future internet

MDPI

Article

Certificateless Provable Group Shared Data Possession with Comprehensive Privacy Preservation for Cloud Storage

Hongbin Yang, Shuxiong Jiang *, Wenfeng Shen and Zhou Lei

School of Computer Engineering and Science, Shanghai University, Shanghai 200444, China; hbyoungshu@staff.shu.edu.cn (H.Y.); wfshen@mail.shu.edu.cn (W.S.); leiz@shu.edu.cn (Z.L.)
* Correspondence: clydebear@163.com

Received: 5 May 2018; Accepted: 5 June 2018; Published: 7 June 2018

Abstract: Provable Data Possession (PDP) protocol makes it possible for cloud users to check whether the cloud servers possess their original data without downloading all the data. However, most of the existing PDP schemes are based on either public key infrastructure (PKI) or identity-based cryptography, which will suffer from issues of expensive certificate management or key escrow. In this paper, we propose a new construction of certificateless provable group shared data possession (CL-PGSDP) protocol by making use of certificateless cryptography, which will eliminate the above issues. Meanwhile, by taking advantage of zero-knowledge protocol and randomization method, the proposed CL-PGSDP protocol leaks no information of the stored data and the group user's identity to the verifiers during the verifying process, which is of the property of comprehensive privacy preservation. In addition, our protocol also supports efficient user revocation from the group. Security analysis and experimental evaluation indicate that our CL-PGSDP protocol provides strong security with desirable efficiency.

Keywords: cloud storage; provable data possession; privacy-preserving; certificateless cryptography

1. Introduction

In recent years, cloud computing [1] has received considerable attention from research communities in academia, as well as industry. As an important part of cloud computing, cloud storage has become a popular choice for people to deploy their data storage, which brings a number of benefits. Users are relieved of the burden of the management of a great deal of data. Furthermore, universal data access with independent geographical location is highly convenient.

However, a number of vulnerabilities that led to various attacks have left many potential users worried [2]. Thus, many researchers focus on creating trusted cloud services that provide the necessary security guarantees. Santos et al. [3] presented a trusted cloud-computing platform (TCCP), which offers a closed box execution environment for IaaS services. TCCP guarantees confidential execution of guest virtual machines. It also enables customers to attest to the IaaS provider and to determine if the service is secure before their VMs are launched into the cloud. In 2016, Paladi et al. [4] presented a security framework for cloud infrastructure. This framework included a trusted VM launching and a domain-based stored data protection protocols. The trusted VM launching protocol is used before deploying guest VMs, and trust is established by the remotely attesting host platform configuration. Meanwhile, the domain-based storage protection protocol ensures data confidentiality in remote storage by using cryptographic methods.

Furthermore, unlike traditional storage, the cloud stored data is outside of the control of the users, which entails security risks in terms of confidentiality, integrity, and availability of data and service [5].

One of the major concerns of cloud users is the integrity of their outsourced data. Moreover, the cloud server may not be fully trusted to report incidents of data loss in order to protect its reputation [6–8]. As a result, it is necessary for cloud users to periodically check whether their outsourced data are stored properly. Provable Data Possession (PDP) scheme is a primitive one that can be used to convince cloud users that their data are kept intact.

As is well known, it is impractical for cloud users to frequently download all the cloud data due to the expensive cost of bandwidth. Additionally, the traditional methods for data integrity checking, like hash function and Message Authorization Code (MAC), cannot be applied directly, because the cloud server may only store the hash code of the original data in order to reduce storage costs. In 2007, Ateniese et al. [9] proposed a provable data possession scheme to check the integrity of data, which employed RSA algorithm to construct homomorphic verifiable authenticators of the data blocks, meaning that the cloud servers were able to prove the data integrity with low communication overheads and computational costs. After that, many researchers proposed corresponding system models and security models based on the first PDP scheme. Later, the POR model was introduced in Juel et al. [10], which used an error-correcting code to establish a sentinel-based POR scheme. Shacham et al. [11] developed a proof of retrievability scheme based on the BLS signature [12], which not only eliminates the constraint on checking times, but also shortens the size of authenticator. In order to support the dynamic data operation on cloud data, Ateniese et al. [13] presented a scalable and efficient provable data possession scheme based on hash functions and symmetric key encryptions. The limitation is that this scheme cannot support data block insertion. Erway et al. [14] proposed a full-dynamic PDP Scheme by utilizing the authenticated flip table. Similarly, Cash et al. [15] proposed a dynamic POR scheme that relies on oblivious RAM protocols. Wang et al. [16] made further improvements to the previous dynamic PDP schemes by using Merkle hash tree (MHT). Liu et al. [17] introduced a top-down, levelled, multi-replica, MHT-based data auditing scheme for dynamic big data storage in the cloud, which supports fully dynamic data updates and authentication of block indices.

Public verifiability is one of the most important properties for PDP schemes, which means that external verifiers are able to check the integrity of cloud data. With this practical property, cloud users can delegate the rights to check data possession to a third-party auditor (TPA) to do the periodical checking job. However, data privacy may be leaked to the TPA during the checking process, which may cause financial loss for the users who have stored confidential or sensitive data on cloud servers. Recently, a number of schemes [16,18–22] have been developed that allow a TPA to check the integrity of the stored data. Wang et al. [20] proposed a privacy-preserving public cloud data auditing system by combining the public key-based homomorphic authenticator with random masking. This scheme explained the definition of data privacy against the TPA. Wang et al. [21] performed further study on data privacy preservation and proposed the notion of 'zero-knowledge public auditing' to defeat off-line guessing threat. Yu et al. [22] enhanced the privacy of remote data integrity-checking protocol for secure cloud storage. The aforementioned schemes [16,18–22] all work only in the public key infrastructure (PKI) [23]-based system, which may suffer from heavy public key management. Yu et al. [24] further presented an identity-based PDP scheme to eliminate heavy public key management. This scheme leaks no information of the stored data to the TPA, but cannot protect the privacy of the user's identity. If the data is shared by a group of users, such as a company, the TPA will know all the details of the group users' identities.

In order to reduce the complexity of the PDP scheme, many researchers [24–27] have focused on studying identity-based PDP schemes. By utilizing identity-based cryptography (IBC) [28], there is no need for a PKI to perform complex certificate management such as distribution, storage, revocation, and verification. Unfortunately, IBC has an inherent drawback of key escrow. Wang et al. [29] first proposed a certificateless public auditing mechanism for verifying data integrity in the cloud in order to eliminate the problem of key escrow. In this scheme, Key Generation Center will generate only the partial key, so that in any case it will not compromise the user's private key. Li et al. [30] introduced a

certificateless PDP scheme for shared group data, but it lost the privacy preservation of cloud data and the user's identity to the TPA.

Motivation: In this paper, we mainly focus on preserving the privacy of group shared data against the third-party auditor (TPA) during the integrity checking process. Suppose there is a group of users from one company who share company data on a given cloud server. Each user of the group can upload and share their data on the cloud server. The manager of the group requires a TPA to periodically check the integrity of the outsourced data, but he does not allow the TPA to extract any information related to their data, not even the identity of the group users. Users of the group may leave the company, so the problem of user revocation from the group needs to be considered. Thus, we need a primitive to meet such requirements and to guarantee the integrity of the outsourced data on the cloud server.

Our contribution: The contributions of this paper are summarized as follows:

- First, we propose a new PDP protocol (CL-PGSDP) for group shared data by utilizing certificateless cryptography [31], which eliminates the problems of certificate management and key escrow.
- Second, by making use of the idea of zero-knowledge proof protocol, the equality of discrete logarithm [32–34], and randomization method, we construct a privacy-preserving CL-PGSDP protocol. On the one hand, our protocol leaks no information of the group shared data to the TPA. On the other hand, all the data blocks are signed by group users to get corresponding authentication tags, and the TPA cannot learn any identity information from the challenged data block during the auditing process.
- Third, based on CDH and DL assumptions, we provide detailed security proofs of our new protocol. Additionally, our protocol supports efficient group user revocation. We perform some experiments and show the practicality of our protocol.

Organization: The rest of the paper is organized as follows: In Section 2, we review some preliminaries used in CL-PGSDP construction. In Section 3, we formalize the system model and security model of CL-PGSDP protocol. We describe the concrete construction of the CL-PGSDP protocol in Section 4. We formally prove the correctness, soundness, and comprehensive privacy preservation of our protocol in Section 5. We report the performance and implementation results in Section 6. Section 7 concludes our paper.

2. Preliminaries

In this section, we review some preliminaries knowledge used in this paper, including bilinear pairing, certificateless Cryptography, zero-knowledge proof, and Complexity assumption.

2.1. Bilinear Pairing

Denote G_1 and G_2 as two multiplicative groups with the prime order q. g is a generator of group G_1. A function $e : G_1 \times G_1 \rightarrow G_2$ is called a bilinear pairing [12] if it has the following properties:

- Bilinearity: For all $u, v \in G_1$. and $a, b \in Z_q$, $e(u^a, v^b) = e(u, v)^{ab}$ holds.
- Non-Degeneracy: $e(g, g) \neq 1_{G_2}$, in which 1_{G_2} is the identity of G_2.
- Efficient Computation: $e(u, v)$ can be computed efficiently for all. $u, v \in G_1$.

2.2. Certificateless Cryptography

A Certificateless Cryptography (CLC) [31] scheme is specified by seven randomized algorithms.

1. Setup: This algorithm takes security parameter k as input and returns the system parameters *params* and master-key.
2. Partial-Private-Key-Extract: This algorithm takes *params*, master-key, and entity's ID as inputs and returns a partial private key sk_{ID} for the entity.

3. Set-Secret-Value: This algorithm takes *params* and entity's ID as inputs and outputs this entity's secret value y_{ID}.
4. Set-Private-Key: This algorithm takes *params*, entity's ID, partial private key sk_{ID}, and secret value y_{ID} as inputs and outputs private key S_{ID}.
5. Set-Public-Key: This algorithm takes *params* and secret value y_{ID} as inputs and outputs public key Y_{ID}.
6. Encrypt: This algorithm takes *params*, message m, and public key Y_{ID} as inputs and entity's ID and generates ciphertext σ of the message m if success.
7. Decrypt: This algorithm takes *params*, σ, and S_{ID} as inputs and returns message m.

By making use of certificateless cryptography, we will construct new authenticator of data block.

2.3. Zero-Knowledge Proof

Zero-knowledge proofs [33] are defined as those proofs that convey no additional knowledge other than the correctness of the proposition in question. Here, we introduce one of the zero knowledge protocol: equality of discrete logarithm [34]. Let G be a finite cyclic group with the prime order q; g_1, g_2 are generators of G. The protocol shows that a prover (P) can prove to a verifier (V) that $\log_{g_1} Y_1 = \log_{g_2} Y_2$, $Y_1, Y_2 \in G$ without leaking the secret key to V.

1. P randomly chooses $\rho \in Z_q$, and computes $X_1 = g_1^{\rho}$, $X_2 = g_2^{\rho}$, then sends X_1, X_2 to V.
2. V also randomly chooses $v \in \{0,1\}^{\lambda}$ and sends v to P.
3. P computes $y = \rho - vx (\mathrm{mod} q)$, in which x is the secret key of P, and returns y to V.
4. V accepts the proof if and only if $X_1 = g_1^{y} Y_1^{v}$ and $X_2 = g_2^{y} Y_2^{v}$.

2.4. Security Assumption

Discrete Logarithm (DL) problem [35]: G_1 is a multiplicative cyclic group, g is a generator of G_1. Given $(g, g^a) \in G_1$, compute a.

Definition 1 (DL Assumption). *For any probabilistic polynomial time (PPT) algorithm A, the advantage for A to solve the DL problem in G_1 is negligible, which can be defined as* $Adv_A^{Dl} = Pr\left[A(g, g^a) = a : a \overset{R}{\leftarrow} Z_p^*\right] \leq \varepsilon$.

Computational Deffie-Hellman (CDH) Problem [36]: G_1 is a multiplicative cyclic group, g is a generator of G_1. Given $(g, g^a, g^b) \in G_1$, compute g^{ab}.

Definition 2 (CDH Assumption). *For any PPT algorithm A, the advantage for A to solve the CDH problem in G_1 is negligible, which can be defined as* $Adv_A^{CDH} = Pr\left[A(g, g^a, g^b) = g^{ab} : a, b \overset{R}{\leftarrow} Z_p^*\right] \leq \varepsilon$.

The ε denotes a negligible value in the above definitions.

3. System Model and Security Model

In this section, we introduce the system model and security model of CL-PGSDP protocol.

3.1. CL-PGSDP System

The system model of our scheme is composed of three different entities: user group, cloud service provider (CSP), and the third party public auditor (TPA). Figure 1 illustrates the relationships and interactions among the three entities of the system. The user group includes numbers of users, who have large amount of data to be stored on cloud without keeping a local copy (each user is able to upload, access, and update the outsourced group shared data). We suppose one of the users is the group manager, who sets up the system and generates system parameters. The CSP has significant storage space and computation resources and provides data storage services for cloud users. The CSP

could be semi-trusted and might even hide data corruption incidents to cloud users to maintain their good reputation. The TPA has expertise and capabilities to be delegated by the cloud users to check the data possession of the cloud, but the TPA is also curious in the sense that he is willing to learn some information during the data integrity checking procedure.

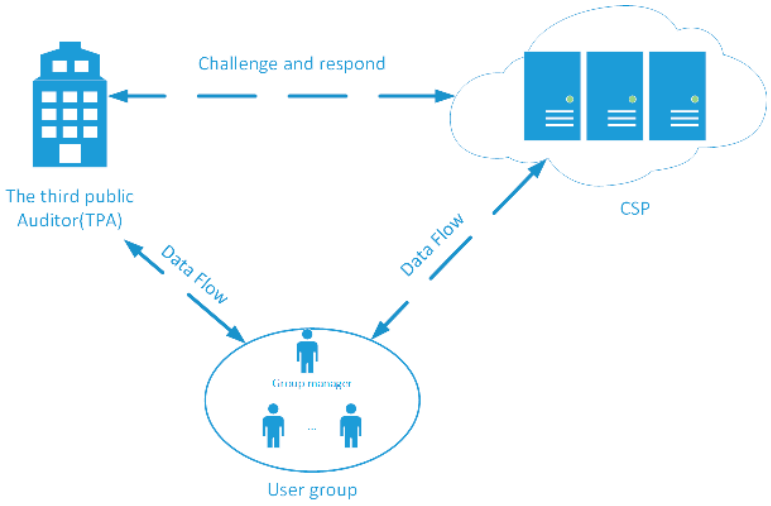

Figure 1. System model of our protocol.

3.2. System Components

Eight algorithms are involved in CL-PGSDP system.

1. Setup is a probabilistic algorithm run by the group manager. It takes a security parameter λ as input and outputs the system parameters *params* and the master key *msk*.
2. Partial-Private-Key-Gen is a probabilistic algorithm run by the group manager. It takes the master key *msk*, a random value δ, and the identity $ID_i \in \{0,1\}^*$ of the user u_i as inputs, and outputs the u_i's partial private key psk_{ID_i}.
3. Secret-Value-Gen is a probabilistic algorithm run by the group user who randomly selects y_{ID_i} as the secret value. Thus, the private key of the group user contains two parts: secret value y_{ID_i} and partial private key psk_{ID_i}.
4. Public-Key-Gen is a probabilistic algorithm performed by the group user u_i to compute the public key. It inputs the u_i's secret value y_{ID_i} and outputs the u_i's public key pk_{ID_i}.
5. Tag-Gen is a probabilistic algorithm executed by the group user u_i to generate authentication tags for data blocks. It takes the u_i's partial private key psk_{ID_i}, the secret value y_{ID_i}, and the data block m_j as inputs, and outputs the tag σ_j of m_j.
6. Challenge is a randomized algorithm run by the TPA. It takes the system parameters *params*, a unique file name, and the count c of the challenged data blocks as inputs, and outputs the challenge information *chal*.
7. Proof-Gen is a probabilistic algorithm run by cloud server to obtain a data possession proof P of the challenged blocks. The inputs include *chal*, the challenged data blocks and tags of the challenged data blocks.
8. Proof-Check is a deterministic algorithm run by the TPA. It inputs the proof P, the challenge information *chal*, and the user's public key. If P is correct, this algorithm outputs 1, otherwise it outputs 0.

3.3. System Security

We consider three security properties, namely, completeness, soundness, and comprehensive privacy preservation against the TPA in our CL-PGSDP protocol.

- Completeness means the cloud server can pass the possession checking procedure as long as the cloud server properly stores the group shared data.
- Comprehensive privacy preservation means that the TPA achieves no information on the data blocks and the user's identity during the integrity checking procedure.
- Soundness states that whenever the cloud server convinces a TPA to accept its proof, the cloud server should actually store the challenged data blocks. According to certificateless cryptography [30,31], we consider three types of probabilistic polynomial-time (PPT) adversaries, namely, A_1, A_2, A_3, and a challenger C in our security model and define the security of our protocol by three games. The details are as follows:

Game 1: This game is played by challenger C and adversary A_1 who wants to substitute the user's public key with any other value, but A_1 cannot access the master key of the system.

Setup: Challenger C runs the Setup algorithm to obtain the system parameters *params* and the master secret key *msk*, and forwards *params* to the adversary A_1, while keeps *msk* confidential.

Queries: A_1 can adaptively issue the following queries to C. C maintains the corresponding query lists, which are initially empty, and responds to the queries to A_1 as follows.

(1) Hash Query. A_1 makes hash function queries to C for any identity ID, and C responds to the hash values to A_1.
(2) Partial Private Key Query. A_1 adaptively chooses different ID and summits it to C for querying the partial private key of the ID. C executes the Partial-Private-Key-Gen algorithm to obtain the partial private key for the ID and sends it to A_1.
(3) Secret Value Query. A_1 adaptively chooses different ID and summits it to C for querying the secret value of the ID. C runs the Secret-value-Gen algorithm to generate the secret value for the ID and sends it to A_1.
(4) Public Key Query. A_1 adaptively chooses different ID and summits it to C for querying the public key of the ID. C performs the algorithm Public-key-Gen to compute the public key for the ID and sends it to A_1.
(5) Public Key Replacement. A_1 can repeatedly select a value to replace the public key of any ID.
(6) Tag Query. A_1 adaptively chooses the tuple (ID, m) and submits it to C for querying the tag of the data block m. C runs Tag-Gen algorithm to generate the tag of data block m and sends it to A_1.

Forge: Finally, A_1 outputs a forged tag σ for the m with the identity ID and the public key pk_{ID}. If the forged tag σ is valid after the above queries, then A_1 wins the game.

Game 2: This game is played by challenger C and adversary A_2 who is able to get the master key but cannot substitute the group user's public key.

Setup: Challenger C runs the Setup algorithm to obtain the system parameters *params* and the master secret key *msk*, and forwards *params* and *msk* to the adversary A_2.

Queries: A_2 can make a number of queries to C adaptively. C maintains the corresponding query lists, which are initially empty, and responds to the queries to A_2 as follows.

(1) Hash Query. A_2 makes hash function queries to C for any identity ID, and C responds the hash values to A_2.
(2) Secret Value Query. A_2 adaptively chooses different ID and summits it to C for querying the secret value of the ID. C runs the Secret-value-Gen algorithm to generate the secret value for the ID and sends it to A_2.

(3) Public Key Query. A_2 adaptively chooses different ID and summits it to C for querying the public key of the ID. C performs the algorithm Public-key-Gen to compute the public key for the ID and sends it to A_2.

(4) Tag Query. A_2 adaptively chooses the tuple (ID, m) and submits it to C for querying the tag of the data block m generated by the ID. C runs Tag-Gen algorithm to generate the tag of data block m and sends it to A_2.

Forge: Finally, A2 outputs a forged tag σ for the m with the identity ID.
If the forged tag σ is valid after the above queries, then A_2 wins the game.

Definition 3. *A CL-PGSDP scheme is secure against adaptive impersonation and forging tag attacks if any PPT adversary A (A_1 or A_2) who plays the above games with the challenger C has only negligible probability ε of winning the games, that is,*

$$\Pr(A_{win}) \leq \varepsilon$$

in which the probability ε is taken over all coin tosses made by A and C.

Game 3: This game is played by challenger C and adversary A_3 who aims to forge the data integrity proof to cheat the TPA. A_3 is regarded as the untrusted CSP. From the Definition 3, we know that it is hard to forge the tag of single data block. Thus, we will focus on the issue of whether A_3 can forge the integrity proof without correct data to pass the challenge.

Setup: Challenger C runs the Setup algorithm to obtain the system parameters *params*, the master secret key *msk*, and partial private key for all users, and only forwards *params* to the adversary A_3.

Tag Queries: A_3 adaptively chooses the tuple (ID, m) and sends it to C for querying the tag of data block m. C runs the algorithm Tag-Gen to generate the tag of m and returns it to A_3.

Challenge: C makes a random challenge *chal* to A_3 and requests A_3 to provide the corresponding data possession proof for *chal*.

Forge: For the challenge *chal*, A_3 generates a proof and sends it to C. If the proof can pass the integrity verification while A_3 does not possess the correct data, A_3 wins the game.

Definition 4. *A CL-PGSDP scheme is secure against adaptive impersonation and forging proof attacks if any PPT adversary A who plays Game 3 with the challenger C has only negligible probability ε of winning the games, that is,*

$$\Pr(A_{win}) \leq \varepsilon$$

in which the probability ε is taken over all coin tosses made by A and C.

4. Our Construction

In this section, we provide a concrete construction of certificateless provable group shared data possession protocol supporting comprehensive privacy preservation for cloud storage. We suppose the number of users in the group is z, and ID_i represent the unique identity of the user u_i, in which $1 \leq i \leq z$. Without losing the generality, we set u_1 as the group manager who will set up system and generate system parameters and the partial private keys for other users. The group shared data M is split into n blocks, denoted as $M = \left\{ m_j \middle| 1 \leq j \leq n, m_j \in Z_q^* \right\}$. In the Partial-Private-Key-Gen algorithm, we employ short signature [12] and randomization method to produce the partial private key for each user of the group. In the Tag-Gen algorithm, we take the advantage of the idea of certificateless cryptography [31] to construct the tags of the data blocks using the partial private key and the secret value. In the challenge phase, the TPA randomly chooses some indexes of the data blocks and corresponding random values as a challenge to the CSP. In the proof generating phase, the CSP computes a response to the TPA. We utilize the idea of zero knowledge proof [33,34] to design

the details of the interaction between the TPA and the CSP. The details of the proposed protocol are as follows:

- Setup. This algorithm is run by u_1. On input of security parameter λ, u_1 chooses two cyclic multiplicative groups, G_1 and G_2, with prime order q, $\log_2 q \leq \lambda$. g is a generator of G_1. There exists a bilinear map $e : G_1 \times G_1 \to G_2$. u_1 selects three secure hash functions $H_1, H_2 : \{0,1\}^* \to G_1$, $H_3 : G_2 \to \{0,1\}^l$, a pseudo-random permutation (PRP) $\pi : Z_q^* \times \{1,\ldots,\mathrm{n}\} \to \{1,\ldots,\mathrm{n}\}$, and a pseudo-random function (PRF) $\phi : Z_q^* \times Z_q^* \to Z_q^*$. u_1 initializes a public log file LF, which is used to record the information of the indexes of the data blocks and the information of the corresponding tag generators. u_1 randomly chooses $x \in Z_q^*$ as master secret key msk and $\delta \in Z_q^*$ as secret value, and computes $P_{pub} = g^x$. u_1 keeps the master secret key msk and δ privately, and publishes the system parameters $params = \left\{G_1, G_2, P_{pub}, H_1, H_2, H_3, e, g, q, LF, \pi, \phi\right\}$.
- Partial-Private-Key-Gen. This algorithm is run by u_1. When receiving the identity $ID_i \in \{0,1\}^*$ of the user u_i, u_1 computes the u_i's partial private key $psk_{ID_i} = \mathrm{H}_1(ID_i + \delta)^x$ and sends psk_{ID_i} and $\mathrm{H}_1(ID_i + \delta)$ to u_i.
- Secret-Value-Gen. This algorithm is run by group user. u_i randomly selects $y_{ID_i} \in Z_q^*$ as the secret value and keeps it privately.
- Public-Key-Gen. This algorithm is run by group user. u_i uses the secret value to compute the public key $Y_{ID_i} = g^{y_{ID_i}}$.
- Tag-Gen. Each user in group can generate tags of data blocks using partial private key and secret value. Suppose user u_i $(0 < i \leq z)$ generates an authentication tag for data block m_j $(0 < j \leq n)$. It takes u_i's partial private key psk_{ID_i}, the secret value s_{ID_i}, and the data block m_j as inputs and outputs the tag σ_j of m_j. The equation for computing tag is $\sigma_j = {psk_{ID_i}}^{m_j} \cdot H_2(\omega_j)^{y_{ID_i}}$, in which $\omega_j = j||fname$, j is the index of data block m_j, and $fname$ denotes the unique identity of data block m_j. Each time the u_i generates a tag for data block m_j, u_i will update the information in public log file LF with the index j of m_j, Y_{ID_i}, and $H_1(ID_i + \delta)$. Actually, LF is a table, and one line of it can be showed as follows:

j	$H_1(ID_i + \delta)$	Y_{ID_i}

- The user u_i uploads the data blocks and its tags to the CSP. The CSP can check the validation of each tag using the following equation:

$$e(\sigma_j, g) = e(H_1(ID_i + \delta)^{m_j}, P_{pub}) \cdot e(H_2(\omega_j), Y_{ID_i}) \tag{1}$$

- Challenge. This algorithm is run by the TPA, who randomly picks c-element subset J of the set $[1, n]$ by pseudo-random permutation (PRP) π; each element in J denotes the index of the challenged data block. The TPA chooses a random element $v_j \in Z_q^*$ for each element in J by pseudo-random function (PRF) ϕ. Let Q be the set $\left\{(j, v_j)\right\}_{j \in \mathrm{J}}$. To generate a challenge, the TPA will search the log file LF according the set J to get the information $\{(j, H_1(ID_i + \delta))\}$. The TPA picks a random value $t \in Z_q^*$ as secret value and computes $T_1 = g^t$, $T_{2j} = e(H(ID_i + \delta), P_{pub})^t$. Let $T_2 = \left\{T_{2j}\right\}, j \in J$. The TPA sends the $chal = \{Q, T_1, T_2\}$ to the server.
- Proof-Gen. Upon receiving the $chal = \{Q, T_1, T_2\}$, the CSP computes $\mu = \prod_{j \in J} {T_{2j}}^{v_j m_j}$, $\sigma = \prod_{j \in J} {\sigma_j}^{v_j}$ and $proof = H_3(e(\sigma, T_1) \cdot \mu^{-1})$, then sends $proof$ as a response to the $chal$ from the TPA.

- Proof-Check. Upon receiving the $proof$ from the CSP, the TPA first searches the publish log file LF to get the information $\{(j, Y_{ID_i})\}$ and checks the equation:

$$proof = H_3(\prod_{\substack{j \in J \\ \mathrm{j} \to i}} e(H_2(j||fname)^{v_j}, Y_{ID_i}^t))$$

in which $j \to i$ means the information of u_i can be find from public log file LF by the index j of data block m_j. If the equation holds, the TPA accepts the proof; otherwise, the proof is invalid. The process of Challenge, Proof-Gen, and Proof-Check are summarized as Figure 2.

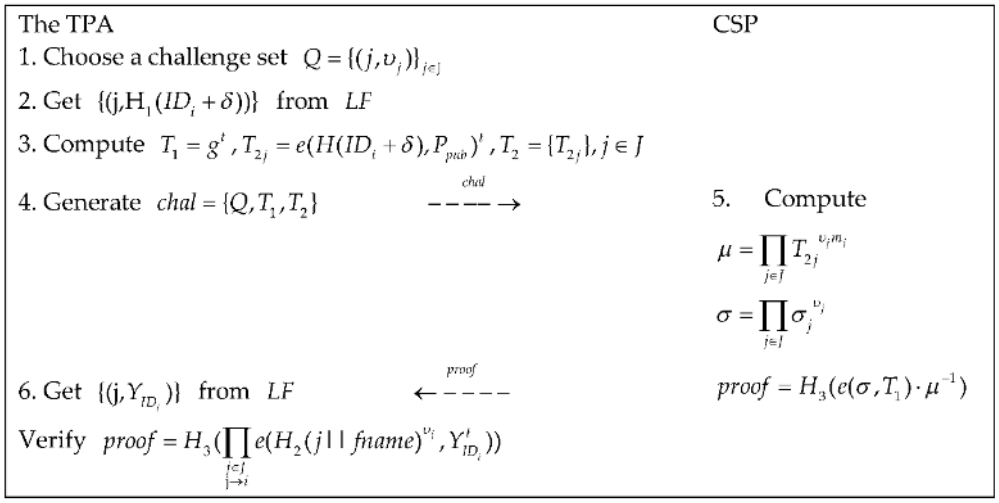

Figure 2. CL-PGSDP protocol.

- Revocation-Tag-Gen. If user u_m, $2 \le m \le z$ leaves the group, and user u_n will be the successor of u_m. The following procedure will efficiently update the tags generated by u_m. It needs u_m, u_n and the CSP online simultaneously.

  (1) The CSP randomly selects $\alpha \in Z_q^*$, and sends it to u_n.

  (2) Upon receiving α, u_n computes $(K_1 = (psk_{ID_n})^{\frac{1}{y_{ID_n}}}, K_2 = \alpha \cdot y_{ID_n})$ and sends (K_1, K_2) to u_m.

  (3) u_m computes $(R_1 = \frac{K_1^{y_{ID_m}}}{sk_{ID_m}}, R_2 = \frac{K_2}{y_{ID_m}})$ and sends it to the CSP.

  (4) When receiving (R_1, R_2), the CSP computes $R_3 = \frac{R_2}{\alpha} = \frac{y_{ID_n}}{y_{ID_m}}$. The CSP will update the tag of the data block $m_{i'}$ by computing the equation $\sigma'_{i'} = (R_1{}^{m_{i'}} \cdot \sigma_{i'})^{R_3}$, in which $\sigma_{i'}$ is the tag generated by u_m. The proof of the correctness of algorithm Revocation-Tag-Gen is as follows:

$$\begin{aligned}\sigma'_{i'} &= (R_1{}^{m_{i'}} \cdot \sigma_{i'})^{R_3} = \left(\left(\frac{sk_{ID_n}^{\frac{y_{ID_m}}{y_{ID_n}}}}{sk_{ID_m}}\right)^{m_{i'}} \cdot sk_{ID_m}{}^{m_{i'}} \cdot H_2(\omega_{i'})^{y_{ID_m}}\right)^{\frac{y_{ID_n}}{y_{ID_m}}} \\ &= \left(\left(sk_{ID_n}^{\frac{y_{ID_m}}{y_{ID_n}}}\right)^{m_{i'}} \cdot H_2(\omega_{i'})^{y_{ID_m}}\right)^{\frac{y_{ID_n}}{y_{ID_m}}} = sk_{ID_n}{}^{m_{i'}} \cdot H_2(\omega_{i'})^{y_{ID_n}}\end{aligned}$$

5. Security Analysis of the New Protocol

In this section, we show that our protocol is secure with the properties of completeness, soundness, and comprehensive privacy preservation.

5.1. Completeness

If the CSP properly stores data, it can always pass the verification. The completeness of the protocol can be demonstrated as follows:

$$
\begin{aligned}
proof \quad &= H_3(e(\sigma, T_1) \cdot \mu^{-1}) = H_3\left(\frac{e(\sigma,T_1)}{\prod\limits_{j\in J} T_{2j}^{v_j m_j}}\right) = H_3\left(\frac{e(\sigma,T_1)}{\prod\limits_{\substack{j\in J\\ j\to i}} e(H_1(ID_i+\delta),P_{pub})^{tv_j m_j}}\right) \\
&= H_3\left(\frac{e(\prod\limits_{j\in I}\sigma_j^{v_j},T_1)}{\prod\limits_{\substack{j\in J\\ j\to i}} e(psk_{ID_i},T_1)^{v_j m_j}}\right) = H_3\left(\prod_{\substack{j\in J\\ j\to i}} \frac{e(\sigma_j^{v_j},T_1)}{e(psk_{ID_i}^{m_j},T_1^{v_j})}\right) = H_3\left(\prod_{\substack{j\in J\\ j\to i}} e(\frac{\sigma_j}{psk_{ID_i}^{m_j}},T_1^{v_j})\right) \\
&= H_3\left(\prod_{\substack{j\in J\\ j\to i}} e(H_2(\omega_j)^{y_{ID_i}},g^{tv_j})\right) = H_3\left(\prod_{\substack{j\in J\\ j\to i}} e(H_2(\omega_j)^{v_j},g^{ty_{ID_i}})\right) = H_3\left(\prod_{\substack{j\in J\\ j\to i}} e(H_2(\omega_j)^{v_j},Y_{ID_i}^{t})\right)
\end{aligned}
$$

5.2. Soundness

Theorem 1. *In the random oracle model, if a PPT adversary A_1 wins Game 1 defined in Section 3 with non-negligible probability ε, then there is an algorithm B that can solve the CDH problem.*

Proof of Theorem 1. Algorithm B is given $(g, g^a, g^b) \in G_1$; its goal is to output $g^{ab} \in G_1$. Algorithm B simulates the challenger and interacts with as A_1 follows.

Setup: B produces the system parameters *params*, secret key $\delta \in Z_q^*$, and sets $P = g^a$, while a keeps unknown.

H_1-Query: At any time, A_1 can query the random oracle H_1. To respond to these queries, algorithm B maintains a list of tuples (ID, k_1, K) as Tab_1. When A_1 queries the oracle H_1 at the identity ID', Algorithm B responds as follows:

(1) If $ID' \in Tab_1$, then algorithm B retrieves the tuple (ID', k_1', K') and responds with K' to A_1.
(2) Otherwise, B picks a random $k_1' \in Z_q^*$ and computes $K' = g^{bk_1'}$. Then, it adds the tuple (ID', k_1', K') to Tab_1 and responds with K' to A_1.

PartialKey-Query: At any time, A_1 can query partial key for any identity ID'. If $ID' \notin Tab_1$, B makes the H_1-Query. Otherwise, B maintains a list of tuples (ID, psk) as Tab_2. When A_1 queries the oracle PartialKey-Query at the identity, Algorithm B responds as follows:

(1) If $ID' \in Tab_2$, then algorithm B retrieves the tuple (ID', psk') and responds psk' to A_1.
(2) Otherwise, B computes $psk' = (K')^a$. Then, it adds the tuple (ID', psk') to Tab_2 and responds psk' to A_1.

SecretValue-Query: At any time, A_1 can query secret value for any identity ID', if $ID' \notin Tab_1$ or $ID' \notin Tab_2$. B firstly makes the H_1-Query or PartialKey-Query for the identity ID'. Then, B randomly chooses a value $x' \in Z_q^*$ as response to A_1.

PublicKey-Query: At any time, A_1 can query public key for any identity ID', if $ID' \notin Tab_1$ or $ID' \notin Tab_2$. B first makes the H_1-Query or PartialKey-Query for the identity ID'. Then, B computes $PK' = g^{x'}$, in which x' is the secret value from SecretValue-Query and responds PK' to A_1.

H_2-Query: At any time, A_1 can query the random oracle H_2 for ω'. Algorithm B also maintains a list of tuples (ω, W) as Tab_3. If $\omega' \in Tab_3$, then algorithm B retrieves the tuple (ω', W') and responds W' to A_1. Otherwise, B randomly selects $k_2 \in Z_q^*$ and computes $W' = g^{k_2}$. Then, it adds the tuple (ω', W') to Tab_3 and responds W' to A_1.

Tag-Query: At any time, A_1 can query tag with (m', ID'). B first checks whether $ID' \in Tab_1$, $ID' \in Tab_2$, and $\omega' \in Tab_3$. If not, B will compute corresponding tuple and update Tab_1, Tab_2, and Tab_3. After that, B can get corresponding information from Tab_1, Tab_2, and Tab_3 and compute the tag T' for (ω', m', ID') by the algorithm Tag-Gen and returns it to A_1.

Forge: Eventually, A_1 outputs $(ID', K', psk', x', PK', W', m', T')$. T' is the forged tag of the data block m' on the identity ID' with the public key PK'.

Analysis: If A_1 wins Game 1, on the one hand, B can get

$$e(T', g) = e(H_1(ID' + \delta)^{m'}, P) \cdot e(H_2(\omega'), PK')$$

according to the verification Equation (1). On the other hand, B can retrieve $H_1(ID' + \delta) = g^{bk_1'}$ from Tab_1 and $H_2(\omega') = g^{k_2}$ from Tab_3. Thus, B gets $e(T', g) = e(g^{bk_1' m'}, g^a) \cdot e(g^{k_2}, PK')$. Finally, we can derive that

$$g^{ab} = \left(\frac{T'}{(PK')^{k_2}} \right)^{\frac{1}{k_1' m'}}$$

which means that B can solve the CDH problem with non-negligible probability ε. However, according to CDH assumption, the advantage for B to solve the CDH problem in G_1 is negligible. Thus, A_1 cannot win Game 1. This completes the proof. □

Theorem 2. *In the random oracle model, if a PPT adversary A_2 wins Game 2 defined in Section 3 with non-negligible probability ε, then algorithm B can solve the CDH problem.*

Proof of Theorem 2. Algorithm B is given $(g, g^a, g^b) \in G_1$; its goal is to output $g^{ab} \in G_1$. Algorithm B simulates the challenger and interacts with as A_2 follows.

Setup: B produces the system parameters *params*, secret key $\delta \in Z_q^*$, and sets $P = g^s$, in which s is master key. B sends *params*, δ, s, and P to A_2.

H_1-Query: At any time, A_2 can query the random oracle H_1. To respond to these queries, algorithm B maintains a list of tuples (ID, l_1, L) as Tab_1. When A_2 queries the oracle H_1 at the identity ID', Algorithm B responds as follows:

(1) If $ID' \in Tab_1$, then algorithm B retrieves the tuple (ID', l_1', L') and responds with L' to A_2.
(2) Otherwise, B picks a random $l_1' \in Z_q^*$ and computes $L' = g^{l_1'}$. Then, it adds the tuple (ID', l_1', L') to Tab_1 and responds with L' to A_2.

SecretValue-Query: Because A_2 knows the master key, there is no need for A_2 to query the partial key. At any time, A_2 can query secret value for any identity ID'. If $ID' \notin Tab_1$, B makes the H_1-Query. Otherwise, B maintains a list of tuples (ID, y, Y) as Tab_2. When A_2 queries the oracle SecretValue-Query at the identity ID', Algorithm B responds as follows:

(1) If $ID' \in Tab_2$, then algorithm B retrieves the tuple (ID', y', Y') and responds with y' to A_2.
(2) Otherwise, B randomly selects $l_2 \in Z_q^*$ and makes $y' = l_2$. Then, it adds the tuple (ID', y') to Tab_2 and responds with y' to A_2.

PublicKey-Query: At any time, A_2 can query public key for any identity ID', if $ID' \notin Tab_1$ or $ID' \notin Tab_2$. B first makes the H_1-Query or PartialKey-Query for the identity ID'. Then, B computes

$Y' = g^{ay'} = g^{al_2}$, in which y' is the secret value from SecretValue-Query, and updates Tab_2. B responds with Y' to A_2.

H_2-Query: At any time, A_2 can query the random oracle H_2 for ω'. Algorithm B also maintains a list of tuples (ω, W) as Tab_3. If $\omega' \in Tab_3$, then algorithm B retrieves the tuple (ω', W') and responds W' to A_2. Otherwise, B randomly selects $l_3 \in Z_q^*$ and computes $W' = g^{bl_3}$. Then, it adds the tuple (ω', W') to Tab_3 and responds with W' to A_2.

Tag-Query: At any time, A_2 can query tag with (m', ID', ω'). B first checks whether $ID' \in Tab_1$, $ID' \in Tab_2$, and $\omega' \in Tab_3$. If not, B will compute corresponding tuple and updates Tab_1, Tab_2, and Tab_3. After that, B can get corresponding information from Tab_1, Tab_2, and Tab_3, and computes the tag T' for (ω', m', ID') using the algorithm Tag-Gen and returns it to A_2.

Forge: Eventually, A_2 outputs (T', ω', m', ID'). T' is the forged tag of the data block m' on the identity ID'.

Analysis: If A_2 wins Game 2, on the one hand, B can get

$$e(T', g) = e(H_1(ID' + \delta)^{m'}, P) \cdot e(H_2(\omega'), Y')$$

according to the verification Equation (1). On the other hand, B can retrieve $H_1(ID' + \delta) = g^{{l_1}'}$ from Tab_1, $Y' = g^{a{l_2}'}$ from Tab_2, and $H_2(\omega') = g^{bl_3}$ from Tab_3. Thus, B gets $e(T', g) = e(g^{{l_1}'m'}, P) \cdot e(g^{bl_3}, g^{al_2})$. Finally, we can derive that

$$g^{ab} = (T')^{\frac{1}{l_1' l_2 l_3 \mathrm{sm}'}}$$

which means that B can solve the CDH problem with non-negligible probability ε. However, according to CDH assumption, the advantage for B to solve the CDH problem in G_1 is negligible. Thus, A_2 cannot win Game 2. This completes the proof. □

Theorem 3. *If the DL assumption holds, the adversary A_3 wins Game 3 only at negligible probability.*

Proof of Theorem 3. Let the challenge information be $chal = \{Q, T_1, T_2\}$. If A_3 outputs $proof'$ and wins Game 3 at non-negligible probability, we can get the verification equation:

$$proof' = H_3(e(\sigma', T_1) \cdot {\mu'}^{-1}) = H_3(\prod_{j \in I} e(H_2(j||fname)^{v_j}, Y_{ID_i}^t))$$

in which σ' is the forged tag for the forged data block m' and μ' is produced by A_3. Assume the real proof is $proof$ and the corresponding information is (σ, μ). We also get the verification equation:

$$proof = H_3(e(\sigma, T_1) \cdot \mu^{-1}) = H_3(\prod_{j \in I} e(H_2(j||fname)^{v_j}, Y_{ID_i}^t))$$

Thus, we can derive from the above two verification equations that

$$\frac{e(\sigma, T_1)}{e(\sigma', T_1)} = \frac{\mu}{\mu'}$$

Because A_3 wins the Game 3, there exists $\sigma = \sigma'$ and at least one data block $m_j \neq {m_j}'$. Suppose $m_j - {m_j}' = \Delta m_j$. Then, we get $\frac{\mu}{\mu'} = 1$, which is $\frac{\prod_{j \in I} {T_{2j}}^{v_j m_j}}{\prod_{j \in I} {T_{2j}}^{v_j {m_j}'}} = 1$. We can get $\prod_{j \in I} {T_{2j}}^{v_j \Delta m_j} = 1$. Based on this conclusion, the DL problem can be solved as follows: Given two elements $g, y \in G_1$ in

which $y = g^a$, we will compute $a \in {Z_q}^*$. We randomly select $\alpha_j, \beta_j \in Z_q^*$ and let ${T_{2j}}^{v_j} = \mathrm{X}_j = g^{\alpha_j} y^{\beta_j}$. We can get following equation:

$$\prod_{j \in I} {T_{2j}}^{v_j \Delta m_j} = \prod_{j \in I} {X_j}^{\Delta m_j} = \prod_{j \in I} \left(g^{\alpha_j} y^{\beta_j}\right)^{\Delta m_j} = g^{\sum\limits_{j \in I} \alpha_j \Delta m_j} y^{\sum\limits_{j \in I} \beta_j \Delta m_j} = 1$$

Then, we can derive $y = g^{-\frac{\sum\limits_{j \in I} \alpha_j \Delta m_j}{\sum\limits_{j \in I} \beta_j \Delta m_j}}$. Since $\Delta m_j \neq 0$, β_j is the random value from ${Z_q}^*$, so the probability of $\sum\limits_{j \in I} \beta_j \Delta m_j = 0$ is only $\frac{1}{q}$. Therefore, we can output the right value of a with non-negligible probability $1 - \frac{1}{q}$. This completes the proof. □

5.3. Comprehensive Privacy Preservation

5.3.1. Data Privacy Preservation

Upon receiving the challenge from the TPA, the cloud server responds with the proof: $proof = H_3(e(\sigma, T_1) \cdot \mu^{-1})$, in which we hide the information of σ and μ using hash function H_3. Furthermore, the TPA just needs to check the following equation: $proof = H_3(\prod\limits_{\substack{j \in J \\ \mathrm{j} \to i}} e(H_2(j||fname)^{v_j}, Y_{ID_i}^t))$, without knowing any information about data file blocks $\{m_i\}$ or their corresponding tags $\{\sigma_i\}$.

5.3.2. User Identity Privacy Preservation

In CL-PGSDP protocol, we design a log file that is used to record the information of the index of data block and the information of its tag generator, including the hash value $H_1(ID_i + \delta)$ and the public key Y_{ID_i} of user u_i. During the auditing process, the TPA gets the hash value $H_1(ID_i + \delta)$ for the challenged data block m_j from the log file. Because the user identity is randomized by $\delta \in Z_q^*$ in Partial-Private-Key-Gen, it is impossible for the TPA to obtain user's real identity. Therefore, the user identity cannot be known by the TPA.

6. Performance and Implementation

In this section, we give the performance analysis and experimental results of our protocol.

6.1. Performance Analysis

We summarize the computational and the communicational cost of our protocol as follows.

Computational cost: For simplicity, we denote by $\mathrm{Exp}_{\mathrm{G2}}$ and $\mathrm{Exp}_{\mathrm{G2}}$ the exponentiations in G_1 and G_2, by $\mathrm{Mult}_{\mathrm{G1}}$ and $\mathrm{Mult}_{\mathrm{G2}}$ the multiplication in G_1 and G_2, by P the pairing computation, and by H the map-to-point hash function, respectively. The original hash function, PRF and PRP operation, addition and multiplication on Z_q^*, and so on are omitted in our evaluation, because the computational cost of them is negligible. Suppose the data is split into n blocks. The main computation of the group manager is generating system parameters and partial private key for each group user. Thus, the main computational cost is $2\mathrm{Exp}_{\mathrm{G1}} + H$. The primary computation of group users is generating tags for data blocks, which is the most expensive operation in our protocol, but fortunately part of it can be done offline. The cost of group users is $(2n + 1)\ \mathrm{Exp}_{\mathrm{G1}} + nH$. The dominated computation of the TPA is generating a challenge and checking the validity of a proof. We suppose all the group users have generated tags and the challenge involves their corresponding tags. Thus, the cost for the TPA is $zP + z\mathrm{Exp}_{\mathrm{G2}} + \mathrm{Exp}_{\mathrm{G1}}$ at most for one challenge. When checking a proof, the cost for the TPA is $2c\mathrm{Exp}_{\mathrm{G1}} + \mathrm{c}P_{\mathrm{G1}} + cH + (c - 1)\ \mathrm{Exp}_{\mathrm{G2}}$. The main computational cost for the CSP is to generate a proof

for a challenge, and the total cost is $(2c-1)$ Mult_{G1} + $c\text{Mult}_{G2}$ + $c\text{Exp}_{G2}$ + $c\text{Exp}_{G1}$ + P. We make a detailed comparison with Li's protocol [30] in Table 1.

Communicational cost: In the challenge phase, the TPA summits $chal = \{Q, T_1, T_2\}$ to the CSP. The CSP responds with a proof to the TPA that is a hash value from H_3. Thus, the communicational cost is $2c \log_2 \lambda + (c+1) \log_2 q + l$.

Table 1. Comparison with Li [30].

	Li [30]	Our CL-PGSDP
Tag Generation cost	$2n\text{Exp}_{G1} + nH$	$2n\text{Exp}_{G1} + nH$
Challenge cost	Negligible cost	$zP + z\text{Exp}_{G2} + \text{Exp}_{G1}$
Proof Generation cost	$c\text{Exp}_{G1} + (c-z)\ \text{Mult}_{G1}$	$(2c-1)$ $\text{Mult}_{G1} + c\text{Mult}_{G2} + c\text{Exp}_{G2} + c\text{Exp}_{G1} + P$
Proof-Check cost	$(z+2)\ P + (c+d)\ \text{Exp}_{G1} + (c+2d)$ $\text{Mult}_{G1} + d\text{Mult}_{G2}$	$2c\text{Exp}_{G1} + cP_{G1} + cH + (c-1)\ \text{Exp}_{G2}$
Working scenario	Group data	Group data
Support data privacy-preserving	No	Yes
Support user identity privacy-preserving	No	Yes

6.2. Experimental Results

To evaluate the efficiency of our proposed protocol, we conduct experiments on Ubuntu 16.04 operation system with Intel i5-6200CPU @ 2.40 GHz and 4 GB memory. We implement the algorithm in C with the Pairing-based Cryptography (PBC) library [37] and the GNU Multiple Precision Arithmetic (GMP) library [38]. We utilize the parameter **a.param,** which provides a symmetric pairing with the fastest speed among all default parameters. The implementation time overhead of the protocol is displayed in following two parts.

In the first part, we carry out an experiment to evaluate the tag generation cost of our protocol. We set the number of the group users at 50, and the size of data ranges from 0.2 MB to 2 MB. The order of G_1 is set at 160-bit, which has the equivalent security level of 1024-bit RSA. Hence, the number of data blocks ranges from 10,000 to 100,000. Figure 3 depicts the comparison on computation cost for tag generation. From Figure 3, we can see that the phase of tag generation of our protocol is separated into two parts: offline and online. As its computational cost is the most expensive one, the group user u_i can preprocess $H_2(\omega_j)^{y_{ID_i}}$, which can be conducted in offline phase. In the online phase of tag generation, group user u_i needs to compute ${psk_{ID_i}}^{m_j}$ for each data block, which is more practical than Li's protocol. The time cost of offline computation of tag generation for 1 MB data file is 150.5 s, while the online time cost is 25.3 s. Without loss of generality, we suppose the blocks are, on average, assigned to group users, which makes the result more acceptable. As we expected that both online and offline time of generating tags increases almost linearly with the increase of the data size. Thus, one shall be able to anticipate the time cost of generating tags for any size of data files.

In the second part, we increase the number of challenged data blocks from 50 to 1000 with an increment of 50 for each test to see the time cost of Challenge, Proof-Gen, and Proof-Check steps. We suppose all the users in the group will get involved in generating the tags of the challenged data blocks. Figure 4 demonstrates the time cost of these three parts, which increase with the increase of the number of challenged data blocks, which is consistent with our previous computational analysis, because when the number of challenged data blocks rises, more random values in Q need to be produced and more $T_2 = \{T_{2j}\}$ need to be computed. The CSP has increasing computation on μ and σ. According to [9], if the CSP has polluted 1% of the data blocks, the TPA can achieve the probability of CSP's misbehavior detection of at least 99% while only needing to make 460 data blocks for a challenge. We can see that it costs the TPA only about 5.75 s to verify a response and the CSP costs about 1.1 s to generate a response when the number of challenged data blocks is 460. We compare the Proof-Check performance between Li's protocol and our protocol in Table 2. We find that our mechanism requires more checking time compared to Li's protocol. Based on our analysis of computation cost from

Table 1, we find that there are more pairing operations (P) during the phase of Proof-Check, which is consistent with our experimental results, because one pairing operation takes more time than other cryptographic operations. However, our mechanism is of comprehensive privacy preservation, while Li's not.

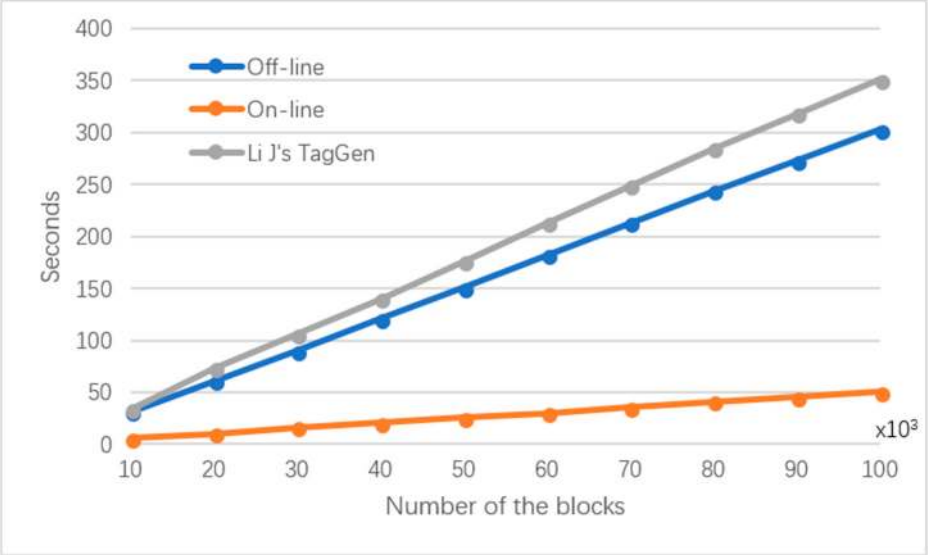

Figure 3. Tag generation time for increased number of data blocks.

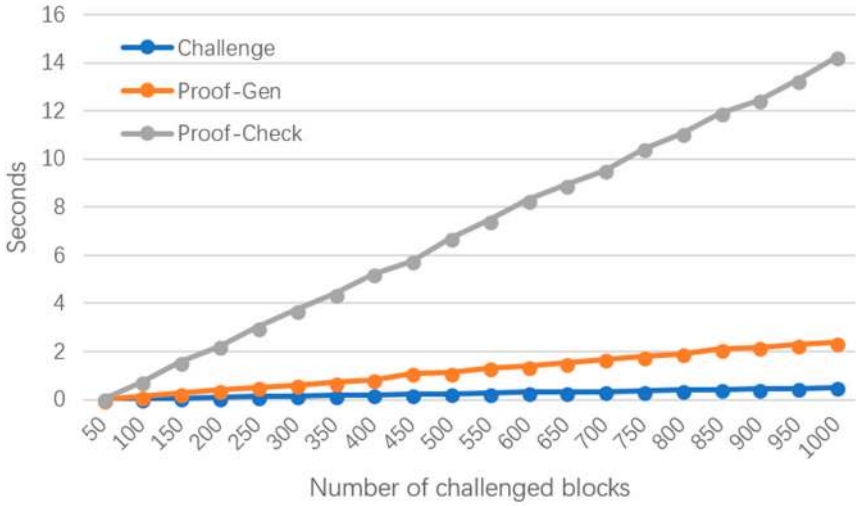

Figure 4. Increasing number of challenges for fixed size of data.

Table 2. Comparison of Proof-Check with Li [30].

	Li [30]	**Our CL-PGSDP**
Number of challenged blocks	460	460
Time cost	2.16 s	5.75 s

7. Conclusions

In this paper, we propose a new PDP protocol for group shared data at untrusted cloud storage, which aims to solve the problems of privacy preservation, including data privacy and the group user identity privacy. By utilizing certificateless cryptography, we eliminate the issues of expensive

certificate management and key escrow. We prove that our protocol is secure, and further illustrate its efficiency through practical experiments. The results show that the proposed protocol is efficient and practical.

Author Contributions: S.J. and H.Y. conceived and designed the experiments, S.J. performed the experiments, S.J. and W.S. analyzed the data, W.S. contributed analysis tools, and S.J. wrote the paper, Z.L. and S.J. revised the manuscript.

Funding: This research was partially supported by Shanghai Innovation Action Plan Project, grant number 16511101200.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mell, P.; Grance, T. *The NIST Definition of Cloud Computing*; Florida Entomological Society: Gaithersburg, MD, USA, 2011.
2. Antonis, M.; Paladi, N.; Gehrmann, C. Security aspects of e-health systems migration to the cloud. In Proceedings of the IEEE 16th International Conference on e-Health Networking, Applications and Services (Healthcom), Natal, Brazil, 15–18 October 2014.
3. Santos, N.; Gummadi, K.P.; Rodrigues, R. Towards Trusted Cloud Computing. In Proceedings of the Conference on Hot Topics in Cloud Computing, San Diego, CA, USA, 14–19 June 2009.
4. Paladi, N.; Gehrmann, C.; Michalas, A. Providing user security guarantees in public infrastructure clouds. *IEEE Trans. Cloud Comput.* **2017**, *5*, 405–419. [CrossRef]
5. Chu, C.; Zhu, W.; Han, J.; Liu, J.K.; Xu, J.; Zhou, J. Security concerns in popular cloud storage services. *IEEE Pervasive Comput.* **2013**, *12*, 50–57. [CrossRef]
6. Miller, R. Amazon Addresses EC2 Power Outages. Available online: http://www.datacenterknowledge.com/archives/2010/05/10/amazon-addresses-ec2-power-outages (accessed on 6 June 2018).
7. Yang, K.; Jia, X. Data storage auditing service in cloud computing: Challenges, methods and opportunities. *World Wide Web* **2012**, *15*, 409–428. [CrossRef]
8. Kaufman, L.M. Data security in the world of cloud computing. *IEEE Secur. Privacy* **2009**, *7*, 61–64. [CrossRef]
9. Ateniese, G.; Burns, R.; Curtmola, R.; Herring, J.; Kissner, L.; Peterson, Z.; Song, D. Provable data possession at untrusted stores. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007; pp. 598–609.
10. Juels, A.; Kaliski, B.S., Jr. PORs: Proofs of retrievability for large files. In Proceedings of the 14th ACM Conference on Computer and Communications Security, Alexandria, VA, USA, 29 October–2 November 2007.
11. Shacham, H.; Waters, B. Compact proofs of retrievability. In Proceedings of the 14th Annual International Conference on the Theory and Application of Cryptology and Information Security, Melbourne, Australia, 7–11 December 2008; pp. 90–107.
12. Boneh, D.; Lynn, B.; Shacham, H. Short signatures from the Weil pairing. In Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security, Gold Coast, Australia, 9–13 December 2001; pp. 514–532.
13. Ateniese, G.; di Pietro, R.; Mancini, L.V.; Tsudik, G. Scalable and efficient provable data possession. In Proceedings of the 4th International Conference on Security and Privacy in Communication Networks, Istanbul, Turkey, 22–25 September 2008; p. 9.
14. Erway, C.; Küpçü, A.; Papamanthou, C.; Tamassia, R. Dynamic provable data possession. In Proceedings of the ACM Transactions on Information and System Security (TISSEC), Chicago, IL, USA, 9–13 November 2009; p. 15.
15. David, C.; Küpçü, A.; Wichs, D. Dynamic proofs of retrievability via oblivious RAM. *J. Cryptol.* **2017**, *30*, 22–57.
16. Wang, Q.; Wang, C.; Li, J.; Ren, K.; Lou, W. Enabling public verifiability and data dynamics for storage security in cloud computing. In Proceedings of the European Symposium on Research in Computer Security, Saint-Malo, France, 21–23 September 2009; pp. 355–370.
17. Liu, C.; Ranjan, R.; Yang, C.; Zhang, X.; Wang, L.; Chen, J. MuR-DPA: Top-down levelled multi-replica merkle hash tree based secure public auditing for dynamic big data storage on cloud. *IEEE Trans. Comput.* **2015**, *64*, 2609–2622. [CrossRef]

18. Yuan, J.; Yu, S. Proofs of retrievability with public verifiability and constant communication cost in cloud. In Proceedings of the International Workshop on Security in Cloud Computing, Hangzhou, China, 8 May 2013; pp. 19–26.
19. Wang, H. Proxy provable data possession in public clouds. *IEEE Trans. Ser. Comput.* **2013**, *6*, 551–559. [CrossRef]
20. Wang, C.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving public auditing for data storage security in cloud computing. In Proceedings of the IEEE INFOCOM, San Diego, CA, USA, 14–19 March 2010; pp. 1–9.
21. Wang, C.; Chow, S.S.M.; Wang, Q.; Ren, K.; Lou, W. Privacy-preserving public auditing for secure cloud storage. *IEEE Trans. Comput.* **2013**, *62*, 362–375. [CrossRef]
22. Yu, Y.; Au, M.H.; Mu, Y.; Tang, S.; Ren, J.; Susilo, W.; Dong, L. Enhanced privacy of a remote data integrity-checking protocol for secure cloud storage. *Int. J. Inf. Secur.* **2015**, *14*, 307–318. [CrossRef]
23. Cooper, D.; Santesson, S.; Farrell, S.; Boeyen, S.; Housley, R.; Polk, W. Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile. Available online: https://tools.ietf.org/html/rfc5280 (accessed on 6 June 2018).
24. Yu, Y.; Au, M.H.; Ateniese, G.; Huang, X.; Susilo, W.; Dai, Y.; Min, G. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 767–778. [CrossRef]
25. Wang, H. Identity-based distributed provable data possession in multicloud storage. *IEEE Trans. Ser. Comput.* **2015**, *8*, 328–340. [CrossRef]
26. Wang, H.; Wu, Q.; Qin, B.; Domingo-Ferrer, J. Identity-based remote data possession checking in public clouds. *IET Inf. Secur.* **2013**, *8*, 114–121. [CrossRef]
27. Zhang, J.; Dong, Q. Efficient ID-based public auditing for the outsourced data in cloud storage. *Inf. Sci.* **2016**, *343*, 1–14. [CrossRef]
28. Boneh, D.; Franklin, M. Identity-Based Encryption from the Weil Pairing. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2001; pp. 213–229.
29. Wang, B.; Li, B.; Li, H.; Li, F. Certificateless public auditing for data integrity in the cloud. In Proceedings of the IEEE Conference on IEEE Communications and Network Security (CNS), National Harbor, MD, USA, 14–16 October 2013; pp. 136–144.
30. Li, J.; Yan, H.; Zhang, Y. Certificateless public integrity checking of group shared data on cloud storage. *IEEE Trans. Serv. Comput.* **2018**. [CrossRef]
31. Al-Riyami, S.S.; Paterson, K.G. Certificateless public key cryptography. In Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, 30 November–4 December 2003; pp. 452–473.
32. Goldwasser, S.; Micali, S.; Rackoff, C. The knowledge complexity of interactive proof systems. *SIAM J. Comput.* **1989**, *18*, 186–208. [CrossRef]
33. Feige, U.; Fiat, A.; Shamir, A. Zero-knowledge proofs of identity. *J. Cryptol.* **1988**, *1*, 77–94. [CrossRef]
34. Chaum, D.; Pedersen, T.P. Wallet databases with observers. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 16–20 August 1992; pp. 89–105.
35. Nyberg, K.; Rueppel, R.A. Message recovery for signature schemes based on the discrete logarithm problem. In Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques, Perugia, Italy, 9–12 May 1997; pp. 182–193.
36. Bao, F.; Deng, R.H.; Zhu, H. Variations of diffie-hellman problem. In Proceedings of the 5th International Conference on Information and Communications Security, Huhehaote, China, 10–13 October 2003; pp. 301–312.
37. Lynn, B. The Pairing-Based Cryptography Library (0.5.14), 2017. Available online: https://crypto.stanford.edu/pbc/ (accessed on 6 June 2018).
38. Free Software Foundation. The GNU Multiple Precision Arithmetic Library. Available online: https://gmplib.org/ (accessed on 6 June 2018).