# An Efficient Identity-Based Provable Data Possession Protocol With Compressed Cloud Storage

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

***Abstract*— Cloud storage is more and more prevalent in practice, and thus how to check its integrity becomes increasingly essential. A classical solution is identity-based (ID-based) provable data possession (PDP), which supports certificateless cloud storage auditing without entire user data. However, existing ID-PDP protocols always require that cloud users outsource data blocks, authenticators and a small-sized file tag to the cloud, and make use of the heavy elliptic curve cryptography over bilinear pairing. These disadvantages would result in vast storage, communication, and computation costs, which is unexpected, especially for resource-limited cloud users. To improve the performance, this paper proposes a novel cryptographic primitive: ID-based PDP with compressed cloud storage. In this model, cloud storage auditing can be achieved by using only encrypted data blocks in a self-verified way, and original data blocks can be reconstructed from the outsourced data. Thus, data owners no longer need to store original data blocks on the cloud. We also use some basic algebraic operations to realize a concrete ID-based PDP protocol with compressed cloud storage, which is quite efficient due to no heavy cryptographic operations involved. The proposed protocol can easily be extended to support the other practical functions by using the primitive replacement technique. The proposed protocol is strictly proven to have the properties of correctness, privacy, unforgeability and detectability. Finally, we give plenty of theoretical analysis and experimental results to validate the efficiency of the proposed protocol.**

Manuscript received August 29, 2021; revised December 25, 2021; accepted March 5, 2022. Date of publication March 11, 2022; date of current version April 5, 2022. The work of Yang Yang was supported by the Fundamental Research Funds for the Central Universities under Grant 2042021kf1030. The work of Yanjiao Chen was supported by the National Natural Science Foundation of China under Grant 61972296. The work of Fei Chen was supported in part by the National Natural Science Foundation of China under Grant 61872243 and in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2020A151501489. The work of Jing Chen was supported in part by the National Natural Science Foundation of China under Grant U1836202, Grant 61772383, Grant 62076187, Grant 61802214, and Grant 62172303; and in part by the National Key Research and Development Program of China under Grant 2021YFB2700200. The associate editor coordinating the review of this manuscript and approving it for publication was Mr. Frederik Armknecht. *(Corresponding author: Yanjiao Chen.)*

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

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

Fei Chen is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518061, China (e-mail: fchen@szu.edu.cn).

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

Digital Object Identifier 10.1109/TIFS.2022.3159152

***Index Terms*— Basic algebraic operations, compressed cloud storage, identity-based integrity auditing, provable data possession.**

## I. Introduction

AS THE amount of data increases sharply, more and more data owners prefer to store their data on the remote cloud for storage burden release, universal data access, etc. Although there are many benefits of cloud storage, it also results in some serious security issues. The cloud cannot guarantee that its storage is always intact due to unintentional mistakes and intentional attacks. As identified in the CSA's white paper [1], the cloud is not responsible for actively notifying its users of data abnormality. What is worse, the cloud might try to cheat the user with falsified data for its economic interests. Consequently, it is extremely important for the users to audit the integrity of cloud storage from time to time.

To address the above problems, a simple way is to download the entire data for integrity verification, which however is far from practice since the amount of cloud storage is extremely large. As a consequence, some well-known cryptographical techniques for integrity verification, such as message digest and secure hash function, cannot work well in this scenario. Fortunately, a new cryptographical primitive, called provable data possession (PDP), was proposed to verify cloud storage integrity in a probabilistic way [2]. In such a primitive, the verifier is able to handle integrity verification by using a few aggregated values of cloud storage instead of the entire data. According to the probabilistic analysis on PDP [2], if 1% of 1,000 outsourced blocks are damaged, the verifier is able to audit only 300 random blocks to disclose the data abnormality with the probability greater than 95%. Thus, PDP has attracted much attention in the secure cloud storage field since it was proposed.

Up to now, many protocols based on PDP were proposed. Among of them, identity-based (ID-based) PDP protocols received the most attention due to the following two advantages: 1) ID-based public auditing is supported, i.e., any verifier having user identity is able to handle integrity verification; 2) no certificate is required for public-key authentication, which can avoid the complex certificate management. However, in existing ID-based PDP protocols, integrity verification is usually implemented using the computationally complex bilinear pairing, which results in a high computation

cost. Furthermore, the user outsources file blocks and their authenticators to the cloud at the same time, which leads to high storage and communication costs. To improve the practicability of ID-based PDP protocol, it is meaningful to realize it by using basic algebraic operations and make outsourced data compressed.

Although some PDP protocols are consisting of only basic algebraic operations, such as [3], [4], they cannot be directly extended to ID-based PDP protocols with compressed cloud storage. The reasons are as follows: 1) they only allow the data owner to initiate private auditing and cannot support public auditing; 2) they require that the user outsources both data blocks and their authenticators for integrity verification. Under this circumstance, we are motivated to design a compressive ID-based PDP with no heavy cryptography. The key of our solution is a new type of data authenticator, which can verify its integrity by itself. Moreover, a random number of these well-designed authenticators can be aggregated into a single value for integrity verification. Thus, the user can only outsource data authenticators to the cloud, which achieves compressed cloud storage. Our solution contains only basic algebraic operations, in which the most complicated operation is modular exponentiation. Remarkably, the resource-limited user only involves several additions and multiplications, which consumes a little workload.

### A. Contribution

This paper further studies how to achieve cloud storage auditing by using ID-based PDP. Our contributions are listed as follows:

- We make the first attempt to design an ID-based PDP protocol based on basic algebraic operations. Compared with existing ID-based PDP protocols with the complexity $O(n^4)$, the proposed protocol only has the complexity $O(n^3)$, which is highly efficient. The proposed protocol builds its security on the well-known computationally infeasible problems, the proposed protocol can protect user privacy against collusive attack, which is not considered in most of the existing ID-based PDP protocols.
- We extend the proposed protocol to support compressed cloud storage. In such a solution, the data owner only stores encrypted data blocks (i.e., data authenticators) and a small-sized file tag on the cloud. Then, storage and communication costs can be hugely reduced since original data blocks are no longer required to be uploaded. As far as we know, our protocol is the first true ID-based PDP with compressed cloud storage. We also present how to make the proposed protocol support other practical functions by using the primitive replacement technique.
- We formalize the system and security models of ID-based PDP with compressed cloud storage. For the proposed protocol, we present detailed proof of its correctness, privacy, unforgeability and detectability. At last, we evaluate its performance from the theoretical analysis and experimental results.

### B. Related Work

The seminal work of probabilistic cloud storage auditing is proposed in [2], based on PDP. Its main advantage is that even though verifiers do not possess user data, they can also efficiently disclose the abnormal data incidents in a probabilistic way.

On the one hand, plenty of PDP protocols were proposed to further improve its performance and enrich its functionality. From the efficiency perspectives, the protocols were committed to realize PDP by using the basic algebraic operations instead of the complex cryptographical primitives. For example, the protocols in [3]–[6] were based on the pseudorandom function, the distributed string equality checking, discrete logarithm problem and error correcting codes, respectively. From the perspectives of functionality, the protocols in [9]–[14] were proposed to support dynamic updates of cloud storage; the protocols with key-exposure resistance were designed in [15]–[17], which achieve dynamic updates of user key; the blockchain-based protocols were constructed in [18]–[20], which realize the properties of impartiality and traceability. One thing to note is that the genuine of users' public keys in the PDP is guaranteed by using the certificates from the public key infrastructure (PKI), which introduces the complex certificate management.

On the other hand, plenty of ID-based PDP protocols emerged, in which users' public keys can be retrieved by their identities and thus the correctness of users' public keys can be guaranteed without the help of the certificates from the PKI. The main advantage of ID-based PDP is that it can effectively eliminate complicated certificate management, which breaks the bottleneck of key distribution. Initially, Wang *et al.* [21] extended the previous PDP-based protocol [10] to support ID-based cloud storage auditing. In this solution, cloud storage is composed of data blocks and authenticators, where data authenticator is generated by first multiplying the hash value of data index and the value of data block mapped to an elliptic curve, and then masking the product value with user's secret key. After the breakthrough [21], ID-based PDP gradually becomes the hot spot in the cloud storage auditing field. Many existing ID-based PDP protocols can be viewed as the variants of [21], such as [22], [33]. These variants further extend the functionality of ID-based PDP from different aspects. Specifically, the protocols in [22]–[24] were able to support user revocation in a cloud storage sharing group, which can prohibit the revoked users from uploading data. The protocols in [25], [26] allowed data owners to delegate their data outsourcing tasks to the third-party proxies. The protocol in [27] achieved the property of zero-knowledge privacy preserving by using the asymmetric group key agreement. The protocol in [28] supported not only dynamical update of cloud storage but also fair arbitration of auditing results. The protocol in [29] realized user key distribution in a fuzzy way, which further lower the burden of key management. The protocol in [30] enabled data owners to sanitize their sensitive information into uniform messy codes before outsourcing to the cloud. The protocols in [31], [32] designed the incentive mechanisms for some participants during cloud storage auditing tasks, where [31] rewarded the unconditionally anonymous data uploaders and [32] introduced the blockchain technique to reward the user who plays the role of the group manager. The protocol in [33] utilized the blockchain to check the auditing results of the TPA, which can resist the misbehavior that TPA

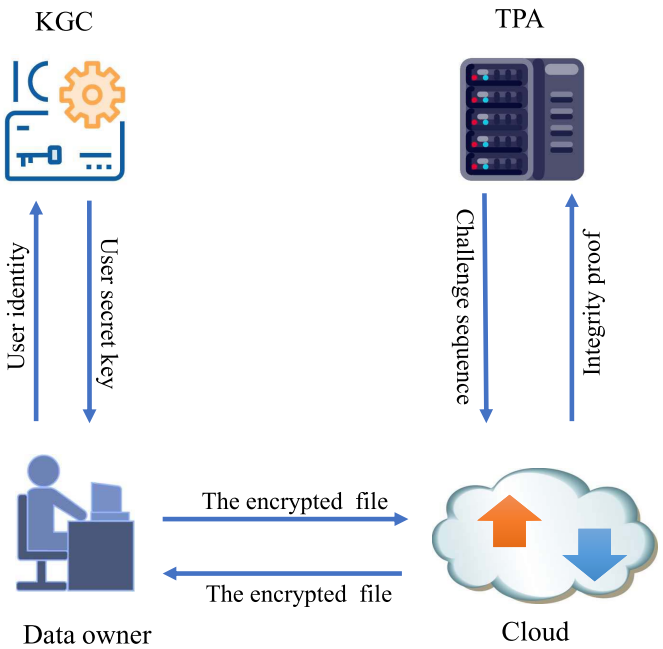


Fig. 1. The system model.

is not getting in touch with the data owner and colluding with the cloud.

Untrivally, all of the above PDP protocols [2]–[32] are relied on both data blocks and their authenticators to achieve cloud storage auditing. A recent compressive PDP protocol in [34] enabled that data owner no longer needed to outsource original data blocks, while keeping cloud storage auditing in effect. However, to support ID-based cloud storage auditing, the data owner in [34] is required to publish a secret value to the third-party auditors, whose correctness needs to guaranteed by the PKI's certificates. This is opposite to the original intention of ID-based PDP. In addition, most of the existing ID-based PDP protocols are built on the complex elliptic curve cryptography (ECC) over bilinear pairing, which results in heavy workloads. To meet the above challenges, this paper first explores a true ID-based PDP protocol with compressed cloud storage, which contains only basic algebra operations.

### C. Organization

In the rest, we introduce some preliminaries of this paper in Section II. An efficient ID-based PDP protocol with compressed cloud storage is presented in Section III. The theoretical analysis is provided in Section IV. Section V evaluates the performance. At last, Section VI presents the conclusion.

## II. Preliminaries

This section first illustrates the system model of ID-based PDP with compressed cloud storage, and then introduce its security model and background knowledge.

### A. System Model

An ID-based PDP protocol with compressed storage contains four entities: key generation center (KGC), data owner, cloud, and third-party auditor (TPA), which are illustrated as follows:

- *KGC*: is to initialize system parameters and then generate user keys.
- *Data owner*: is to store a large amount of data on the cloud, and concern about whether the outsourced data is intact or not.
- *Cloud*: is to provide abundant resources for storing its users' data.
- *TPA*: is to handle integrity verification against cloud storage on behalf of data owners.

As depicted in Fig. 1, the interactions among these four entities are given below. During data outsourcing, the data owner transmits the identity to the KGC for requesting a secret key, and outsources the encrypted file to the cloud for storage. During integrity auditing, the TPA first transmits the challenge sequences to the cloud for requesting the integrity proof and then checks whether the challenged data is intact or not according to the received proof, finally reports the verification results. During data recovery, the data owner downloads the encrypted file from the cloud for decryption.

Next, we summarize the framework of ID-based PDP with compressed cloud storage in Definition 1.

*Definition 1: An ID-based PDP protocol with compressed cloud storage contains seven algorithms*:

- *Setup: Given a security parameter, the KGC produces the master secret key and the public key.*
- *Extract: Given user identity, the KGC distributes the user's secret key to this user. If it is valid, the user accepts it; otherwise, drops it.*
- *Outsource: Given the data file, the data owner outputs the encrypted values of file blocks and the small-sized file tag to the cloud.*
- *Challenge: Given the file tag, the TPA outputs challenge sequences to the cloud.*
- *ProofGen: Given the encrypted file blocks and the challenge sequence, the cloud computes an integrity proof to the TPA.*
- *Verify: Given the file tag, the challenge sequence, and the integrity proof, the TPA outputs the verifying result to the owner of the challenged file.*
- *Recover: Given the encrypted file, the data owner outputs the original data.*

It is intuitive that the above scheme is faced with the following security threats: 1) the third-party entities, including the KGC, the TPA, and the cloud, might be interested in the privacy of data owner, which is driven by economic incentives or personal curiosity; 2) when the audited data is damaged, the cloud might provide the falsified proofs in order to pass the TPA's verification, which is driven by protecting its reputations and interests; 3) when only a small part of user file is corrupted, the TPA's auditing blocks might not contain these broken ones, which is due to the limited length of the auditing sequence.

Based on the above discussion, an ID-based PDP with compressed cloud storage is expected to achieve the following goals:

- Correctness: if the protocol is executed without misbehavior, its correctness verification and data recovery can always be satisfied.
- Privacy: the data owner can prevent the contents of his/her files from the other entities in the protocol.
- Unforgeability: the cloud cannot falsify an integrity proof that is verified successfully.
- Detectability: the TPA can detect the abnormality of cloud storage with a non-negligible probability.
- Efficiency: the costs of storage, communication, and computation are expected to be as small as possible.

### B. Security Model

According to the above design goals, we formalize the security model of ID-based PDP with compressed cloud storage by using the following security definitions.

*Definition 2 (Privacy): An ID-based PDP protocol with compressed cloud storage fulfills the property of privacy if an adversary $\mathcal{A}$ can hardly win the following security game.*

- *Setup*. A challenger $\mathcal{C}$ performs the algorithm Setup and obtains the master secret key and the public key. $\mathcal{C}$ transmits the public key to the adversary $\mathcal{A}$.
- *Query*. $\mathcal{A}$ is allowed to query the following parameters from the challenger $\mathcal{C}$.
  1) *Extract Queries*: $\mathcal{A}$ queries the private key for the identity $ID$. $\mathcal{C}$ performs the algorithm Extract to generate the private key, which is then sent to $\mathcal{A}$.
  2) *Outsource Queries*: $\mathcal{A}$ queries the encrypted blocks for a data file $F$. $\mathcal{C}$ performs the algorithm Outsource to compute the encrypted blocks and sends them to $\mathcal{A}$.
- *Challenge*. $\mathcal{C}$ first generates some random data blocks and then computes their encrypted values through the algorithm Outsource. $\mathcal{C}$ transmits these encrypted blocks to $\mathcal{A}$ for retrieving their original values.
- *Decode*. $\mathcal{A}$ responses the decoded data blocks to the challenge of $\mathcal{C}$. If its decoded data blocks are verified successfully by $\mathcal{C}$, $\mathcal{A}$ is regarded as the winner.

In this security game, $\mathcal{A}$ aims to crack the data blocks from their encrypted values that never have been queried from $\mathcal{C}$. We are required to prove that even though $\mathcal{A}$ makes enough queries, it can still hardly retrieve the unqueried data blocks from their encrypted values.

*Definition 3 (Unforgeablility): An ID-based PDP protocol with compressed cloud storage achieves the property of unforgeability if the probability that the adversary $\mathcal{A}$ wins the security game against the challenger $\mathcal{C}$ is negligible in the security parameter, where the security game is given below:*

- *Setup*. $\mathcal{C}$ performs the algorithm Setup and produces the master secret key and the public key.
- *Query*. $\mathcal{A}$ is allowed to launch different types of queries against the challenger $\mathcal{C}$, which are the same as the queries identified in Definition 2 and thus are omitted here.
- *ProofGen phase*. For a file $F$ that the Outsource query has been made, $\mathcal{A}$ runs the algorithms Challenge and ProofGen to compute a challenge $chal$ and the corresponding proof $\Gamma$ based on the its encrypted blocks from the Outsource queries, and transmits $(chal, \Gamma)$ to $\mathcal{C}$. $\mathcal{C}$ handles the verification of $\Gamma$ by using the algorithm Verify, and returns the result to $\mathcal{A}$.
- *Output phase*. At last, $\mathcal{A}$ randomly selects a file $F$ on the condition that $F$ must not have been queried in the Outsource queries. $\mathcal{A}$ produces and feeds back an integrity proof $\Gamma$. If $\mathcal{A}$ can produce $\Gamma$ that passes the verification of $\mathcal{C}$ with a non-negligible probability $\epsilon$, $\mathcal{A}$ is considered as $\epsilon$-admissible.

In this security game, the goal of $\mathcal{A}$ is to forge a valid proof of the data blocks that never have been queried to, which can pass the integrity verification of $\mathcal{C}$. We intend to prove that even though $\mathcal{A}$ makes enough queries, the forged proof of the unqueried data blocks can pass the verification of $\mathcal{C}$ with a negligible probability.

*Definition 4 (Detectability): An ID-based PDP protocol with compressed cloud storage reaches the property of detectability if only a tiny fraction of the user file on the cloud is damaged, the TPA can still disclose this abnormality with a non-negligible probability.*

### C. Identity-Based Signature

An identity-based signature usually contains four algorithms described below [35].

- *Setup*$(\lambda) \rightarrow MSK$. Given the security parameter $\lambda$, KGC is to output the master secret key $MSK$ and the system public key $PK$.
- *Extract*$(ID, MSK) \rightarrow SK_{ID}$. Given a user identity $ID$ and the master secret key $MSK$, KGC is to output the secret key $SK_{ID}$ for this user.
- *Sign*$(m, ID, SK_{ID}) \rightarrow \tau$. Given a user's message $m$, identity $ID$ and secret key $SK_{ID}$, the user is to output a signature $\tau$ of this message $m$.
- *Verify*$(m, \tau, ID, PK) \rightarrow \nu$. Given a user's message $m$, a signature $\tau$, identity $ID$ and the public key $PK$, verifier is to output $\nu = 0$ if this signature is valid, and otherwise $\nu = 1$.

### D. Mathematical Background

Suppose that a multiplicative cyclic group $G$ has the prime order $p$ and the generator $g$.

*Computational Diffie-Hellman (CDH) Problem*: is to solve $g^{xy}$ with the input of $g, g^x, g^y \in G$, where $x, y \in Z_p^*$ are unknown.

*Divisible Computation Diffie-Hellman (DCDH) Problem*: is to solve $g^{y/x}$ with the input of $g, g^x, g^y \in G$, where $x, y \in Z_p^*$ are unknown.

*Discrete Logarithm (DL) Problem*: is to solve $x \in Z_p^*$ with the input of $g, g^x \in G$.

Notably, the probabilities of solving the CDH, the DCDH, and the DL in a probabilistic polynomial time (PPT) are negligible. At last, the main notations in this paper are summarized in Table I.

TABLE I
NOTATIONS

| Notation | Description |
|---|---|
| $\mathbb{Z}_p$ | A set of $\{0, 1, \ldots, p-1\}$ |
| $x \in \mathbb{Z}_p$ | A random number chosen from $\mathbb{Z}_p$ |
| $x << y$ | $x$ is much smaller than $y$ |
| $x/y$ or $\frac{x}{y}$ | A division of $x$ by $y$ |
| $x \mod y$ | A modulus after division of $x$ by $y$ |
| $floor(\cdot)$ | Round down to the nearest integer |
| $H(\cdot)$ | A secure hash function |
| $SSig(\cdot)$ | An identity-based secure digital signature function |

## III. The Proposed Protocol

### A. Overview

To compress the size of cloud storage, we propose to handle integrity verification by simply using the encrypted data blocks without the help of their original values. For this purpose, we first introduce a lemma below.

*Lemma 1: Given $a, b \in Z_p$ and $y = ax + b$, but $x$ is unknown. If $b < a$, one can have*

$$y = floor(y/a) \cdot a + b. \tag{1}$$

*Proof:*

$$\begin{aligned} floor(y/a) \cdot a + b &= floor((ax + b)/a) \cdot a + b \\ &= floor(x + b/a) \cdot a + b \end{aligned} \tag{2}$$

Due to $b < a$, we can observe $b/a < 1$ and then $floor(x + b/a) = x$. Substituting $floor(x + b/a) = x$ into (2), we can directly obtain

$$floor(y/a) \cdot a + b = x \cdot a + b = y$$

The proof is completed. □

Assume that $x$ is a data block, $y$ is its encrypted value, and $(a, b)$ is the secret key kept by the cloud user. After outsourcing $y$ to the cloud, the user can easily validate its integrity by checking whether (1) holds or not. This implies that we can utilize Lemma 1 to design a private PDP protocol with compressed cloud storage when only a single data block is audited. However, PDP is a probabilistic integrity checking framework for cloud storage. To convince the cloud user, the number of challenged blocks is required to be much greater than 1. To meet this challenge, we give the following theorem derived from Lemma 1.

*Theorem 1: Given $a \in Z_p$ and a hash function $H : \{0, 1\}^* \to Z_p$, one computes $\Gamma = \sum_{j=1}^{n} e_j y_{i_j}$, where $y_{i_j} = ax_{i_j} + H(i_j)$. If $\sum_{j=1}^{n} e_j H(i_j) < a$, one can have*

$$\Gamma = floor(\Gamma/a) \cdot a + \sum_{j=1}^{n} e_j H(i_j). \tag{3}$$

*Proof:*

$$\begin{aligned} &floor(\Gamma/a) \cdot a + \sum_{j=1}^{n} e_j H(i_j) \\ &= floor\Big(\sum_{j=1}^{n} e_j x_{i_j} + \frac{\sum_{j=1}^{n} e_j H(i_j)}{a}\Big) \cdot a + \sum_{j=1}^{n} e_j H(i_j) \end{aligned} \tag{4}$$

If $\sum_{j=1}^{n} e_j H(i_j) < a$, then (4) can be transformed into

$$\begin{aligned} floor(\Gamma/a) \cdot a + \sum_{j=1}^{n} e_j H(i_j) &= \sum_{j=1}^{n} e_j a x_{i_j} + \sum_{j=1}^{n} e_j H(i_j) \\ &= \sum_{j=1}^{n} e_j (a x_{i_j} + H(i_j)) \\ &= \Gamma \end{aligned}$$

The proof is completed. □

Assume that the user computes the encrypted value of the data block $x_i$ as $y_i = ax_i + H(i)$, where $a$ and $H$ are the secret key. The user stores the encrypted data blocks on the cloud. In each integrity auditing, the data owner first transmits a challenge sequence $chal = \{i_1, i_2, \ldots, i_n; e_1, e_2, \ldots, e_n\}$ to the cloud, and then receives a proof $\Gamma = \sum_{j=1}^{n} e_j y_{i_j}$, finally validates the integrity of cloud storage according to (3). In such way, we can achieve a private PDP protocol with compressed cloud storage based on Theorem 1.

However, the data owner and the cloud might dispute the integrity verification results for their own interests. For example, the data owner frames that the valid proof cannot pass the verification for compensation, and the cloud with data corruption claims that its proof can be verified successfully to avoid penalty. To solve this problem, a simple way is to introduce a TPA for auditing cloud storage on behalf of data owners. In such a way, the TPA is required to learn the values of $a$ and $H$ for integrity verification. This brings a serious security challenge: the TPA can freely access, insert, delete and modify user data on the cloud. This implies that the user cannot directly transmits his/her secret key to the TPA for public integrity auditing. To meet this challenge, we introduce a novel theorem as follows.

*Theorem 2: Given a large prime $p$, a random $q$ much smaller than $p$, $a \in Z_p$, $b, c, r \in Z_q$, $H : \{0, 1\}^* \to Z_p$, one computes $\Gamma = \sum_{j=1}^{n} e_j y_{i_j}$, where $y_{i_j} = a(x_{i_j} + bH(i_j)) + cx_{i_j}$. If $\hat{a} = a/r$ and $\sum_{j=1}^{n} e_j c x_{i_j} < \hat{a}$, then*

$$g^{cfloor(\Gamma/\hat{a}) \cdot \hat{a}} = g^{a(\Gamma - floor(\Gamma/\hat{a}) \cdot \hat{a})} \cdot g^{abc \sum_{j=1}^{n} e_j H(i_j)} \mod p, \tag{5}$$

*where $g \in Z_p$ is the generator of a multiplicative cyclic group $G$ with the prime order $p$.*

*Proof:*

$$\begin{aligned} floor(\Gamma/\hat{a}) \cdot \hat{a} = floor\Bigg(&\sum_{j=1}^{n} e_j r(x_{i_j} + bH(i_j)) \\ &+ \Big(\sum_{j=1}^{n} e_j c x_{i_j}\Big)/\hat{a}\Bigg) \cdot \hat{a}. \end{aligned}$$

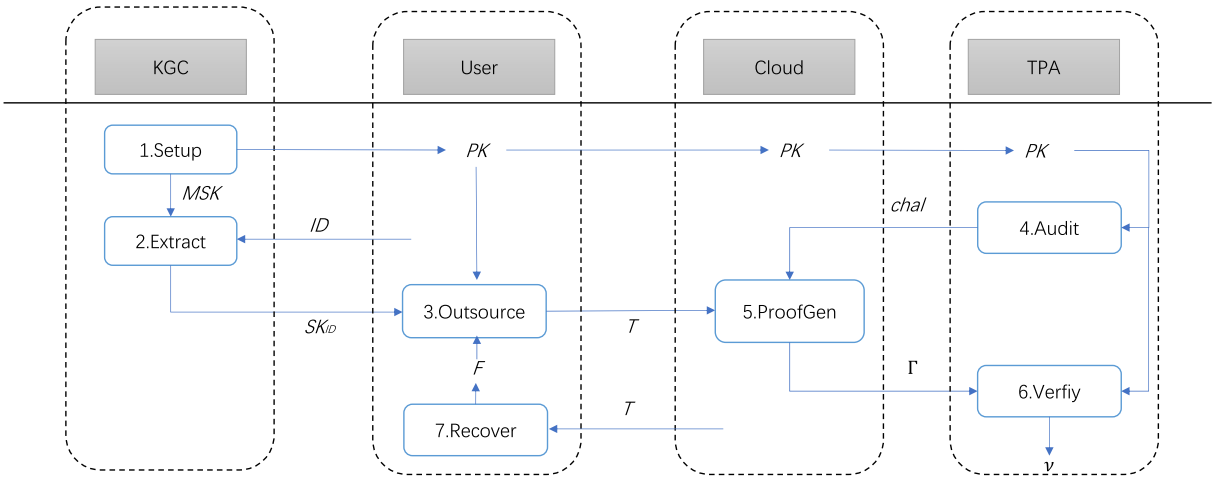


Fig. 2. The flow chart of the overall structure of the proposed IDbased PDP protocol.

Since $\sum_{j=1}^{n} e_j cx_{i_j} < \hat{a}$, and then

$$floor(\Gamma/\hat{a}) \cdot \hat{a} = \sum_{j=1}^{n} e_j a(x_{i_j} + bH(i_j)). \tag{6}$$

From (6), we can have

$$\Gamma - floor(\Gamma/\hat{a}) \cdot \hat{a} = \sum_{j=1}^{n} e_j cx_{i_j}.$$

and then

$$\begin{aligned} g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} &= g^{a\sum_{j=1}^{l} e_j cx_{i_j}} \cdot g^{abc\sum_{j=1}^{n} e_j H(i_j)} \mod p \\ &= g^{a(\Gamma - floor(\Gamma/\hat{a})\cdot\hat{a})} \cdot g^{abc\sum_{j=1}^{n} e_j H(i_j)} \mod p \end{aligned}$$

The proof is completed. □

Assume that the encrypted value of the data block $x_i$ is computed as $y_i = a(x_i + bH(i)) + cx_i$, where $(a, b, c, r)$ is the secret key and $(\hat{a}, H, g, g^a, g^a, g^{abc})$ is the public key. The TPA generates the challenge sequence $chal = \{i_1, i_2, \ldots, i_n; e_1, e_2, \ldots, e_n\}$. While obtaining the challenge sequence, the cloud computes the integrity proof as $\Gamma = \sum_{j=1}^{n} e_j y_{i_j}$. According to the received proof $\Gamma$, the TPA verifies the integrity of cloud storage through examining whether (5) holds or not. Thus, we can realize a public PDP protocol with compressed cloud storage based on Theorem 2. To avoid complex certificate management, we further introduce an ID-based signature algorithm to transform the above public PDP protocol to the ID-based PDP protocol. Please refer to the next subsection for the details.

### B. Protocol Construction

This subsection summarizes the detailed description of the proposed ID-based PDP protocol with compressed cloud storage, which is shown in Fig. 2. To make our description concise, we will omit some 'mod $p$' in our modular exponentiation.

*1) Setup*$(\lambda) \rightarrow (MSK, PK)$*:* The KGC initializes the master secret key $MSK$ and the public key $PK$ of the whole system.

- According to the security parameter $\lambda$, the KGC determines a large prime $p$, a random $q$ much smaller than $p$, two random elements $g, \sigma \in Z_p$ and a hash function $H : \{0, 1\}^* \rightarrow Z_p$.
- The KGC sets $MSK = \sigma$ and $PK = \{p, q, g, g^\sigma, H\}$.

*2) Extract*$(ID) \rightarrow SK_{ID}$*:* The KGC produces the secret key $SK_{ID}$ of a user with the identity $ID$, and the user checks its validity.

- According to the received $ID$, the KGC first randomly picks $\varsigma \in Z_p$ and then calculates $a' = \varsigma + \sigma H(ID) \mod (p-1)$.
- The KGC sets $SK_{ID} = a'$, which is fed back to the corresponding user together with $g^\varsigma$.
- The user checks the correctness of $SK_{ID}$ by judging whether

$$g^{a'} = g^\varsigma \cdot g^{\sigma H(ID)} \mod p. \tag{7}$$

If yes, the data owner accepts it; otherwise, asks for it again.

*3) Outsource*$(F; SK_{ID}, PK) \rightarrow (T, \tau)$*:* The user generates the encrypted values of all blocks in the data file $F$ and the file tag.

- The user determines four random elements $a'' \in Z_p$ and $b, c, r, l \in Z_q$. To resist the escrow attack [23], the data owner updates the secret key as $a = a' + a''$. The data owner then computes $\hat{a} = a/r$. Note that $ql \ll \hat{a}$ is required be satisfied.
- The data owner partitions the data file $F$ into $\{x_1, x_2, \ldots, x_m\}$ and encrypts the data block $x_i$ into $y_i = a(x_i + bH(name||i)) + cx_i$, where $x_i \in Z_l$, $name$ is the identifier of $F$ and $1 \le i \le m$.
- The user generates the set of the encrypted blocks as $T = \{y_1, y_2, \ldots, y_m\}$ and the file tag as $\tau = name\|l\|m\|\hat{a}\|g^a\|g^c\|g^{abc}\|spk\|SSig(name\|l\|m\|\hat{a}\|g^a\|g^b\|g^{abc}, ssk)$, where $SSig$ is an identity-based secure digital signature function due to no PKI involved [36]–[38], $spk$ and $ssk$ are its public and secret keys. Finally, the data owner outsources $(T, \tau)$ to the cloud and privately keeps $(a, b, c, r)$ at local. Note that file tag $\tau$ is only composed of a few values for data auditing, whose size is much smaller than data blocks and their authenticators.

*4) Challenge$(\cdot)\rightarrow chal$:* The TPA produces a challenge sequence $chal$.

- The TPA first retrieves the file tag $\tau$ from the cloud and then verifies its validity using its signature. If invalid, the TPA terminates the audit and reports this failure; otherwise, the TPA proceeds to parse the elements $l$, $m$, $\hat{a}$, $g^a$ and $g^{ac}$ from $\tau$.
- The TPA determines the indices of the challenged blocks $\{i_1, i_2, \ldots, i_n\}$ from $[1, m]$ and the random challenge elements $\{e_1, e_2, \ldots, e_n\}$. According to Theorem 2, the selection of $\{e_1, e_2, \ldots, e_n\}$ is required to make $\sum_{j=1}^{n} e_j ql < \hat{a}$ hold.
- The TPA generates the challenge sequence as $chal = \{i_1, i_2, \ldots, i_n; e_{i_1}, e_{i_2}, \ldots, e_{i_n}\}$, which is then sent to the cloud.

*5) ProofGen$(T) \rightarrow \Gamma$:* In this algorithm, the cloud generates an integrity proof $\Gamma = \sum_{j=1}^{n} e_{i_j} y_{i_j}$ as the response to the TPA's challenge.

*6) Verify$(chal, \Gamma; \tau) \rightarrow \nu$:* In this algorithm, the TPA handles integrity verification by checking whether

$$g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} = g^{a(\Gamma - floor(\Gamma/\hat{a})\cdot\hat{a})} \cdot g^{abc\sum_{j=1}^{n} e_j H(i_j)} \mod p.$$

If yes, the TPA outputs $\nu = 0$, which implies that cloud storage is still intact; otherwise, $\nu = 1$, meaning that cloud storage is corrupted.

*7) Recover$(y_i) \rightarrow x_i$:* The user reconstructs the original data block $x_i$ from its encrypted value $y_i$ by computing $x_i = (y_i - floor(y_i/\hat{a}) \cdot \hat{a})/c$, where $1 \le i \le m$.

### C. Functional Extension

This subsection discusses the functional extension of the proposed protocol.

Up to now, many ID-based PDP protocols with functional extension were proposed in [22]–[32], which are designed based on the cryptographical primitive of the seminal work [21], i.e., ID-based PDP over bilinear pairing. This paper explores a new primitive: ID-based PDP with compressed cloud storage. By replacing the primitive in the existing ID-based protocols, we can easily extend the proposed protocol to support massive functions. Next, we take three examples to illustrate how our solution works.

1) To support user revocation, the protocol in [24] introduces a group manager for managing the group key, which is taken as the partial key of each group user. Once revoking a user, the group manager distributes a new group key to all non-revoked users for updating their secret keys. In such a way, the revoked users cannot upload the data to cloud anymore since they are unable to compute the valid data authenticators by using the outdated secrete keys. In the same way as [24], we can make the proposed protocol support user revocation. For this purpose, we need to slightly modify the algorithms Extract and Outsource, and then add a new algorithm Revoke. The details are as follows.

■ Extract$(ID) \rightarrow SK_{ID}$: The KGC produces the secret key $SK_{ID}$ of a group user with identity $ID$, and then the group manager updates $SK_{ID}$ with a group key.

- According to the received $ID$, the KGC first randomly picks $\varsigma_1, \varsigma_2 \in Z_p$, and calculates $a_1 = \varsigma_1 + \sigma H(ID) \mod (p-1)$. The the KGC transmits $\varsigma_2$ to the group manager for computing the partial key.
- The KGC sets $SK_{ID}$ equal to $a_1$, which is transmitted to the group user.
- The group user checks the correctness of $SK_{ID}$ by judging whether $g^{a_1} = g^{\varsigma_1} \cdot g^{\sigma H(ID)} \mod p$ holds or not. If no, the group user requests it again; otherwise, goes to the next step.
- The group manager sets the number of user revocations as $RN = 0$, which is then transmitted to the cloud and the group users.
- The group manager selects a random element $\delta \in Z_p$, and then computes $a_2 = \varsigma_2 + \delta H(ID, a_1, RN) \mod (p-1)$.
- The group manager sets the group key as $SK_G = a_2$, which is then sent to the group users together with $g^{\delta}$.
- The group user checks the correctness of $SK_G$ by judging whether $g^{a_2} = g^{\varsigma_2} \cdot g^{\delta H(ID, a_1, RN)} \mod p$. If yes, the group user accepts it and updates $SK_{ID}$ as $a' = a_1 + a_2$; otherwise, asks for it again.

■ Outsource$(F; SK_{ID}, PK) \rightarrow (T, \tau)$: In this algorithm, there are only two modifications:

- The group user embeds the number of user revocations $RN$ into the file identifier, which is changed from $name$ into $name||RN$.
- The cloud verifies whether $RN$ in the uploaded data is the newest. If yes, the cloud stores the received data; otherwise, just drops it.

■ Revoke$(F; SK_{ID}, PK) \rightarrow (T, \tau)$: In this algorithm, the group manager regenerates a group secret key for all non-revoked users.

- The group manager updates the number of user revocations as $RN = RN + 1$.
- The group manager and user re-execute the last three steps in the algorithm Extract.

2) To support data dynamics, the protocol in [34] introduces an index array, denoted as $ind\_arr$, to manage the relationship between block indices and encryption indices, where block index indicates the true location of the block and encryption index refers to the index used to generate the encrypted block value. Note that the index of $ind\_arr$ is exactly the block index and its value represents the encryption index. That is to say, the encryption index $i'$ of the $i$-th block can be represented as $i' = ind\_arr[i]$. Initially, block indices and encryption indices are the same, which however become inconsistent after data dynamics, including block insertion, deletion and modification. Thus, we need to slightly modify the algorithm Verify as follows.

■ Verify$(chal, \Gamma; \tau) \rightarrow \nu$: In this algorithm, the TPA carries out integrity verification by judging whether

$$g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} = g^{a(\Gamma - floor(\Gamma/\hat{a})\cdot\hat{a})} \cdot g^{abc\sum_{j=1}^{n} e_j H(i'_j)} \mod p.$$

where $i'_j = ind\_arr[i_j]$. If yes, the TPA outputs $\nu = 0$; otherwise, $\nu = 1$.

3) To support batch auditing, we can easily achieve this due to our aggregated computation on integrity

proof. Suppose that user data are distributedly stored on $K$ clouds, and the $k$-th cloud stores the encrypted data file $T_k = \{y_{k,1}, y_{k,2}, \ldots, y_{k,m}\}$ and the file tag $\tau_k = name_k \| l_k \| m \| \hat{a}_k \| g^{a_k} \| g^{c_k} \| g^{a_k b_k c_k} \| spk \| SSig(name_k \| l_k \| m \| \hat{a}_k \| g^{a_k} \| g^{c_k} \| g^{a_k b_k c_k} \| spk, ssk)$, where $1 \leq k \leq K$. In order to audit these $K$ user files simultaneously (i.e., batch auditing), we slightly modify the algorithms Audit, Prove and Verify as follows.

■ $\mathsf{Audit}(\cdot) \rightarrow chal$: In this algorithm, there is only one modification: the TPA respectively transmits the challenge sequence $chal_k = \{i_{k,1}, i_{k,2}, \ldots, i_{k,n}; e_{k,1}, e_{k,2}, \ldots, e_{k,n}\}$ to $k$-th clouds, where $1 \leq k \leq K$.

■ $\mathsf{ProofGen}(T_k) \rightarrow \Gamma_k$: In this algorithm, the $k$-th cloud generates an integrity proof $\Gamma_k = \sum_{j=1}^{n} e_{k,j} y_{k,i_{k,j}}$ as the response to the TPA's challenge.

■ $\mathsf{Verify}(chal_k, \Gamma_k; \tau_k) \rightarrow \nu$: In this algorithm, the TPA executes integrity verification by judging whether

$$\begin{aligned} & g^{\sum_{k=1}^{K} c_k floor(\Gamma_k/\hat{a}_k)\cdot \hat{a}_k} \\ & \quad = g^{\sum_{k=1}^{K} a_k (\Gamma_k - floor(\Gamma_k/\hat{a}_k)\cdot \hat{a}_k)} \\ & \qquad \cdot g^{\sum_{k=1}^{K} a_k b_k c_k \sum_{j=1}^{n} e_{k,j} H(name_k \| i_{k,j})} \mod p. \end{aligned} \tag{8}$$

If yes, the TPA outputs $\nu = 0$; otherwise, $\nu = 1$. Note that (8) is actually a linear combination of (5) and thus can be easily derived from (5).

### D. Further Discussion

Among of these existing works, the closest one to our work is the solution of proof of retrievability (PoR) in [8]. Although both of these two schemes only involve basic algebraic operations, making them look similar, they have two essential differences: 1) our solution supports identity-based public cloud storage auditing (i.e., anyone with the identity of a user can act as an auditor for this user), but PoR only supports private cloud storage auditing (i.e., only the data owner can behave as an auditor). This is because that user privacy in PoR cannot be guaranteed in case of public auditing; 2) the encrypted data blocks (i.e., data authenticators) can achieve the property of self-verification in our solution, which is not held in PoR. This is due to the fact that the original data blocks in PoR cannot be reconstructed from their authenticators without the help of their indices. In such way, our solution no longer requires original data blocks stored on the cloud, which also implies that the cloud only needs to provide the aggregated value of encrypted data blocks for integrity verification. However, in PoR, data owner is required to store both original data blocks and their authenticators on the cloud, and the cloud needs to provide both of their aggregated values for integrity verification.

## IV. Theoretical Analysis

This section analyzes the correctness and security of the proposed protocol.

### A. Correctness

*Theorem 3: If the KGC, the user, the cloud, and the TPA perform honestly, the following conditions are satisfied: 1) the user always accepts the secret key from the KGC; 2) the TPA always passes the integrity verification in the case that cloud storage is intact; 3) the user always successfully reconstructs the original data from cloud storage.*

*Proof:* 1) The correctness of user secret key is guaranteed by (7). According to Fermat's little theorem [39], it holds that $g^{p-1} = 1 \mod p$. We can then observe that

$$\begin{aligned} g^{a'} &= g^{(\varsigma + \sigma H(ID)) \mod (p-1)} \mod p \\ &= g^{\varsigma} \cdot g^{\sigma H(ID)} \mod p \end{aligned}$$

2) The correctness of integrity verification is determined by (5). According to Theorem 2, (5) is always true if cloud storage is intact.

3) The correctness of users' data reconstruction is derived as follows. Since $x_i \in Z_l, c \in Z_q$ and $ql < \hat{a}$, we can have $floor(cx_i/\hat{a}) = 0$. Then,

$$\begin{aligned} & (y_i - floor(y_i/\hat{a}) \cdot \hat{a})/c \\ & = (y_i - floor\Big(r(x_i + bH(name||i)) + \frac{cx_i}{\hat{a}}\Big) \cdot \hat{a})/c \\ & = (a(x_i + bH(name||i)) + cx_i - a(x_i + bH(name||i))/c \\ & = x_i \end{aligned} \tag{9}$$

From (9), we can conclude that a user can truly reconstruct the data from the cloud if the downloaded data is verified to be intact. □

### B. Privacy

*Theorem 4: Except for the data owner, even though all other entities (including the KGC, the cloud, and the TPA) are collusive, they still cannot obtain the original user blocks.*

*Proof:* We prove the privacy of our protocol based on the security game identified in Definition 2. Suppose $\mathcal{A}$ behaves as the KGC, the cloud, and the TPA, who aims to crack the information of the data owner. According to Definition 2, $\mathcal{A}$ is able to query the values of user's partial key and encrypted blocks. Since user blocks are involved in encrypted blocks and integrity proofs, our proof are mainly from these two perspectives.

1) From a set $T$ of encrypted blocks, $\mathcal{A}$ can have $y_i = a(x_i + bH(name||i)) + cx_i$, where $1 \leq i \leq m$. Then, $\mathcal{A}$ can have the following observations:

$$\begin{cases} floor(y_i/\hat{a}) \cdot \hat{a} = a(x_i + bH(name||i)) \\ y_i - floor(y_i/\hat{a}) \cdot \hat{a} = cx_i \end{cases} \tag{10}$$

It can be observed that $\mathcal{A}$ can utilize (10) to solve the value of $x_i$ if $a$, $b$, or $c$ is given. However, $a$, $b$ and $c$ are all hard to solve due to the following reasons: on one hand, $a$ can hardly be decoded from $\hat{a} = a/r$ and $a = a' + a''$ without the values of $r$ and $a''$, where $\hat{a}$ and $a'$ can be learned from $\tau$ and $SK_{ID}$, respectively; on the other hand, $a, b, c$ can hardly be decoded from $(g, g^a, g^c, g^{abc})$ due to the hard DL problem. Thereafter, $\mathcal{A}$ cannot reconstruct the original user data from the encrypted blocks.

2) From a set $T$ of encrypted blocks, $\mathcal{A}$ can produce integrity proofs with the input of challenge sequences. We start from a simple case that only a single encrypted block $y_i$ is

challenged. Without loss of generality, the challenge element is supposed to be $e_i = 1$. In such way, the integrity proof is computed as $\Gamma = y_i$. According to (5), the integrity verifying equation becomes

$$g^{cfloor(y_i/\hat{a})\cdot\hat{a}} = g^{a(y_i - floor(y_i/\hat{a})\cdot\hat{a})} \cdot g^{abcH(name||i)} \mod p. \tag{11}$$

Substituting (10) into (11),

$$g^{a(x_i + bH(name||i))} = g^{acx_i} \cdot g^{abcH(name||i)} \mod p. \tag{12}$$

Combining (11) and (12), $\mathcal{A}$ can have

$$\begin{cases} g^{ax_i} = g^{a(y_i - floor(y_i/\hat{a})\cdot\hat{a})} \cdot (g^{abc}/g^{ab})^{H(name||i)} \mod p \\ g^{ax_i} = g^{floor(y_i/\hat{a})\cdot\hat{a}}/g^{abH(name||i)} \mod p \end{cases}$$

However, $\mathcal{A}$ can hardly solve $g^{x_i}$ from $(g, g^a, g^{ax_i})$ due to the hardness of the DCDH problem and thus $x_i$ is also unsolvable. This also implies that $\mathcal{A}$ can hardly solve the original user blocks from the integrity proof when a single encrypted block is challenged. For the general case of auditing multiple encrypted blocks, we can have the same observation, which is because that its integrity verifying equation can be viewed as the combination of (11). After that, $\mathcal{A}$ cannot recover the original user data from the integrity proofs. □

### C. Unforgeability

*Theorem 5: If the challenged user data is damaged, it is almost impossible for the cloud to pass the integrity verification of the TPA.*

*Proof:* We prove the unforgeability of our protocol based on a list of security games.

1) We first introduce a new security game $SG_1$, identical to the security game identified in Definition 3.

2) We then introduce another security game $SG_2$. This game is similar to $SG_1$, and their difference is that $\mathcal{A}$ makes enough different types of queries. $\mathcal{C}$ observes the instances of integrity verification and feeds back the verification results to $\mathcal{A}$. If $\mathcal{A}$ can pass the verification of $\mathcal{C}$ by outputting a new integrity proof based on the blocks that have not been queried in the Outsource queries, $\mathcal{C}$ aborts and reports failure.

*Analysis:* We show that if $\mathcal{A}$ has an advantage $\epsilon$ to forge a valid integrity proof in $SG_2$, then $\mathcal{C}$ can solve a CDH problem with the advantage at least $\frac{\epsilon}{2^{\lambda+l}}$, where $l$ is the number of challenge elements for generating the proof. The interactions between $\mathcal{C}$ and $\mathcal{A}$ are simulated as follows.

Setup: $\mathcal{C}$ setups the public key through the algorithm Setup in the proposed protocol, which is then transmitted $\mathcal{A}$.

Extract Queries: $\mathcal{A}$ adaptively requests the secret key $SK_{ID}$ for any user identity $ID$. A list of Extract queries and responses, denoted as $\mathcal{E}^{list}$, is maintained by $\mathcal{C}$. If the identity has been queried, $\mathcal{C}$ returns the recorded $SK_{ID}$ in the list; otherwise, $\mathcal{C}$ firstly sets $SK_{ID}$ equal to a random $a' \in Z_p$ and then returns it to $\mathcal{A}$, finally appends $(ID, SK_{ID})$ to $\mathcal{E}^{list}$.

Hash Queries: $\mathcal{A}$ adaptively picks any $(name, i)$ to query the hash oracle $H$. A list of $H$ queries and responses, denoted as $\mathcal{H}^{list}$, is maintained by $\mathcal{C}$. Once receiving a new query, if the $i$-th query has already been recorded in $\mathcal{H}^{list}$, $\mathcal{C}$ just simply feeds back the recorded value; otherwise, $\mathcal{C}$ randomly chooses $w_i \in Z_p$, and then sets $H(name||i) = w_i$ as the response to the $i$-th query and appends $(name, i, w_i)$ to $\mathcal{H}^{list}$.

Outsource Queries: $\mathcal{A}$ adaptively picks $(ID, name, i, x_i)$ to request the encrypted block $y_i$. It is assumed that and $ID$ and $(name, i)$ have been queried in the queries of Extract and Hash; if not, $\mathcal{A}$ does queries at first. $\mathcal{C}$ also introduces a list $\mathcal{U}^{list}$, which is composed of $(ID, name, \hat{a}, g^a, g^c, g^{abc})$. If there is no tuple on $\mathcal{U}^{list}$ including $(ID, name)$, $\mathcal{C}$ randomly selects $a'' \in Z_p$, $b, c, r \in Z_q$, and calculates $a = a' + a''$, $\hat{a} = a/r$, $g^a$, $g^c$, $g^{abc}$ in order to ensure that such a tuple exists in $\mathcal{U}^{list}$. According to the received query on $(ID, name, i, x_i)$, $\mathcal{C}$ computes $y_i = a(x_i + bw_i) + cx_i$, which is then fed back to $\mathcal{A}$.

ProofGen: $\mathcal{A}$ adaptively requests the verification result of the proof $\Gamma$ by selecting any file $F$ under the identity $ID$. Once receiving the proof $\Gamma$, $\mathcal{C}$ executes the algorithm Verify. If $\Gamma$ is valid, $\mathcal{C}$ outputs $\nu = 0$ and otherwise $\nu = 1$.

Output: $\mathcal{A}$ outputs a forged proof $\Gamma$ on $(ID, name, chal, F)$, which never has been queried for $F$ in the Outsource queries. If $\Gamma$ is valid, $\mathcal{C}$ reports failure and terminates the game. In such way, we can build a simulator which takes $\mathcal{A}$ as a subroutine in order to solve a given instance of the CDH problem during a probabilistic polynomial time. Assuming that the public $(ID, name, \hat{a}, g^a, g^c, g^{abc})$ is given to the simulator. Note that if there is no tuple on $\mathcal{U}^{list}$ including $(ID, name)$, then the simulator issues a query itself to ensure that such a tuple exists. The goal of the simulator is to compute $h^c$ from a triple $(g, g^c, h)$. Let $\Sigma = \sum_{j=1}^{l} e_j x_{i_j}$, where $e_j \in chal$ and $x_{i_j} \in F$. On one hand, $\Gamma$ is assumed to be verified successfully, and thus

$$g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} = g^{ac\Sigma} \cdot g^{abc\sum_{j=1}^{l} e_j w_{i_j}} \mod p. \tag{13}$$

On the other hand, by using the oracle-replay technique, $\mathcal{A}$ can replay the Hash queries and forge another valid proof $\Gamma^*$. In what follows,

$$g^{cfloor(\Gamma^*/\hat{a})\cdot\hat{a}} = g^{ac\Sigma} \cdot g^{abc\sum_{j=1}^{l} e_j w^*_{i_j}} \mod p. \tag{14}$$

Dividing (14) by (13), the simulator can have

$$g^{c\Delta F} = g^{abc\Delta W} \mod p,$$

where $\Delta F = (floor(\Gamma^*/\hat{a}) - floor(\Gamma/\hat{a})) \cdot \hat{a}$ and $\Delta W = \sum_{j=1}^{l} e_j(w^*_{i_j} - w_{i_j})$. Note that $\Delta F \neq 0$ under the condition of $\Delta W \neq 0$. By assuming $h = g^{ab}$, the simulator can obtain

$$h^c = g^{c\Delta F/\Delta W},$$

which is the solution of a CDH instance, unless $\Delta W$ is equal to 0. As $w_{i_j} \in Z_p$, the probability of $\Delta W = 0$ is greater than $1/2^{\lambda+l}$. Thus, we can have

$$Adv_{\mathcal{A}}[Game2] > \frac{Adv_{\mathcal{A}}[Game1]}{2^{\lambda+l}}, \tag{15}$$

where $Adv_{\mathcal{A}}$ represents the advantage of the adversary.

3) Finally, we introduce a security game $SG_3$, which is similar to $SG_2$. The only difference is that $\mathcal{C}$ records all its query responses to $\mathcal{A}$ in $SG_3$. If the proof $\Gamma^*$ provided by $\mathcal{A}$ is accepted by $\mathcal{C}$, but it is not equal to the integrity proof $\Gamma$ generated by using the locally recorded data, $\mathcal{C}$ reports failure and terminates the game.

TABLE II
PERFORMANCE COMPARISON OF DIFFERENT PROTOCOLS

| Protocol | Storage | Communication | Computation | | |
|---|---|---|---|---|---|
| | | | Data owner | Cloud | TPA |
| Wang et al. [21] | $m(\|p\|+\|q\|)$ | $2m\|p\|+n\|q\|$ | $m\|p\|^2\|q\|$ | $n\|p\|^2\|q\|$ | $\|p\|^4$ |
| The proposed protocol | $m\|p\|$ | $m\|p\|+n\|q\|$ | $m\|p\|\|q\|$ | $n\|p\|\|q\|$ | $\|p\|^3$ |

*Analysis*: If $\mathcal{A}$ can forge a valid proof in $SG_3$ with an advantage $\epsilon$, we can build a simulator to solve a given instance of the DL problem with the advantage at least $\frac{\epsilon}{2^\lambda}$. Specifically, our simulator takes $\mathcal{A}$ as a subroutine and intends to compute a value $\alpha$ satisfying $h' = h^\alpha$. Let $\Sigma = \sum_{j=1}^{l} e_j x_{i_j}$, where $e_j \in chal$ and $x_{i_j} \in F$. On one hand, the simulator can generate an integrity proof $\Gamma$ using the data recorded by itself, which satisfies that

$$g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} = g^{ac\Sigma} \cdot g^{abc\sum_{j=1}^{l} e_j w_{i_j}} \mod p. \quad (16)$$

On the other hand, we assume that $\mathcal{A}$ can forge an integrity proof $\Gamma^*$ verified successfully, i.e.,

$$g^{cfloor(\Gamma^*/\hat{a})\cdot\hat{a}} = g^{ac\Sigma} \cdot g^{abc\sum_{j=1}^{l} e_j w_{i_j}} \mod p. \quad (17)$$

Dividing (17) by (16), we can directly have

$$g^{cfloor(\Gamma/\hat{a})\cdot\hat{a}} = g^{cfloor(\Gamma^*/\hat{a})\cdot\hat{a}} \mod p,$$

and can further imply that

$$1 = g^{\Delta F} \mod p, \quad (18)$$

where $\Delta F = (floor(\Gamma^*/\hat{a}) - floor(\Gamma/\hat{a}))\cdot\hat{a}$. According to the assumption $\Gamma \neq \Gamma^*$ of this game, one can have $\Delta F \neq 0$. By assuming $g = h'^{\alpha}h^{\beta}$, and then (18) can be transformed into

$$h' = h^{-\frac{\beta\Delta F}{\alpha\Delta F}} = h^{-\frac{\beta}{\alpha}} \mod p$$

which is the solution of a DL instance, unless $\alpha$ is equal to 0. Due to $\alpha \in Z_p$, the probability of $\alpha = 0$ is greater than $1/2^\lambda$. Thus, we can have

$$Adv_{\mathcal{A}}[Game3] > \frac{Adv_{\mathcal{A}}[Game2]}{2^\lambda},$$

Based on the above discussion, we can observe that the cloud can hardly pass the integrity verification of the TPA by using the forged data. □

### D. Detectability

*Theorem 6: If c of m blocks are corrupted and n blocks are challenged, the probability $P_c$ of disclosing this abnormal incident is $1 - \prod_{k=0}^{n-1}\frac{(m-c-k)!}{m-k}$, where ()! is the factorial function.*

*Proof:* Denote $n_c$ as the number of the corrupted blocks that are audited. Then, we can have

$$\begin{aligned} P_c &= 1 - P\{n_c = 0\} \\ &= 1 - \frac{m-c}{m}\cdot\frac{m-c-1}{m-1}\cdot\cdots\cdot\frac{m-c-(n-1)}{m-(n-1)} \\ &= 1 - \prod_{k=0}^{n-1}\frac{(m-c-k)!}{m-k}. \end{aligned}$$

It can be observed that $P_c$ keeps approaching to 1 as $n$ and $c$ increase. When $n$ or $c$ is equal to its largest value $m$ (i.e., all $m$ data blocks are audited or corrupted), one can easily have $P_c = 1$. □

## V. PERFORMANCE EVALUATION

This section analyzes the performance of the proposed protocol from the perspectives of the storage, communication, and computation costs. Also, we give a comparison between the proposed protocol and the ID-based PDP protocol in [21] (the extended version of the PDP protocol in [10]). The reason why we choose [21] as the comparison is that almost all ID-based PDP protocols are its variants, such as [22]–[32]. What is more, we build a prototype system of the proposed protocol, and provide massive experimental results to validate the practicability of our protocol.

### A. Theoretical Results

For convenience, we concentrate on the highest degree of the cost, which plays a decisive part in the whole cost. Let $|p|$ and $|q|$ represent the lengths of an element in $Z_p$ and $Z_q$, respectively.

*1) Storage Cost:* It depends mostly on cloud storage. In our protocol, cloud storage is mainly determined by the encrypted data blocks, whose storage cost is about $m|p|$. However, cloud storage is mainly caused by user blocks, and their authenticators in the state-of-the-art [21], which incurs about $m(|p|+|q|)$ storage cost.

*2) Communication Cost:* It relies largely on the interaction between the user, the cloud, and the TPA. Thus, the proposed protocol's communication cost is about $m|p|+n|q|$ for each data outsourcing and auditing. In the previous [21], the data owner is required to outsource both data blocks and their authenticators to the cloud, which needs about $2m|p|+n|q|$ communication cost for each data outsourcing and auditing.

*3) Computation Cost:* It is mainly determined by the operations of block encryption, proof generation, and integrity verification, and thus we omit the computation cost of the KGC here, which is negligible. For the data owner, he/she generates the encrypted value of each data block in Outsource, which requires about $m|p||q|$ computation cost. For the cloud, it generates an integrity proof in ProofGen, which leads to about $n|p||q|$ computation cost. The TPA handles integrity verification in Audit and Verify, which results in about $|p|^3$ computation cost. For the previous [21], its computation costs of the user, the cloud and the TPA are $m|p|^2|q|$, $n|p|^2|q|$ and $|p|^4$, respectively.

The above analytical results are collected in Table II. It can be observed that: 1) compared with the state-of-the-art [21],

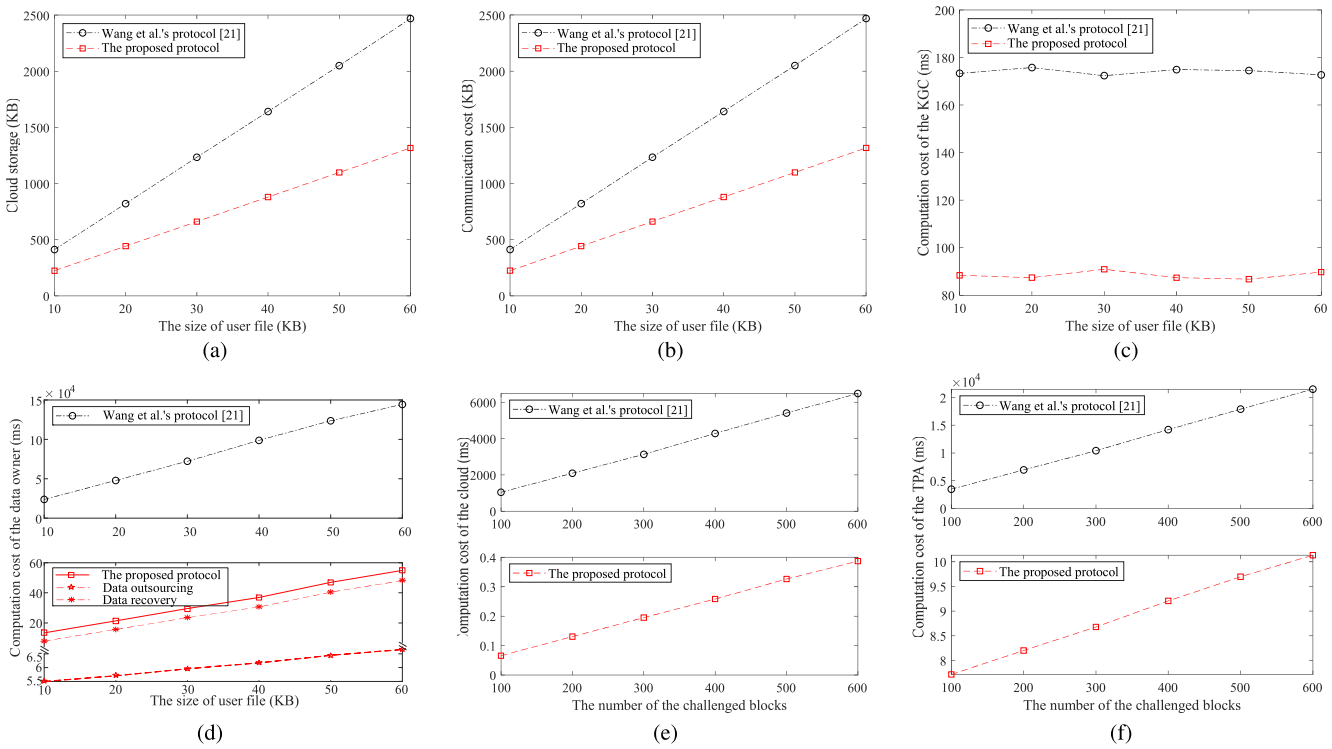


Fig. 3. The performance comparison between the proposed protocols and the classic [21]. (a) Cloud storage. (b) Communication cost. (c) Computation cost of the KGC. (d) Computation cost of the data owner. (e) Computation cost of the cloud. (f) Computation cost of the TPA.

the storage and communication costs of our protocol are almost reduced by half, which is due to no original data blocks involved in cloud storage; 2) the proposed protocol has much lower computation cost than the protocol in [21], since our protocol is implemented by the basic algebraic operations instead of the heavy ECC-based bilinear pairing.

### *B. Experimental Results*

We construct the experiments on a laptop with Intel i5-7200U CPU and 8-GB RAM, constructed using JAVA. In our setting, the lengths of $|p|$, $|q|$ and $|l|$ are set to 1,024 bits, 160 bits and 320 bits. The protocol in [21] is implemented by using the open-source jPBC [40]. To make the security of our protocol and the protocol in [21] at the same level, we determine the elliptic curve in [21] as $y^2 = x^3 + x$ with the 512-bits group elements and the 160-bits group order [18], [40]. To guarantee the accuracy of division operation, we use the internal class "BigDecimal" to store the result of division operation, which can store a large number of decimal digits. In addition, our hash function is implemented by using the standard secure hash algorithms (SHA) of National Institute of Standards and Technology (NIST). Specifically, we repeatedly execute SHA to compute the hash value of the input string concatenated with a random number until the total bit length of these hash values exceeds the size of the mapping field, and then determine the final hash value by eliminating the exceeding bit values.

*1) Storage Cost:* Its experimental results are shown in Fig. 3(a). It can be observed that the storage costs of these two protocols are both linearly proportional to the number of data blocks, which are as expected. We can also observe that the storage cost of the proposed protocol is not only lower than the protocol in [21], but also grows with the number of blocks more slowly than the protocol in [21]. This also implies that the proposed protocol can obtain more and more storage gain with the number of blocks continues to increase. The reason is that the proposed protocol only needs to store the encrypted data blocks and a small-sized tag on the cloud, while the protocol in [21] also needs to store original data blocks at the same time. In addition, cloud storage is more than the size of the original file blocks, which is due to the encryption operations performed on these blocks.

*2) Communication Cost:* Its experimental results are provided in Fig. 3(b). It can be found that the communication costs of these two protocols also increase with the number of blocks, which however are a little higher than their storage costs. The reason is that the communication cost also contains the transmission of the challenge sequences and the integrity proofs in addition to the outsourced data. Furthermore, the communication cost of the proposed protocol is reduced to some extent in comparison to the protocol in [21], and the same observation also holds for the growth rate of the communication cost relative to the number of blocks. It is suggested that the gap of the communication costs between these two protocols would become bigger and bigger as the number of blocks increases. This is because that the proposed protocol only outsources the encrypted data blocks and its integrity proof only contains one aggregated values, while the protocol in [21] outsources two data sets (i.e., file blocks and their authenticators) and its integrity proof is composed of two aggregated values (i.e., the aggregation of data blocks and

authenticators). In addition, the communication cost is also greater than the size of the user file, which is due to the same reason as the storage cost.

*3) Computation Cost:* Its experimental results are given in Figs. 3(c)-3(f). It can be learned that the proposed protocol has computational advantages over the protocol in [21] from the perspectives of the KGC, the user, the cloud, and the TPA, which is as expected. Especially, the proposed protocol hugely reduces the computational costs of the user, the cloud, and the TPA by several orders of magnitude. For the KGC, the computation costs of these two protocols are almost constant. This is due to the fact that the KGC is utilized to setup the whole system parameters, which has no relationship to the number of blocks. For the user, the computation costs of these two protocols are linearly proportional to the number of data blocks, which is because that the user has to calculate the encrypted value of each data block. In our protocol, the computation cost of data recovery is higher than that of data outsourcing, which is due to the division operation with a high precision. For the cloud, the computation costs of these two protocols grow with the number of the challenged blocks incurred by the aggregation operations performed on the challenged data. For the TPA, the computation costs of these two protocols also grow with the number of the challenged blocks. The reason is that the challenged data indices are required to be aggregated during integrity verification. It is worthy to note that the proposed protocol can realize the off-line outsourcing computation on 50 KB user files, the on-line challenge against 500 data blocks within 60 ms and 10 ms, both of which are quite efficient.

## VI. Conclusion

This paper first proposes a novel cryptographical primitive: ID-based PDP with compressed cloud storage, and then investigates a concrete protocol consisting of only basic algebraic operations. In comparison to the existing protocols, the proposed protocol can greatly lower storage, communication, and computation costs. We give strict proof to show that our solution realizes the property of correctness, privacy, unforgeability, and detectability. We also give an illustrative example to show that the proposed protocol can be easily extended to support the other practical functions by using the primitive replacement technique. Finally, we evaluate the performance of the proposed protocol through massive theoretical analysis and experimental simulation, which further validates the effectiveness of the proposed protocol. In the future, it is interesting to design more ID-based PDP protocols with compressed cloud storage. It is also meaningful to study other functional extensions of ID-based PDP with compressed cloud storage in order to fit more application scenarios.

## References


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

[2] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.

[3] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 17, no. 4, pp. 1–29, 2015.

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

[5] J. Zhang, Y. Yang, Y. Chen, and F. Chen, "A secure cloud storage system based on discrete logarithm problem," in *Proc. IEEE/ACM 25th Int. Symp. Quality Service (IWQoS)*, Jun. 2017, pp. 1–10.

[6] F. Chen, F. Meng, T. Xiang, H. Dai, J. Li, and J. Qin, "Towards usable cloud storage auditing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 31, no. 11, pp. 2605–2617, Nov. 2020.

[7] A. Juels and B. S. Kaliski, "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 584–597.

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

[9] G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netowrks*, 2008, pp. 1–10.

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

[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 Trans. Inf. Forensics Security*, vol. 12, no. 10, pp. 2402–2415, Oct. 2017.

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

[13] B. Sengupta, A. Dixit, and S. Ruj, "Secure cloud storage with data dynamics using secure network coding techniques," *IEEE Trans. Cloud Comput.*, early access, Jun. 5, 2020, doi: 10.1109/TCC.2020.3000342.

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

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

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

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

[18] H. Wang, Q. Wang, and D. He, "Blockchain-based private provable data possession," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2379–2389, Sep./Oct. 2021, doi: 10.1109/TDSC.2019.2949809.

[19] Y. Li, Y. Yu, R. Chen, X. Du, and M. Guizani, "IntegrityChain: Provable data possession for decentralized storage," *IEEE J. Sel. Areas Commun.*, vol. 38, no. 6, pp. 1205–1217, Jun. 2020.

[20] Y. Xu, J. Ren, Y. Zhang, C. Zhang, B. Shen, and Y. Zhang, "Blockchain empowered arbitrable data auditing scheme for network storage as a service," *IEEE Trans. Services Comput.*, vol. 13, no. 2, pp. 289–300, Mar. 2020.

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

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

[23] J. Li, H. Yan, and Y. Zhang, "Certificateless public integrity checking of group shared data on cloud storage," *IEEE Trans. Services Comput.*, vol. 14, no. 1, pp. 71–81, Feb. 2021.

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

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

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

[27] Y. Yu *et al.*, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[28] L. Zhou, A. Fu, G. Yang, H. Wang, and Y. Zhang, "Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics," *IEEE Trans. Dependable Secure Comput.*, early access, Aug. 4, 2021, doi: 10.1109/TDSC.2020.3013927.
[29] Y. Li, Y. Yu, G. Min, W. Susilo, J. Ni, and K.-K. R. Choo, "Fuzzy identity-based data integrity auditing for reliable cloud storage systems," *IEEE Trans. Dependable Secure Comput.*, vol. 16, no. 1, pp. 72–83, Jan./Feb. 2019.
[30] W. Shen, J. Qin, J. Yu, R. Hao, and J. Hu, "Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 2, pp. 331–346, Feb. 2019.
[31] H. Wang, D. He, J. Yu, and Z. Wang, "Incentive and unconditionally anonymous identity-based public provable data possession," *IEEE Trans. Services Comput.*, vol. 12, no. 5, pp. 824–835, Sep. 2019.
[32] L. Huang *et al.*, "IPANM: Incentive public auditing scheme for non-manager groups in clouds," *IEEE Trans. Dependable Secure Comput.*, vol. 19, no. 2, pp. 936–952, Mar./Apr. 2022, doi: 10.1109/TDSC.2020.3004827.
[33] J. Xue, C. Xu, J. Zhao, and J. Ma, "Identity-based public auditing for cloud storage systems against malicious auditors via blockchain," *Sci. China Inf. Sci.*, vol. 62, no. 3, pp. 1–16, Mar. 2019.
[34] Y. Yang, Y. Chen, and F. Chen, "A compressive integrity auditing protocol for secure cloud storage," *IEEE/ACM Trans. Netw.*, vol. 29, no. 3, pp. 1197–1209, Jun. 2021.
[35] J. C. Cha and J. H. Cheon, "An identity-based signature from gap Diffie-Hellman groups," in *Proc. Int. Workshop Public Key Cryptogr.*, vol. 2567, 2003, pp. 18–30.
[36] K. G. Paterson, "ID-based signatures from pairings on elliptic curves," *Electron. Lett.*, vol. 38, no. 18, pp. 1025–1026, Nov. 2002.
[37] F. Hess, "Efficient identity based signature schemes based on pairings," in *Selected Areas in Cryptograph*. Berlin, Germany: Springer, 2003, pp. 310–324.
[38] H. Jin, H. Debiao, and C. Jianhua, "An identity based digital signature from ECDSA," in *Proc. 2nd Int. Workshop Educ. Technol. Comput. Sci.*, 2010, pp. 627–630.
[39] V. Shoup, *A Computational Introduction to Number Theory and Algebra*, 2nd ed. Cambridge, U.K.: Cambridge Univ. Press, 2012.
[40] D. Angelo and I. Vincenzo. *JPBC: Java Pairing Based Cryptography*. Accessed: Dec. 4, 2013. [Online]. Available: http://gas.dia.unisa.it/projects/jpbc/

**Yang Yang** received the Ph.D. degree in computer architecture from Wuhan University in 2018. He is currently an Assistant Professor with the Zhongnan University of Economics and Law, China. His research interests include cloud computing security and wireless physical communication.

**Yanjiao Chen** (Senior Member, IEEE) received the Ph.D. degree in computer science and engineering from The Hong Kong University of Science and Technology in 2015. She is currently a Bariren Researcher with Zhejiang University, China. Her research interests include computer networks, wireless system security, cloud computing, and network economy.

**Fei Chen** (Member, IEEE) received the Ph.D. degree in computer science and engineering from The Chinese University of Hong Kong. He joined the College of Computer Science and Engineering, Shenzhen University, China, as a Lecturer, in 2015. His research interests include information and network security, data protection, and privacy.

**Jing Chen** (Member, IEEE) received the Ph.D. degree in computer science from the Huazhong University of Science and Technology, Wuhan. He is currently a Full Professor with the Computer School, Wuhan University. He has published more than 80 research papers in many international journals and conferences, such as IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON MOBILE COMPUTING, INFOCOM, SECON, and TrustCom. His research interests include cloud security and network security.