's identity. Zhang et al. [25] considered the problem of user revocation in the cloud storage auditing and presented an identity-based shared data integrity auditing scheme supporting real efficient user revocation. Shen et al. [26] proposed an identity-based cloud storage auditing scheme for shared data. In this scheme, the cloud file can be shared and used by others under the condition that the sensitive information is hidden.

Most end users, such as smart phone, have constrained computation resources and computation capabilities. In order to alleviate the user's computation burden, many lightweight cloud storage auditing schemes were proposed. Li. et al. [27] and Wang et al. [28], respectively, proposed cloud storage auditing schemes for low-performance end devices based on online/offline signatures. In these two schemes, most heavy computations are executed in the offline phase. Nonetheless, the user still needs to carry out lightweight computations for generating data signatures in the online phase. In order to deal with this problem, Wang et al. [5] proposed an identity-based cloud storage auditing scheme by introducing a proxy to help users generate data signatures. In this scheme, the user does not need to consume computation resources to generate data signatures. Shen et al. [6] designed a lightweight data integrity auditing scheme for cloud storage, in which the Third Party Medium (TPM) is

introduced to process data on behalf of users. Wang et al. [8] constructed a comprehensive cloud storage auditing scheme. In this scheme, the user delegates the task of data signature generation to a dedicated proxy. However, all these schemes that introduce the proxy do not consider the problem of proxy update. In addition, the problem of workload-based payment was also not taken into account. Actually, these problems are common and essential in real-world applications as discussed above.

1.2. Organization. The rest of this paper is organized as follows. We present the system model and design goals in Section 2. In Section 3, we give the notations and the definition and review several cryptographic knowledge. The proposed cloud storage auditing scheme is introduced in Section 4. The security proof and performance evaluation of the proposed scheme are given, respectively, in Section 5 and Section 6. Section 7 concludes the paper.

2. System Model and Design Goals

2.1. System Model. As shown in Figure 1, there are five types of entities in our system model, that is, the user, the cloud, the proxy, the Private Key Generator (PKG), and the TPA.

(1) User: The user is the data owner, who has massive data files to be stored to the cloud. Most end users, such as smart phone and PDAs, have limited computation resources and computation capabilities.

(2) Cloud: The cloud has enormous storage space and computation resources and offers data storage services for the user.

(3) Proxy: The proxy is authorized by the user and helps the user to process and upload data to the cloud.

(4) PKG: The PKG is responsible for generating global parameters, the partial private key, and the private keys for the proxy, the user, and the cloud, respectively, according to their identities.

(5) TPA: The TPA is in charge of executing cloud storage auditing on behalf of the user. The TPA can check whether the cloud stores the user's data correctly by performing the challenge-response protocol with the cloud.

In our system model, when a user would like to store his data to the cloud, he will delegate a proxy to help him to process data. Meanwhile, the user generates the warrant-signature pair and the time private key and then sends them to the proxy. The proxy with a valid warrant can generate the signatures for data and upload these data blocks along with the signature set to the cloud on behalf of the user. When the proxy is revoked or the proxy's warrant expires, this proxy cannot process data for the user any more. The user pays for the proxy based on the proxy's workload. Same as the system mode in cloud storage schemes [5, 8, 12, 25, 29], we do not take any collusion into account in this system.

2.2. Design Goals. To support proxy update, workload-based payment, and lightweight identity-based cloud storage auditing, our designed scheme should achieve the following goals:

(1) *Auditing correctness*: to ensure that the auditing proof generated by the cloud can pass the TPA's verification, if the cloud possesses the user's intact data.

(2) *Auditing soundness*: to guarantee that the cloud cannot pass the validation of the TPA if it does not keep the user's data correctly.

(3) *Lightweight*: to reduce the computation burden of generating data signatures and verifying data integrity for the user with constrained computation capabilities.

(4) *Secure proxy update*: to ensure that the revoked proxy or the expired proxy cannot process data on behalf of the user any more.

(5) *Efficient proxy update*: to guarantee that the data signatures generated by the proxy do not need to be recomputed even if this proxy is revoked or his warrant expires.

(6) *Effective payment*: to ensure that the user can pay for the proxy based on the workload of the proxy.

3. Notations, Definition, and Cryptographic Knowledge

3.1. Notations. In Table 1, we show the notations used in the description of our scheme.

3.2. Definition

Definition 1. A lightweight identity-based cloud storage auditing scheme supporting proxy update and workload-based payment is composed by the following eight algorithms:

(1) *Setup*: The setup algorithm is run by the PKG. It takes as input the security parameter k and outputs the master private key x and the public parameters pp.

(2) *Extract*: The extract algorithm is run by the PKG. It takes as input the public parameters pp, the user identity ID_u, the cloud identity ID_c, the proxy identity ID_p, and the master private key x and generates the user ID_u's private key sk_u, the cloud ID_c's private key sk_c, and the proxy ID_p's partial private key sk_p. The user verifies the correctness of the private key sk_u. The cloud checks the correctness of the private key sk_c. The proxy verifies the correctness of the partial private key sk_p.

(3) *ProxyKeyGen*: The proxy private key generation algorithm is run by the user and the proxy. It takes as input the warrant's start time T_s and end time T_e, the user identity ID_u, the proxy identity ID_p, and the partial private key sk_p. The user generates the time private key sk_T and the warrant-signature pair $(m_w, (R_1, \sigma_1))$. The proxy generates the proxy private

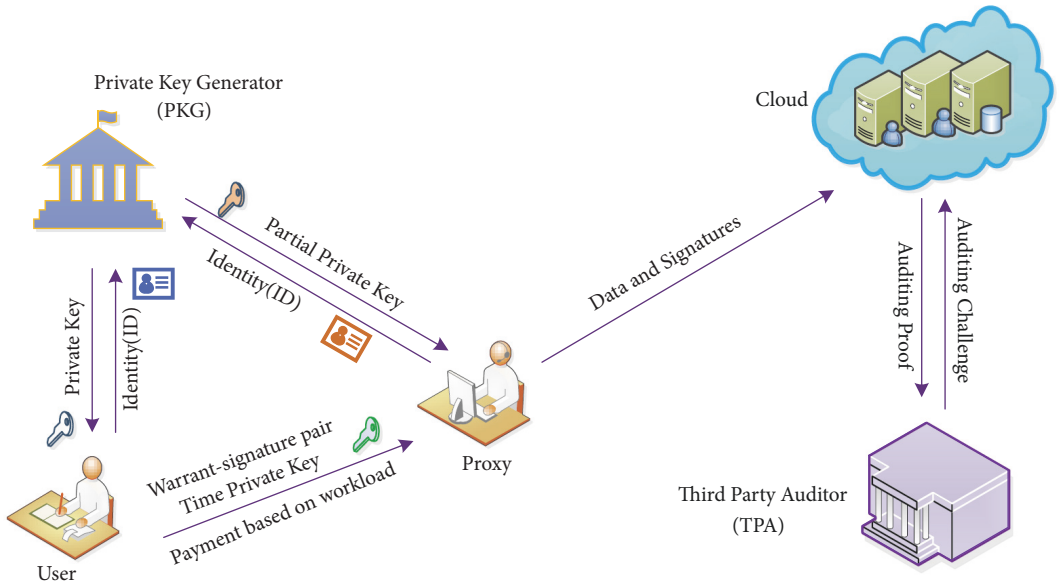

Figure 1: System model of our cloud storage auditing.

Table 1: Notations.

Notation	Meaning
G_1, G_2	Multiplicative cyclic groups with prime order p
g, u	Two generators of group G_1
e	A bilinear pairing map $e : G_1 \times G_1 \longrightarrow G_2$
H	A cryptographic hash function, $H : \{0,1\}^* \times G_1 \longrightarrow Z_p^*$
h	A cryptographic hash function, $h : \{0,1\}^* \longrightarrow G_1$
k	The security parameter
x	The master private key
ID_u	The user identity
ID_p	The proxy identity
ID_c	The cloud identity
sk_{ID_u}	The user ID_u's private key
sk_{ID_p}	The proxy ID_p's partial private key
sk_{ID_c}	The cloud ID_c's private key
m_w	The warrant
T_s, T_e	The warrant's start time and end time
(R_T, σ_T)	The time private key
(R_{ID_p}, R_T, σ)	The proxy private key
F	The file
n	The number of data blocks of file F
m_i $(i = 1, 2, \ldots, n)$	The i-th block of file F
σ_i	The signature of data block m_i
Φ	The signature set of data blocks
N	The upload number
μ	The linear combination of data blocks
σ	An aggregated data signature

key (R_{ID_p}, R_T, σ) according to the time private key and the partial private key.

(4) *SignGen*: The signature generation algorithm is run by the proxy. It takes as input the file name *name*, the file F, the warrant's start time T_s and end time T_e, and the proxy private key (R_{ID_p}, R_T, σ) and generates the set of signatures Φ and the file tag *tag*.

(5) *ProofGen*: The proof generation algorithm is run by the cloud. It takes as input the file F, the corresponding signature set Φ, and the auditing challenge *chal* and generates an auditing proof P that is used to prove the cloud stores the file correctly.

(6) *ProofVerify*: The proof verification algorithm is run by the TPA. It takes as input the public parameters *pp*, the auditing challenge *chal,* and the auditing proof P and returns "1" if the verification passes or "0" if otherwise.

(7) *ProxyUpdate*: The proxy update algorithm is run by the user and the proxy. It takes as input the new warrant's start time T_s and end time T_e, the user identity ID_u, the proxy identity ID_p, and the partial private key sk_p. The user generates a new time private key and a new warrant-signature pair. The proxy generates the proxy private key according to the new time private key and the partial private key.

(8) *Payment*: The payment algorithm is run by the cloud, the proxy, and the user. It takes as input the upload number N and the cloud's private key sk_c and generates the signature β corresponding to N. The proxy and the user check the validity of β.

3.3. Cryptographic Knowledge

(1) Bilinear Maps. A bilinear map is a map $e : G_1 \times G_1 \longrightarrow G_2$, where G_1 and G_2 are two multiplicative cyclic groups of order p with the following properties:

(a) *Computability*: there exists an efficiently computable algorithm to calculate $e(u, v)$ for $u, v \in G_1$.

(b) *Bilinearity*: for all $u, v \in G_1$ and $x, y \in Z_p^*$, $e(u^x, v^y) = e(u, v)^{xy}$.

(c) *Nondegeneracy*: $e(g, g) \neq 1$, where g is the generator of G_1.

(2) Computational Diffie-Hellman (CDH) Problem. Given g^x and g^y, where g is a generator of a multiplicative group G_1 with the prime order p, and $x, y \in Z_p^*$ are unknown, calculate $g^{xy} \in G_1$. The CDH assumption in G_1 holds if it is computationally infeasible to solve the CDH problem in G_1.

(3) Discrete Logarithm (DL) Problem. Given g^a, where g is a generator of a multiplicative group G_1 with the prime order p, and $a \in Z_p^*$ is unknown, calculate a. The DL assumption in G_1 holds if it is computationally infeasible to solve the DL problem in G_1.

4. The Proposed Scheme

In our scheme, the file F is divided into n blocks, i.e., $F = (m_1, m_2, \ldots, m_n)$, where m_i denotes the ith block of file F. In previous cloud storage auditing schemes [18, 32], there is a signature *SSig* that is used to guarantee the correctness of the file identifier name. Without loss of generality, we also utilize a similar identity-based signature *SSig* to ensure the correctness of the file identifier name, the warrant's valid time period, the proxy identity, and the verification values in our scheme. We assume *ssk* is the secret key for the signature *SSig*. The proxy holds this secret key. In addition, we assume the upload number is N, which is used to record how many files are uploaded to the cloud by the proxy. The upload number is initialized to 0. The details of the proposed scheme are as follows.

(1) Setup

(a) The PKG randomly selects two multiplicative cyclic groups G_1, G_2 with the prime order p, a bilinear map $e : G_1 \times G_1 \longrightarrow G_2$, two generators $g, u \in G_1$, and two cryptographic hash functions $H : \{0,1\}^* \times G_1 \longrightarrow Z_p^*$ and $h : \{0,1\}^* \longrightarrow G_1$.

(b) The PKG randomly picks $x \in Z_p^*$ as its master private key and computes $Y = g^x$.

(c) The PKG publishes the global parameters $pp = (G_1, G_2, p, g, e, u, Y, H, h)$ and keeps his master private key x.

(2) Extract. The PKG generates private keys for the user ID_u and the cloud ID_c, respectively, and generates the partial private key for the proxy ID_p. The user ID_u and the cloud ID_c can verify the correctness of their private keys, respectively. The proxy ID_p can check correctness of the partial private key.

(a) After receiving the user's identity ID_u, the PKG randomly chooses $r_{ID_u} \in Z_p^*$, and calculates $R_{ID_u} = g^{r_{ID_u}}$ and $\sigma_{ID_u} = r_{ID_u} + xH(ID_u, R_{ID_u})$. Set the user's private key $sk_{ID_u} = (R_{ID_u}, \sigma_{ID_u})$. The PKG sends sk_{ID_u} to the user ID_u.

Upon receiving the private key from the PKG, the user ID_u validates the correctness of sk_{ID_u} by checking whether the following equation holds or not:

$$g^{\sigma_{ID_u}} = R_{ID_u} Y^{H(ID_u, R_{ID_u})}. \quad (1)$$

If (1) holds, the user ID_u accepts the private key sk_{ID_u}; otherwise, the user rejects it.

(b) Similarly, receiving the cloud's identity ID_c, the PKG calculates the cloud's private key $sk_{ID_c} = (R_{ID_c}, \sigma_{ID_c})$, where $R_{ID_c} = g^{r_{ID_c}}$ and $\sigma_{ID_c} = r_{ID_c} + xH(ID_c, R_{ID_c})$ and then sends it to the cloud. The cloud can verify the correctness of sk_{ID_c} by checking the equation $g^{\sigma_{ID_c}} = R_{ID_c} Y^{H(ID_c, R_{ID_c})}$.

(c) Receiving the proxy's identity ID_p, the PKG calculates the proxy's private key, and the proxy can get its partial private key $sk_{ID_p} = (R_{ID_p}, \sigma_{ID_p})$, where $R_{ID_p} = g^{r_{ID_p}}$ and $\sigma_{ID_p} = r_{ID_p} + xH(ID_p, R_{ID_p})$. The proxy can check the correctness of sk_{ID_p} by checking the equation $g^{\sigma_{ID_p}} = R_{ID_p} Y^{H(ID_p, R_{ID_p})}$.

(3) ProxyKeyGen. This process is illustrated in Figure 2. The user ID_u firstly sets the warrant's start time and end time, then generates the time private key and the warrant-signature pair, and finally sends all the above messages to the proxy ID_p. The proxy ID_p can verify the correctness of the above messages, respectively, and then computes the proxy private key based on the partial private key and the time private key.

(a) The user ID_u executes the following steps.

(i) The user ID_u sets the warrant's start time T_s and end time T_e. The user ID_u selects $r_T \in_R Z_p^*$ and computes $R_T = g^{r_T}$ and $\sigma_T = r_T + \sigma_{ID_u} \cdot H(T_s \parallel T_e, R_T)$. Let $sk_T = (R_T, \sigma_T)$ be the time private key.

(ii) The user ID_u generates a warrant m_w, which is used to authorize a designated proxy. The warrant m_w consists of the user's identity ID_u, the proxy's identity ID_p, and the warrant's valid time period (start time T_s and end time T_e). Let the warrant be denoted as $m_w = ID_u \parallel ID_p \parallel T_s \parallel T_e$. The user ID_u picks $r_1 \in_R Z_p^*$ and calculates the warrant m_w's signature as follows: $R_1 = g^{r_1}$ and $\sigma_1 = r_1 + \sigma_{ID_u} \cdot H(m_w, R_1)$.

(iii) The user ID_u sends the warrant's valid time period (T_s, T_e), the time private key (R_T, σ_T), warrant-signature pair $(m_w, (R_1, \sigma_1))$, and the verification value R_{ID_u} to the proxy ID_p.

(b) Upon receiving the messages from the user ID_u, the proxy ID_p does the following steps:

(i) The proxy ID_p verifies the validity of warrant-signature pair $(m_w, (R_1, \sigma_1))$ by checking whether the following equation holds:

$$g^{\sigma_1} = R_1 R_{ID_u}^{H(m_w, R_1)} Y^{H(ID_u, R_{ID_u}) \cdot H(m_w, R_1)}. \quad (2)$$

If (2) does not hold, the proxy ID_p rejects the warrant m_w from the user ID_u; otherwise, he accepts the warrant and then does step (ii).

(ii) The proxy ID_p checks the correctness of time private key (R_T, σ_T) by validating the following equation:

$$g^{\sigma_T} = R_T R_{ID_u}^{H(T_s \| T_e, R_T)} Y^{H(ID_u, R_{ID_u}) \cdot H(T_s \| T_e, R_T)}. \quad (3)$$

If (3) holds, the proxy ID_p computes $\sigma = (\sigma_{ID_p} + \sigma_T)(mod p)$ and sets the proxy private key as (R_{ID_p}, R_T, σ).

(4) SignGen. This process is illustrated in Figure 3. The proxy ID_p calculates the signatures for data blocks with the proxy private key and then generates the file tag which is used to guarantee the correctness of the file name, the warrant's valid time period, the proxy identity, and some verification values. Finally, the proxy ID_p sends the file F, its corresponding signature set, the file tag, the warrant's valid time period, and the warrant-signature pair to the cloud. The cloud firstly checks whether the proxy is in the public revocation list, then verifies the validity of the warrant-signature pair and the file tag, and finally generates a signature for the upload number.

(a) The proxy ID_p performs the following operations.

(i) The proxy ID_p computes the signature β_i of the data block m_i with the proxy private key σ as follows:

$$\beta_i = \left(h\left(name \| i \| ID_p\right) u^{m_i}\right)^{\sigma}, \quad (4)$$

where $name \in Z_p^*$ is the identifier of the file F. Denote the set of signatures as $\Phi = \{\beta_i\}_{i \in [1,n]}$.

(ii) The proxy ID_p calculates the file tag $tag = name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p \parallel SSig_{ssk}(name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p)$.

(iii) The proxy ID_p sends the file F, its corresponding signatures set Φ, the file tag tag, the warrant's valid time period (T_s, T_e), and the warrant-signature pair $(m_w, (R_1, \sigma_1))$ to the cloud.

(iv) The proxy ID_p sets the upload number $N = N + 1$, which is used to record how many files are uploaded to the cloud.

(b) After receiving the above messages, the cloud executes the following operations.

(i) The cloud verifies whether the proxy ID_p is in the public revocation list. If the proxy ID_p is not in the revocation list, the cloud does the following step (ii); otherwise, the cloud regards the proxy ID_p as a revoked proxy and refuses his request.

(ii) The cloud checks the validity of the warrant-signature pair $(m_w, (R_1, \sigma_1))$ based on (2). If the current time period is not in the valid time period (T_s, T_e), then the warrant m_w can be regarded as an invalid warrant and the cloud refuses the request of the proxy; otherwise, the cloud continues to perform the following step (iii).

(iii) The cloud checks the validity of the file tag by verifying whether the signature $SSig_{ssk}(name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p)$ is a valid signature or not based on the proxy identity ID_p. If it is not, the cloud believes this signature is invalid; otherwise, the cloud does the last step.

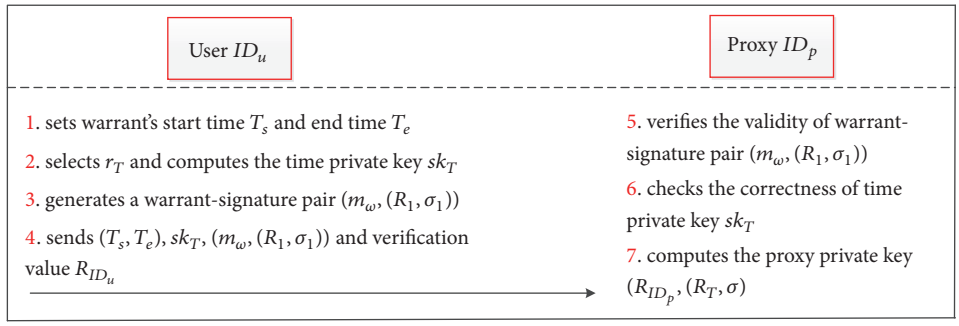

Figure 2: The process of proxy private key generation.

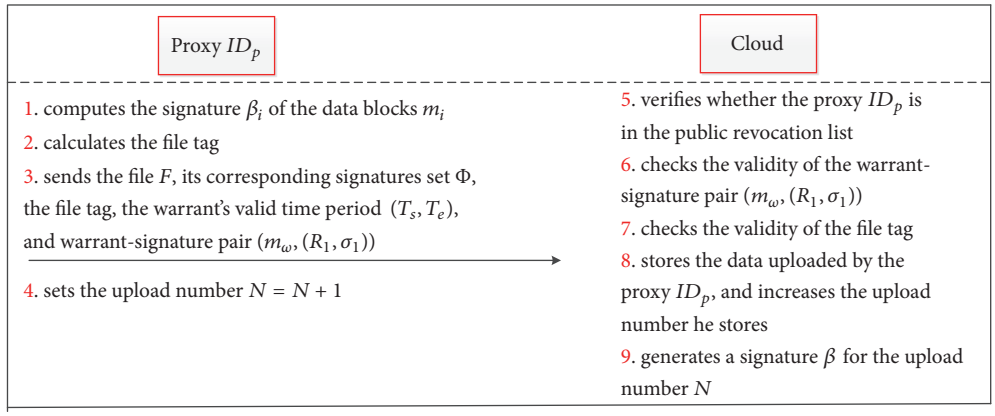

Figure 3: The process of signature generation.

(iv) The cloud stores the data uploaded by the proxy ID_p and increases the upload number he stores to keep pace with the number N stored by the proxy. Then the cloud generates a signature $\beta = SSig_{sk_c}(N)$ with its private key sk_c for the upload number N.

Remark 2. The proxy usually uploads data in bulk to the cloud. Thus, the cloud may only need to generate a signature for the newest upload number rather than every upload number.

(5) ProofGen. The TPA firstly verifies the validity of file tag and then generates and sends an auditing challenge to the cloud. The cloud responds to an auditing proof to the TPA.

(a) The TPA firstly verifies the correctness of the file tag *tag* by checking whether $SSig_{ssk}(name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p)$ is a valid signature. If it is not, the TPA does not execute the auditing task; otherwise, he parses $name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p$ to obtain the file name *name*, the warrant's valid time period (start time T_s and end time T_e), the verification values R_{ID_p}, R_T, and the proxy identity ID_p. Then the TPA randomly picks a c-element subset I from the set $[1, n]$ and selects a random $v_i \in Z_p^*$ for each $i \in I$. The TPA sends the auditing challenge $chal = \{i, v_i\}_{i \in I}$ to the cloud.

(b) After receiving the auditing challenge, the cloud calculates $\mu = \sum_{i \in I} m_i v_i$ and $\Omega = \prod_{i \in I} \beta_i^{v_i}$. Next, the cloud returns $P = \{\mu, \Omega\}$ as the auditing proof to the TPA.

(6) ProofVerify. The TPA verifies whether the following equation holds or not:

$$e(\Omega, g) = e\left(\prod_{i \in I} h\left(name \left\| ID_p \right\| i\right)^{v_i} \cdot u^{\mu}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s \| T_e, R_T)} \cdot Y^{H(ID_p, R_{ID_p}) + H(ID_u, R_{ID_u}) H(T_s, T_e, R_T)}\right) \tag{5}$$

If (5) holds, output "1"; otherwise, output "0."

(7) ProxyUpdate

(a) When the proxy ID_p is revoked by the user ID_u before the expiration of the valid time period, the cloud will put this revoked proxy's identity ID_p in a public revocation list. Then, the user ID_u authorizes a new proxy $ID_{p'}$ and sets the warrant's valid time period for this proxy $ID_{p'}$ and then computes the time private key based on the warrant's valid time period

and generates a warrant-signature pair for the new proxy $ID_{p'}$. Finally, the user ID_u sends all the above messages to the new proxy $ID_{p'}$. The proxy $ID_{p'}$ can validate the correctness of the messages he received and next compute the proxy private key based on the partial private key generated by the PKG and the time private key.

(i) The user ID_u sets the warrant's start time T_s and end time T_e. The user ID_u selects $r_T \in_R Z_p$ and computes $R_T = g^{r_T}$ and $\sigma_T = r_T + \sigma_{ID_u} \cdot H(T_s, T_e, R_T)$. Let (R_T, σ_T) be the time private key.

(ii) The user ID_u generates a warrant m_w, which is used to authorize a new proxy $ID_{p'}$. The warrant m_w consists of the user's identity ID_u, the proxy's identity $ID_{p'}$, and the warrant's start time T_s and end time T_e. Let the warrant be denoted as $m_w = ID_u \parallel ID_{p'} \parallel T_s \parallel T_e$. The user ID_u picks $r_1 \in_R Z_p^*$ and computes the warrant m_w's signature as follows: $R_1 = g^{r_1}$ and $\sigma_1 = r_1 + \sigma_{ID_u} \cdot H(m_w, R_1)$.

(iii) The user ID_u sends the warrant's valid time period (T_s, T_e), the time private key (R_T, σ_T), and the warrant-signature pair $(m_w, (R_1, \sigma_1))$ to the proxy $ID_{p'}$.

(iv) The proxy $ID_{p'}$ verifies the validity of warrant-signature pair $(m_w, (R_1, \sigma_1))$ by (2). Then, the proxy $ID_{p'}$ checks the correctness of time private key (R_T, σ_T) by (3) and calculates the proxy private key $(R_{ID_{p'}}, R_T, \sigma)$ based on the time private key (R_T, σ_T) and the partial private key $(R_{ID_{p'}}, \sigma_{ID_{p'}})$, where $\sigma = (\sigma_{ID_{p'}} + \sigma_T)(mod p)$.

(b) When the proxy ID_p's warrant expires, the user ID_u can select to continue employing this proxy. The user ID_u generates a new warrant for the proxy ID_p by updating the warrant's start time and end time. The operations are similar to the above case (a) in *ProxyUpdate* algorithm.

When the proxy is revoked, the cloud will put this revoked proxy's identity in a public revocation list. It makes the revoked proxy unable upload new file to the cloud any more. In addition, when the proxy's warrant expires, he does not have the new valid warrant and the expired warrant he keeps cannot pass the verification of (2) in *ProxyKeyGen* algorithm. Therefore, neither the revoked proxy nor the expired proxy can upload a new file to the cloud any more.

Remark 3. If a proxy is in the public revocation list, he can be deleted from this revocation list when his warrant expires. Thus, the size of public revocation list will not increase unlimitedly.

(8) Payment. When the proxy ID_p's warrant expires or the proxy ID_p is revoked, the user ID_u will pay for the proxy ID_p based on his workload.

(a) The proxy ID_p requests the cloud to send him the signature of the newest upload number N.

(b) Upon receiving the request, the cloud computes the signature of the newest upload number as $\beta = SSig'_{sk_c}(N \parallel ID_p)$ and returns β to the proxy ID_p.

(c) The proxy ID_p uses the cloud identity to check the validity of signature β by inputting his identity ID_p and the upload number N he records and then shows the signature β from the cloud and the newest upload number N to the user ID_u.

(d) The user ID_u verifies whether the signature β is valid or not via the cloud identity ID_c. When this signature is valid, the user ID_u pays for the proxy ID_p based on the newest upload number N.

5. Security Analysis

Theorem 4 (correctness). *Our proposed scheme satisfies the following properties:*

(1) *Private key correctness: If the private keys and the partial private key generated by the PKG are correct, then these private keys and the partial private key are able to pass the checking of the user, the cloud, and the proxy, respectively.*

(2) *Warrant-signature pair correctness: If the warrant-signature pair generated by the user is valid, then this warrant-signature pair is able to pass the proxy's checking.*

(3) *Time private key correctness: If the time private key generated by the user is correct, then this time private key is able to pass the proxy's verification.*

(4) *Auditing correctness: If the auditing proof generated by the cloud is valid, this proof is able to pass the TPA's verification.*

Proof.

(1) Given a correct private key $sk_{ID_u} = (R_{ID_u}, \sigma_{ID_u})$ generated by the PKG, the verification equation (1) in *Extract* algorithm can be presented as follows:

$$g^{\sigma_{ID_u}} = g^{r_{ID_u} + xH(ID_u, R_{ID_u})} = g^{r_{ID_u}} \cdot g^{xH(ID_u, R_{ID_u})} = R_{ID_u} Y^{H(ID_u, R_{ID_u})} \quad (6)$$

Similarly, if the private key $sk_{ID_c} = (R_{ID_c}, \sigma_{ID_c})$ and the partial private key $sk_{ID_p} = (R_{ID_p}, \sigma_{ID_p})$ generated by the PKG are correct, we can ensure that the private key sk_{ID_c} and the partial private key sk_{ID_p} are able to pass the verification of the cloud and the proxy, respectively.

(2) Given a valid warrant-signature pair $(m_w, (R_1, \sigma_1))$ generated by the user ID_u, the verification equation (2) in *ProxyKeyGen* algorithm holds.

$$\begin{aligned} g^{\sigma_1} &= g^{r_1+\sigma_{ID_u}\cdot H(m_w,R_1)} = g^{r_1}\cdot g^{\sigma_{ID_u}\cdot H(m_w,R_1)} \\ &= R_1\cdot g^{(r_{ID_u}+xH(ID_u,R_{ID_u}))\cdot H(m_w,R_1)} \\ &= R_1\cdot g^{r_{ID_u}\cdot H(m_w,R_1)}\cdot g^{xH(ID_u,R_{ID_u})\cdot H(m_w,R_1)} \\ &= R_1 R_{ID_u}^{H(m_w,R_1)} Y^{H(ID_u,R_{ID_u})\cdot H(m_w,R_1)} \end{aligned} \tag{7}$$

(3) Given a time private key $sk_{ID_c} = (R_{ID_c}, \sigma_{ID_c})$ generated by the user ID_u, the verification equation (3) in *ProxyKeyGen* algorithm can be presented as follows:

$$\begin{aligned} g^{\sigma_T} &= g^{r_T+\sigma_{ID_u}\cdot H(T_s\|T_e,R_T)} = g^{r_T}\cdot g^{\sigma_{ID_u}\cdot H(T_s\|T_e,R_T)} \\ &= R_T\cdot g^{(r_{ID_u}+xH(ID_u,R_{ID_u}))\cdot H(T_s\|T_e,R_T)} \\ &= R_T\cdot g^{r_{ID_u}\cdot H(T_s\|T_e,R_T)}\cdot g^{xH(ID_u,R_{ID_u})\cdot H(T_s\|T_e,R_T)} \\ &= R_T R_{ID_u}^{H(T_s\|T_e,R_T)} Y^{H(ID_u,R_{ID_u})\cdot H(T_s\|T_e,R_T)} \end{aligned} \tag{8}$$

(4) Given a valid proof $P = \{\mu, \Omega\}$ generated by the cloud, the verification equation (5) in *ProofVerify* algorithm can be proved as follows:

$$\begin{aligned} e(\Omega, g) &= e\left(\prod_{i\in I}\beta_i^{\nu_i}, g\right) \\ &= e\left(\prod_{i\in I}\left(h\left(name\left\|ID_p\right\|i\right)\cdot u^{m_i}\right)^{\sigma\cdot\nu_i}, g\right) \\ &= e\left(\prod_{i\in I}\left(h\left(name\left\|ID_p\right\|i\right)\cdot u^{m_i}\right)^{\nu_i}, g^{\sigma}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot\prod_{i\in I}u^{m_i\cdot\nu_i}, g^{\sigma}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\sum_{i\in I}\nu_i\cdot m_i}, g^{\sigma}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, g^{\sigma_{ID_p}+\sigma_T}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, \right. \\ &\quad \left. g^{r_{ID_p}+xH(ID_p,R_{ID_p})+r_T+\sigma_{ID_u}H(T_s\|T_e,R_T)}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, R_{ID_p}\cdot Y^{H(ID_p,R_{ID_p})}\right. \\ &\quad \left.\cdot R_T\cdot g^{\sigma_{ID_u}\cdot H(T_s\|T_e,R_T)}\right) = e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\right. \\ &\quad \left.\cdot u^{\mu}, R_{ID_p}\cdot Y^{H(ID_p,R_{ID_p})}\cdot R_T\right. \\ &\quad \left.\cdot g^{(r_{ID_u}+xH(ID_u,R_{ID_u}))\cdot H(T_s\|T_e,R_T)}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, R_{ID_p}\cdot Y^{H(ID_p,R_{ID_p})}\right. \\ &\quad \left.\cdot R_T\cdot R_{ID_u}^{H(T_s\|T_e,R_T)}\cdot Y^{H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \\ &= e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, R_{ID_p}\cdot R_T\right. \\ &\quad \left.\cdot R_{ID_u}^{H(T_s\|T_e,R_T)}\cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \end{aligned} \tag{9}$$

□

Theorem 5 (auditing soundness). *In our proposed scheme, the cloud cannot pass the validation of the TPA if it does not keep the user's data correctly.*

Proof. We construct a knowledge extractor and utilize the method of knowledge proof to accomplish the following proof. If the cloud does not keep the user's data correctly but can pass the validation of the TPA, then we can repeatedly interact between the proposed scheme and the knowledge extractor to extract the intact challenged data blocks. We will prove this theorem by a sequence of games.

Game 1. The adversary sends a query to the challenger for obtaining the signatures of a series of data blocks. The challenger generates the corresponding signatures for these data blocks and sends them to the adversary. Then the challenger submits an auditing challenge $chal = \{i, v_i\}_{i\in I}$ to the adversary. The adversary generates the auditing proof $P = \{\mu, \Omega\}$ and sends it to the challenger.

Game 2. Game 2 is identical to Game 1, with one difference. That is the challenger keeps a list of responses to the queries from the adversary. The challenger observes each instance of the challenge-respond process with the adversary. If the challenger finds the aggregated signature Ω is not equal to $\prod_{i\in I}\beta_i^{\nu_i}$, he declares failure and aborts.

Analysis. Assume that the auditing proof $\{\mu, \Omega\}$ is generated by the honest prover based on the correct file F. From the correctness of our scheme, we get

$$\begin{aligned} e(\Omega, g) = e\left(\prod_{i\in I}h\left(name\left\|ID_p\right\|i\right)^{\nu_i}\cdot u^{\mu}, R_{ID_p}\cdot R_T\right. \\ \left.\cdot R_{ID_u}^{H(T_s\|T_e,R_T)}\cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \end{aligned} \tag{10}$$

Assume that the forged auditing proof $\{\mu', \Omega'\}$ is generated by the adversary based on the corrupted file F', where $F' \neq F$. Because the forgery is successful, we get

$$e\left(\Omega', g\right) = e\left(\prod_{i\in I} h\left(name \left\| ID_p \right\| i\right)^{v_i} \cdot u^{\mu'}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \tag{11}$$

Obviously, $\mu' \neq \mu$; otherwise $\Omega' = \Omega$, which contradicts our above assumption. We define $\Delta\mu = \mu - \mu'$ and show a simulator that could break the challenge CDH instance with this adversary as follows.

Given $(g, g^{\varepsilon}, \varphi) \in G_1$, the simulator outputs φ^{ε}. Set $u = g^a\varphi^b$, where $a, b \in Z_p^*$ are two random elements chosen by the simulator. Meanwhile, the verification value is set as $Y = g^{\varepsilon}$.

The simulator selects a random element $r_i \in Z_p^*$ for each i $(1 \le i \le n)$ in the challenge and programs the random oracle at i as

$$H\left(name \left\| ID_p \right\| i\right) = \frac{g^{r_i}}{\left(g^{am_i} \cdot \varphi^{bm_i}\right)} \tag{12}$$

The simulator can compute β_i, since we get

$$\begin{aligned} H\left(name \left\| ID_p \right\| i\right) \cdot u^{m_i} &= \frac{g^{r_i}}{\left(g^{am_i} \cdot \varphi^{bm_i}\right)} \cdot u^{m_i} \\ &= \frac{g^{r_i}}{\left(g^{am_i} \cdot \varphi^{bm_i}\right)} \cdot \left(g^a\varphi^b\right)^{m_i} \\ &= \frac{g^{r_i}}{\left(g^{am_i} \cdot \varphi^{bm_i}\right)} \cdot \left(g^{am_i} \cdot \varphi^{bm_i}\right) = g^{r_i} \end{aligned} \tag{13}$$

The simulator calculates $\beta_i = (h(name\|i\|ID_p)u^{m_i})^{\sigma} = (g^{r_i})^{\sigma}$.

Dividing (10) by (11), we have

$$\begin{aligned} e\left(\frac{\Omega}{\Omega'}, g\right) &= e\left(u^{\Delta\mu}, R_{ID_p} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)}\right) \\ &= e\left(\left(g^a\varphi^b\right)^{\Delta\mu}, g^{r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \\ &= e\left(g^{a\Delta\mu}, g^{r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \\ &\quad \cdot e\left(\varphi^{b\Delta\mu}, g^{r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \\ &= e\left(g^{a\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))} \cdot Y^{a\Delta\mu\cdot(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T))}, g\right) \\ &\quad \cdot e\left(\varphi^{b\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))}, g\right) \\ &\quad \cdot e\left(\varphi^{b\Delta\mu}, Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right). \end{aligned} \tag{14}$$

So, we obtain

$$\begin{aligned} &e\left(\frac{\Omega}{\Omega'} \cdot g^{-a\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))} \cdot Y^{-a\Delta\mu\cdot(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T))} \cdot \varphi^{-b\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))}, g\right) \\ &= e\left(\varphi^{b\Delta\mu}, Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right)\Big) \\ &= e\left(\varphi, Y\right)^{(b\Delta\mu(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)))} \\ &= e\left(\varphi, g^{\varepsilon}\right)^{(b\Delta\mu(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)))} \\ &= e\left(\varphi^{\varepsilon}, g\right)^{(b\Delta\mu(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)))}. \end{aligned} \tag{15}$$

Then, we can learn that

$$\varphi^{\varepsilon} = e\left(\frac{\Omega}{\Omega'} \cdot g^{-a\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))} \cdot Y^{-a\Delta\mu\cdot(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T))} \cdot \varphi^{-b\Delta\mu\cdot(r_{ID_p}+r_T+r_{ID_u}H(T_s\|T_e,R_T))}\right)^{1/(b\Delta\mu(H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)))}. \tag{16}$$

Note that the probability we can find a solution to CDH problem is the same as the probability $b\Delta\mu(H(ID_p, R_{ID_p}) + H(ID_u, R_{ID_u})H(T_s \| T_e, R_T)) \neq 0 \bmod p$. The probability of $b\Delta\mu(H(ID_p, R_{ID_p}) + H(ID_u, R_{ID_u})H(T_s \| T_e, R_T)) \neq 0 \bmod p$ is $1 - 1/p$. Then, we can find a solution to CDH problem with a probability $1 - 1/p$, which contradicts the assumption that the CDH problem in G_1 is hard.

Therefore, if the difference between the adversary's probabilities of success in Game 1 and Game 2 is nonnegligible, we can construct a simulator that utilizes the adversary to solve the CDH problem.

Game 3. Game 3 is identical to Game 2, with one difference. The challenger still maintains and observes each instance of the proposed scheme. For one of these instances, if the

challenger finds a linear combination μ' of data blocks is not equal to the expected μ, the challenger declares failure and aborts.

Analysis. Assume that the honest prover generates a corret auditing proof $\{\mu, \Omega\}$ based on the correct file F. From the correctness of our scheme, we get

$$e(\Omega, g) = e\left(\prod_{i\in I} h\left(name \| ID_p \| i\right)^{v_i} \cdot u^{\mu}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \tag{17}$$

Assume that the adversary generates a forged auditing proof $\{\mu', \Omega'\}$ based on the corrupted file F', where $F' \neq F$. Because the forgery is successful, we get

$$e(\Omega', g) = e\left(\prod_{i\in I} h\left(name \| ID_p \| i\right)^{v_i} \cdot u^{\mu'}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \tag{18}$$

According to Game 2, we learn that $\Omega' = \Omega$. Define $\Delta\mu = \mu - \mu'$, and we show a simulator that could solve the challenge DL instance with this adversary as follows.

Given $(g, \varphi) \in G_1$, the simulator would like to calculate a value x which satisfies $\varphi = g^x$. The simulator randomly selects two elements $a, b \in Z_p^*$ and sets $u = g^a\varphi^b$.

Based on the above two verification equations, we get

$$\begin{aligned} &e\left(\prod_{i\in I} h\left(name \| ID_p \| i\right)^{v_i} \cdot u^{\mu}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) = e(\Omega, g) \\ &= e(\Omega', g) = e\left(\prod_{i\in I} h\left(name \| ID_p \| i\right)^{v_i} \cdot u^{\mu'}, R_{ID_p} \cdot R_T \cdot R_{ID_u}^{H(T_s\|T_e,R_T)} \cdot Y^{H(ID_p,R_{ID_p})+H(ID_u,R_{ID_u})H(T_s\|T_e,R_T)}\right) \end{aligned} \tag{19}$$

Therefore, we obtain

$$u^{\mu} = u^{\mu'}, \tag{20}$$

and therefore that

$$1 = u^{\Delta\mu} = \left(g^a\varphi^b\right)^{\Delta\mu} = g^{a\Delta\mu} \cdot \varphi^{b\Delta\mu}. \tag{21}$$

Since $\varphi = g^x$, we can find the solution to the DL problem by calculating

$$\varphi = g^{-a\Delta\mu/b\Delta\mu}, \tag{22}$$

and then $x = -a\Delta\mu/b\Delta\mu$. As defined above, $\Delta\mu \neq 0$. However, b is zero with the probability $1/p$, which is negligible. Then, we can solve the DL problem with a probability of $1 - 1/p$, which contradicts the assumption that the DL problem in G_1 is hard.

Therefore, if the difference between the adversary's probabilities of success in Game 2 and Game 3 is nonnegligible, we can construct a simulator that utilizes the adversary to solve the DL problem.

From the above analysis, we can know that the malicious cloud cannot pass the validation of the TPA if it does not keep the user's data correctly. □

Theorem 6 (the security of proxy update). *In our proposed scheme, the revoked proxy or the expired proxy cannot process data on behalf of the user any more.*

Proof. When the revoked proxy or the expired proxy uploads the file and its corresponding signatures to the cloud, the cloud firstly checks whether this proxy is in the public revocation list and then verifies the validity of the warrant-signature pair $(m_w, (R_1, \sigma_1))$ and the file tag tag.

Firstly, if this proxy is in the public revocation list, the cloud rejects this file and signatures from this proxy. Secondly, the cloud checks the validity of the warrant-signature pair $(m_w, (R_1, \sigma_1))$ by the verification equation (2). If the current time period is not in the valid time period (T_s, T_e), then the warrant-signature pair cannot pass the verification. The cloud regards this proxy as a revoked proxy or an expired proxy and refuses this proxy's request. Finally, the cloud checks the validity of the file tag by verifying whether $SSig_{ssk}(name \parallel T_s \parallel T_e \parallel R_{ID_p} \parallel R_T \parallel ID_p)$ is a valid signature or not. If it is not, the cloud considers that this file tag is generated by a revoked proxy or an expired proxy and then rejects the request of this proxy.

Therefore, only the proxy, who is not in the public revocation list and possess a valid warrant, is able to process data on behalf of the user. □

Theorem 7 (the soundness of payment). *In our proposed scheme, the user will pay for the proxy based on the proxy's real workload assuming the cloud does not collude with the proxy.*

Proof. Firstly, when the cloud receives the data uploaded by the proxy, it will set the upload number N and generate the signature β with its private key for this upload number N. Therefore, the signature β that the proxy sends to the user is generated by the cloud according to the real upload number N. The proxy cannot forge this signature.

Secondly, the proxy can verify whether the signature β from the cloud is valid according to his real upload number N. Thus, the signature β that the proxy sends to the user is recognized by the proxy.

In conclusion, the user believes the signature β sent by the proxy is valid. The payment that the user pays for the proxy is based on the proxy's real workload in the absence of collusion. □

6. Performance Evaluation

In this section, we first present the functionality comparison of our scheme with different cloud storage auditing schemes and then compare the computation overhead and the communication overhead of our scheme with that of Wang et al. scheme [5]. At last, we show the experimental results of our scheme.

6.1. Functionality Comparison. Table 2 shows a functionality comparison of our scheme with different cloud storage auditing schemes [5, 6, 18, 25, 30, 31] in terms of public verifiability, lightweight computation, proxy update, certificate management simplification, and workload-based payment. Our scheme is the only scheme that meets all of the aforementioned properties.

6.2. Performance Analysis and Comparison

(1) Computation Overhead. Since the scheme of [5] is the lightweight cloud storage auditing scheme which achieves public verifiability and certificates management simplification, we compare our scheme with it with regard to the computation overhead of different entities in Table 3. In our scheme, the main cost of the PKG is generating global parameters, the partial private key and private keys for the proxy, the user, and the cloud, respectively. Therefore, the computation overhead of the PKG is $4E_{G_1} + 3(A_{G_1} + M_{Z_p^*} + H_{Z_p^*})$, where E_{G_1} and A_{G_1}, respectively, denote one exponentiation operation and one addition operation in G_1, and $M_{Z_p^*}$ and $H_{Z_p^*}$, respectively, denote one multiplication operation and one hashing operation in Z_p^*. The dominated computation overhead of the proxy for computing proxy private key and data signatures is $A_{Z_p^*} + n(M_{G_1} + 2E_{G_1} + H_{G_1})$, where $A_{Z_p^*}$ denotes one addition operation in Z_p^*, M_{G_1} and H_{G_1}, respectively, denote one multiplication operation and hash operation in G_1, and n is the total number of data blocks. When generating a warrant-signature pair and the time private key, the cost of the user is $2(A_{Z_p^*} + M_{Z_p^*} + H_{Z_p^*})$, which can be done offline. The computation overhead of the cloud for generating an auditing proof is $(c-1)Mul_{G_1} + cExp_{G_1} + cMul_{Z_p^*} + (c-1)Add_{Z_p^*}$, where c is the number of challenged data blocks. The computation overhead of the TPA mainly comes from verifying the correctness of the auditing proof, which needs $2P + cH_{G_1} + (c+4)E_{G_1} + (c+4)M_{G_1} + M_{Z_p^*} + 4H_{Z_p^*}$. P denotes one pairing operation.

When the scheme is built from the bilinear parings, the computation overhead of the scheme mainly comes from the exponentiation operation, the pairing operation, and the multiplication operation in G_1. The other operations, such as the operation in Z_p^* and hashing operation, cost less computation overhead. Thus, comparing with the scheme of [5], our scheme and the scheme of [5] cost almost the same computation overhead on the proxy, the user, and the cloud sides. In order to generate the cloud private key and perform the auditing task, our scheme needs to cost more computation overhead than the scheme of [5] on the PKG and the cloud sides. What is more, our scheme supports proxy update and workload-based payment, while the scheme of [5] cannot support that.

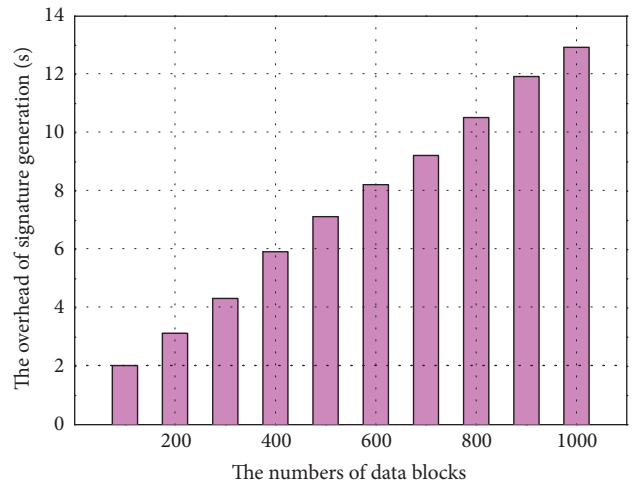

Figure 4: The computation overhead for generating data signatures.

(2) Communication Overhead. As introduced in Section 4, the communication overhead of the proposed scheme mainly comes from the TPA and the cloud. We can see from Table 4, for an auditing challenge $chal = \{i, v_i\}_{i \in I}$, the communication overhead of the TPA in our scheme is $c \cdot (|n| + |p|)$ bits, where $|n|$ is the size of an element of set $[1, n]$, and $|p|$ is the size of an element in Z_p^*. In the scheme of [5], the communication overhead of the TPA is $|c| + 2|p|$ bits for an auditing challenge $chal = \{c, k_1, k_2\}$, where $k_1, k_2 \in Z_p^*$ are random values. For generating an auditing proof $P = \{\lambda, \sigma\}$, the communication overhead of the cloud in our scheme and the scheme of [5] both are $|p| + |q|$ bits, where $|q|$ is the size of an element in G_1.

6.3. Experimental Results. In this subsection, we conduct experiments on the proposed scheme by utilizing the GNU Multiple Precision Arithmetic (GMP) [33] and the Pairing-Based Cryptography (PBC) Library [34]. These experiments are implemented on a Linux machine with an Intel Pentium 2.30GHz processor and 8GB memory and coded based on C programming language. In experiments, we set the size of an element in Z_p^* to be $|p|$ =160 bits, the base field size to be 512 bits, and the size of data file to be 20MB composed by 1,000,000 blocks.

(1) Performance of Signature Generation. In our scheme, the proxy helps the user to generate data signatures. To evaluate the performance of signature generation, we implement the experiment by increasing the number of data blocks n from 100 to 1000. From Figure 4, it is easily observed that the time of generation signatures linearly increases with the number of data blocks. Thus, we can conclude that our scheme greatly alleviates the user's computation burden for generating data signatures.

(2) Performance of Auditing. The performance of auditing on the TPA side and that on the cloud side are, respectively,

Table 2: Functionality comparison of different cloud storage auditing schemes.

Schemes	Public verifiability	Lightweight computation	Proxy update	Certificate management simplification	Workload-based payment
Wang et al. [5]	Yes	Yes	No	Yes	No
Shen et al. [6]	Yes	Yes	No	No	No
Yu et al. [18]	Yes	No	No	No	No
Zhang et al. [25]	Yes	No	No	Yes	No
Shacham et al. [30]	Yes	No	No	No	No
Wang et al. [31]	No	No	No	No	No
Ours	Yes	Yes	Yes	Yes	Yes

Table 3: The computation overhead of our scheme and Wang et al. scheme [5] for different entities.

Entity	Computation overhead (Our scheme)	Computation overhead (Wang et al.[5])
PKG	$4E_{G_1} + 3(A_{G_1} + M_{Z_p^*} + H_{Z_p^*})$	$3E_{G_1} + 2(A_{G_1} + M_{Z_p^*} + H_{Z_p^*})$
Proxy	$A_{Z_p^*} + n(M_{G_1} + 2E_{G_1} + H_{G_1})$	$A_{Z_p^*} + M_{Z_p^*} + H_{Z_p^*} + n(M_{G_1} + 2E_{G_1} + H_{G_1})$
User	$2(A_{Z_p^*} + M_{Z_p^*} + H_{Z_p^*})$	$A_{Z_p^*} + M_{Z_p^*} + H_{Z_p^*}$
Cloud	$(c-1)Mul_{G_1} + cExp_{G_1} + cMul_{Z_p^*} + (c-1)Add_{Z_p^*}$	$(c-1)Mul_{G_1} + cExp_{G_1} + cMul_{Z_p^*} + (c-1)Add_{Z_p^*}$
TPA	$2P + cH_{G_1} + (c+4)E_{G_1} + (c+4)M_{G_1} + M_{Z_p^*} + 4H_{Z_p^*}$	$2P + (c+2)H_{G_1} + (c+3)E_{G_1} + (c+3)M_{G_1} + 3H_{Z_p^*} + 2A_{Z_p^*}$

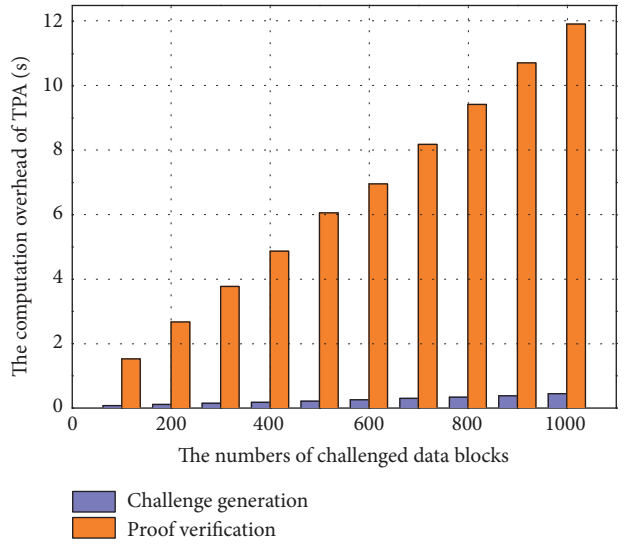

Figure 5: The computation overhead of the TPA in the phase of auditing.

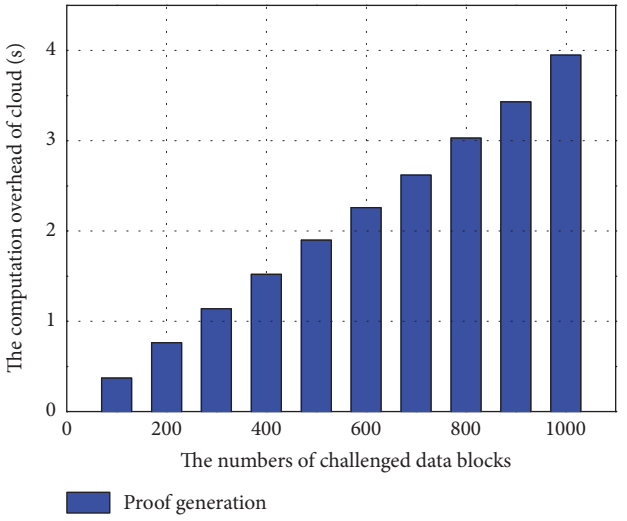

Figure 6: The computation overhead of the cloud in the phase of auditing.

shown in Figure 5 and Figure 6. In these experiments, we select to challenge different data blocks from 100 to 1000 increased by an interval of 100. As shown in Figure 5, the TPA's computation overhead for generating challenge and verifying proof both grow linearly as the number of challenged data blocks. The computation overhead for verifying proof ranges from 1.512s to 11.931s, while the computation overhead for generating challenge grows slowly, just ranging from 0.043s to 0.422s. In Figure 6, it can be seen that the cloud's computation overhead for generating proof ranges from 0.371s to 3.958s. From the above analysis, we can conclude that the more data blocks are challenged, the more computation overhead need to be spent on the TPA and the cloud sides.

Table 4: The communication overhead of our scheme and Wang et al. scheme [5].

Entity	Phase	Communication overhead (Our scheme)	Communication overhead (Wang et al. [5])
TPA	Auditing challenge	$c \cdot (\|n\| + \|p\|)$	$\|c\| + 2\|p\|$
Cloud	Auditing proof	$\|p\| + \|q\|$	$\|p\| + \|q\|$

7. Conclusion

In this paper, we propose an identity-based cloud storage auditing scheme, which achieves lightweight computation on the user side by introducing a proxy and supports proxy update and workload-based payment for the proxy. In our scheme, the task of generating data signatures is executed by the proxy with a valid warrant. The revoked proxy and the proxy with expired warrant cannot help the user process data any more. We pay for the proxy based on the workload in cloud storage auditing. The security analysis and the experiment results show that our scheme provides strong security with desirable efficient efficiency.

Data Availability

No data were used to support this study.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

This research is supported by National Natural Science Foundation of China (61772311) and the Open Project of the State Key Laboratory of Information Security, Institute of Information Engineering, Chinese Academy of Sciences (2017-MS-05).

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, 2012.

[2] G. Ateniese, R. Burns, R. Curtmola et al., "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 598–609, Virginia, Va, USA, November 2007.

[3] Y. Zhang, H. Zhang, R. Hao, and J. Yu, "Authorized identity-based public cloud storage auditing scheme with hierarchical structure for large-scale user groups," *China Communications*, vol. 15, no. 11, pp. 111–121, 2018.

[4] 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. 1-1, 2018.

[5] 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, 2016.

[6] W. Shen, J. Yu, H. Xia, H. Zhang, X. Lu, and R. Hao, "Light-weight and privacy-preserving secure cloud auditing scheme for group users via the third party medium," *Journal of Network and Computer Applications*, vol. 82, pp. 56–64, 2017.

[7] B. Wang, S. S. M. Chow, M. Li, and H. Li, "Storing shared data on the cloud via security-mediator," in *Proceedings of the 2013 IEEE 33rd International Conference on Distributed Computing Systems, ICDCS 2013*, pp. 124–133, USA, July 2013.

[8] 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.

[9] A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07)*, pp. 584–597, ACM, Alexandria, Va, USA, November 2007.

[10] M. Sookhak, A. Gani, M. K. Khan, and R. Buyya, "Dynamic remote data auditing for securing big data storage in cloud computing," *Information Sciences*, vol. 380, pp. 101–116, 2017.

[11] 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.

[12] B. Wang, B. Li, and H. Li, "Panda: public auditing for shared data with efficient user revocation in the cloud," *IEEE Transactions on Services Computing*, vol. 8, no. 1, pp. 92–106, 2015.

[13] 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.

[14] 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.

[15] C. Liu, J. Chen, L. T. Yang et al., "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, 2014.

[16] H. Tian, Y. Chen, C.-C. Chang et al., "Dynamic-hash-table based public auditing for secure cloud storage," *IEEE Transactions on Services Computing*, vol. 10, no. 5, pp. 701–714, 2017.

[17] J. Yu, R. Hao, H. Xia, H. Zhang, X. Cheng, and F. Kong, "Intrusion-resilient identity-based signatures: concrete scheme in the standard model and generic construction," *Information Sciences*, vol. 442/443, pp. 158–172, 2018.

[18] 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, 2017.

[19] W. Shen, G. Yang, J. Yu, H. Zhang, F. Kong, and R. Hao, "Remote data possession checking with privacy-preserving authenticators for cloud storage," *Future Generation Computer Systems*, vol. 76, pp. 136–145, 2017.

[20] G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, "Enabling public auditing for shared data in cloud storage supporting

identity privacy and traceability," *The Journal of Systems and Software*, vol. 113, pp. 130–139, 2016.

[21] 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*, 2017.

[22] 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, 2014.

[23] Y. Yu, M. H. Au, G. Ateniese et al., "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, 2017.

[24] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Transactions on Dependable and Secure Computing*, vol. 16, no. 1, pp. 72–83, 2019.

[25] 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*, 2018.

[26] 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, 2018.

[27] 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, 2016.

[28] Y. Wang, Q. Wu, B. Qin, S. Tang, and W. Susilo, "Online/offline provable data possession," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 5, pp. 1182–1194, 2017.

[29] B. Wang, H. Li, and M. Li, "Privacy-preserving public auditing for shared cloud data supporting group dynamics," in *Proceedings of the 2013 IEEE International Conference on Communications, ICC 2013*, pp. 1946–1950, Hungary, June 2013.

[30] H. Shacham and B. Waters, "Compact proofs of retrievability," *Journal of Cryptology. The Journal of the International Association for Cryptologic Research*, vol. 26, no. 3, pp. 442–483, 2013.

[31] H. Wang, "Proxy provable data possession in public clouds," *IEEE Transactions on Services Computing*, vol. 6, no. 4, pp. 551–559, 2013.

[32] Q.-A. Wang, C. Wang, K. Ren, W.-J. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 22, no. 5, pp. 847–859, 2011.

[33] "The gnu multiple precision arithmetic library (gmp)," http://gmplib.org.

[34] B. Lynn, *The Pairing-Based Cryptographic Library*, 2015, https://crypto.stanford.edu/pbc/.