IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received March 13, 2022, accepted April 4, 2022, date of publication April 12, 2022, date of current version April 20, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3166920

Identity-Based Privacy Preserving Remote Data Integrity Checking With a Designated Verifier

GENQING BIAN[1], RUI ZHANG[1], AND BILIN SHAO[2]
[1]School of Information and Control Engineering, Xi'an University of Architecture and Technology, Xi'an 710055, China
[2]School of Management, Xi'an University of Architecture and Technology, Xi'an 710055, China

Corresponding author: Rui Zhang (zhangrui@xauat.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872284, and in part by the Natural Science Basic Research Program of Shaanxi Province under Grant 2021JLM-16 and Grant 2022JM365.

ABSTRACT The remote data possession checking mechanism can effectively verify the integrity of outsourced data, which can usually be divided into public verification and private verification. The verifier of public verification can be any cloud user, while private verification can only be the data owner. However, in most practical situations, the data owner expects that only a specific verifier can perform integrity checking tasks and that verifier cannot gain any knowledge about the data. Yan *et al.* proposed a remote data possession checking scheme with the designated verifier, which can guarantee that only the designated verifier can check data integrity, whereas others cannot do it. However, this scheme relies on public key infrastructure technology and does not consider data privacy protection issues. To overcome these shortcomings, we propose an identity-based remote data possession checking scheme that satisfies the data owner's requirement to specify a unique verifier. Moreover, in this scheme, we use a random integer to blind data integrity proof to protect data privacy and use Merkle hash tree structure to achieve dynamic update of data. At the same time, our scheme can avoid the complex certificate management in public key infrastructure. We proved the safety of our scheme based on the discrete logarithm assumption and the computational Diffie-Hellman assumption. Theoretical analysis and experimental results show that our scheme is feasible and effective in practical applications.

INDEX TERMS Identity-based cryptography, designated verifier, privacy protection, data dynamics.

I. INTRODUCTION

As an integral part of cloud computing, cloud storage has attracted more and more users to outsource their data to the cloud service provider (CSP) due to its advantages of scale, flexibility, scalability, and economic benefits [1]. However, cloud security issues cannot be ignored [2]. Cloud storage weakens cloud users' physical control over data, leaving users ignorant of data status on the cloud. The CSP may knowingly conceal outsourced data leakage or loss due to various security threats. More seriously, the CSP may intentionally delete data that users do not use frequently to save space in order to provide storage services for more users [3]. In order to verify whether the data is stored completely in the cloud, many remote data possession checking (RDPC) schemes have been proposed.

The associate editor coordinating the review of this manuscript and approving it for publication was Md. Abdur Razzaque.

At present, a majority of the existing RDPC schemes rely on public key infrastructure (PKI) technology. PKI technology requires multiple certificate management operations such as certificate generation, delivery, storage, verification, and revocation, increasing computational and communication costs. In addition, if malicious hackers control the certificate authority (CA), the security of PKI can hardly be fully guaranteed. In order to solve this problem, identity-based cryptography (IBC) came into being. In IBC technology, the user's public key is its identity (ID number, email address, and other information that uniquely identifies the user), avoiding the introduction and management of public-key certificates. Therefore, it is more efficient and secure to use IBC to build an RDPC scheme.

According to the identity of the verifier, RDPC schemes can be divided into: private verification [4], [5] and public verification [6], [7]. The verifier of private verification can only be the data owner, while public verification allows any cloud user to verify data integrity. If private verification is

used, the data owner needs to perform cumbersome checking regularly, which increases the computational cost. In addition, the fairness and authority of the checking results may also be affected. Therefore, most RDPC schemes use public verification to check outsourced data. However, the introduction of the verifier also bring new security requirements. The verifier may be curious about outsourced data and will try to obtain some data content during the integrity checking. If the data stored in the cloud is private or confidential, the curiosity of the verifier is unacceptable for the data owner. Therefore, data privacy protection is essential. Encrypting files before data outsourcing can alleviate data privacy problems, but this approach increases the processing burden for the data owner. It turns privacy protection issues into key management issues, and data breaches can still occur when decryption keys are exposed. Moreover, for shared data, the encrypted data cannot be used by other users. Therefore, it is necessary to consider the data privacy protection issues.

In many realistic situations, the data owner may expect to designate a specific verifier to check outsourced data, while other verifiers cannot perform such work. For example, a user previously verifies private outsourced data by himself. However, he is restricted from surfing the Internet and cannot perform private verification because he is on the battlefield. In this case, he hopes to designate a trusted verifier to check the outsourced data. In addition, because the data is private, the user does not want the verifier to obtain the data content. Another example is a company storing its business information in the cloud. Its competitors may fake identities to verify data and obtain business information about the company. Therefore, the company needs to designate a verifier, and due to the confidentiality of business information, it needs to consider data privacy protection. In both of the above cases, private or public verification does not apply. Yan *et al.* [8] proposed a scheme to ensure that only the designated verifier can check the data integrity.

Since outsourced data is not always static, and the data owner may need to update data frequently, it is critical to support dynamic data operations. We improve the scheme in [8] and propose an identity-based RDPC scheme with the designated verifier that supports privacy protection and dynamic data operations, which can better adapt to the actual situation.

A. MOTIVATION AND CONTRIBUTION

In this paper, we improve the RDPC scheme with the designated verifier in [8]. Considering the complex certificate management operations in PKI and the semi-trusted problem of the verifier, we propose a new RDPC scheme. The main contributions of this paper are as follows:

1) Based on the IBC technology, we implement an RDPC scheme with a designated verifier, avoiding the problem of certificate management.
2) Our scheme achieves data privacy. The CSP uses a random integer to blind the data integrity proof, ensuring that the verifier does not obtain any data content.
3) Our scheme uses the Merkle hash tree (MHT) to support dynamic data operations and meet the requirements of frequent data updates.
4) We prove our scheme's security and evaluate the scheme's computation cost and communication cost. Finally, experimental results show that our scheme has feasibility and better efficiency.

B. RELATED WORK

In 2007, Ateniese *et al.* [5] first proposed a provable data possession (PDP) model to ensure data ownership on untrusted clouds. Jules *et al.* [9] defined a proof of recoverability (POR) model that not only verifies data integrity but also restores data if it becomes corrupt, but the data owner can only verify the data a limited number of times. To solve this problem, Shacham and Waters [10] improved [9] to design a compact POR that can verify data infinitely. Many researchers put forward various schemes to improve data security and audit efficiency based on the PDP and POR models.

According to the verification mode, most schemes are mainly divided into private verification and public verification. They do not satisfy the situation where the data owner only allows a specific verifier to check the data. To solve this problem, Ren *et al.* [11] presented a designated-verifier PDP scheme. However, Yan *et al.* [8] proved that this scheme is not resistant to replay attacks and designed a new RDPC protocol with a designated verifier. Shen *et al.* [12] proposed a delegable PDP model that allows trusted third parties to check the data integrity at the authority of the data owner. However, this model needs to check all data blocks for each challenge during the verification process, increasing the computational cost. Wang [13] introduced proxy cryptography into the cloud computing field, relying on the warrant to delegate the proxy to check the data possession.

The schemes above rely on PKI technology and require complex and time-consuming certificate management operations. Wang [14] proposed an identity-based RDPC scheme in multi-cloud storage. In this scheme, the user's public key is a unique identifier. Chang *et al.* [15] first introduced the related-key attack problem into an identity-based signature scheme and defined a security model for it. Chen and Chang [16] revealed an intrinsic relationship between identity-based proof of retrievability and identity-based network coding, enriching the current constructions for both network coding and cloud storage.

It should be noted that the above scheme checks the integrity of the data by a third-party verifier. Considering the problem that verifier may leak the data owner's data, many integrity checking schemes [17], [18] and [19] with privacy protection have been proposed. Wang *et al.* [20] designed a scheme using homomorphic linear authenticators and random masks. To reduce certificate management operations, Zhang and Dong [21] proposed a scheme of identity-based public auditing with data privacy protection. However, the scheme has high storage cost, and it cannot resist attacks from

TABLE 1. Functionality comparison with related schemes.

Schemes	The designated verifier	Privacy protection	Dynamic data operations	Type
Scheme in [8]	Yes	No	No	PKI
Scheme in [11]	Yes	No	No	PKI
Scheme in [12]	Yes	No	No	PKI
Scheme in [13]	Yes	No	No	PKI
Scheme in [22]	No	Yes	No	IBC
Scheme in [23]	No	Yes	No	IBC
Scheme in [25]	No	No	Yes	IBC
Scheme in [27]	No	Yes	Yes	PKI
Scheme in [29]	No	Yes	Yes	PKI
Scheme in [30]	No	Yes	Yes	PKI
Our scheme	Yes	Yes	Yes	IBC

untrusted CSP. Yu *et al.* [22] proposed an identity-based protocol with data privacy protection and formalized the security model of zero-knowledge privacy. Li *et al.* [23] proposed an identity-based RDPC scheme that makes use of a homomorphic verifiable tag to decrease the system complexity.

To support dynamic data operations on the cloud, Wang *et al.* [24] first explained the problems that may be encountered in embedding data block indices into tags. Such as the tag of the file block after the index of the inserted data block needs to be recalculated, and the malicious CSP may use tags of the new and old blocks with the same index to alter any other blocks and their tags. Then they proposed a protocol supporting fully dynamic operations on the block level using the MHT. Shang *et al.* [25] proposed an identity-based dynamic data audit scheme. However, Li *et al.* [26] pointed out that in this scheme the CSP can obtain the user's private key from the tag. While supporting dynamic data operations, privacy protection issues still need to be considered. Sun *et al.* [27] proposed a privacy protection method in the cloud big data streaming environment. However, Li *et al.* [28] proved that the scheme could not resist the untrusted CSP. If the CSP forges the cloud data and tags, it could still pass the verification. Liu *et al.* [29] proposed a public cloud audit scheme that supports dynamic data and privacy protection. However, in this scheme, the data owner first encrypts the data, and then the verifier calculates the digital signature, reducing the security and flexibility of the scheme. Wu *et al.* [30] proposed a mechanism combining data deduplication with dynamic data operations in the privacy-preserving public auditing for secure cloud storage.

We perform a functional comparison of our scheme with several related schemes, as shown in Table 1. In terms of the designated verifier, schemes in [8], [11]–[13] designate a verifier by embedding the verifier information into the tags or using an authorization. However, other schemes in Table 1 use public verification that any verifier can perform checking tasks. In terms of privacy protection, schemes in [22], [23], [27] and [30] combines homomorphic verification with random mask, and scheme in [29] encrypts and then signs the data block, all of which ensure that the verifier knows nothing about the data content during the data integrity checking. However, other schemes in Table 1 do not consider that computing the linear combination of data blocks in the data integrity proof may deduce the data content. In terms of dynamic data operations, schemes in [25], [29] and [30] respectively adopt different data structures to support data updates effectively. The schemes in [27] can adaptively extend the authentication data structure they define when data streams arrive dynamically. Nevertheless, other schemes focus on static data and also cannot be directly extended to dynamic data schemes. The schemes in [22], [23] and [25] are based on IBC, whose public key is the user's identity. However, other schemes rely on PKI technology, which requires a certificate issued by the CA to ensure the authenticity of the user's public key. Moreover, our scheme is the only one that can meet all the above functions.

C. ORGANIZATION

The paper is organized as follows. The preliminaries of our scheme are introduced in Section II. Section III gives the concrete construction of our scheme and its security analysis. The performance analysis of our scheme is demonstrated in Section IV. Finally, we conclude our paper in section V.

II. PRELIMINARIES

In this section, we briefly introduce bilinear mapping, discrete logarithm assumption, computational Diffie-Hellman assumption, and Merkle hash tree structure.

A. BILINEAR MAPS

Let G_1 and G_2 are multiplicative cyclic groups with large prime order q. $e : G_1 \times G_1 \rightarrow G_2$ is a blinear map, which has the following properties:

1) Computability: for all $\mu, \nu \in G_1$, there exists an efficiently computable algorithm to calculate $e(\mu, \nu)$.
2) Blinearity: for all $a, b \in Z_q^*$ and $\mu, \nu \in G_1$, it holds that
$$e(\mu^a, \nu^b) = e(\mu, \nu)^{ab}.$$
3) Non-degeneracy: for any $\mu, \nu \in G_1$, it has $e(\mu, \nu) \neq 1_{G_2}$.

B. DISCRETE LOGARITHM ASSUMPTION

1) Discrete Logarithm (DL) Assumption
Suppose g is a generator of a multiplicative cyclic group G_1. For any probabilstic polynomial time (PPT)

algorithm $\mathcal{A}$, given g^a with the unknown elements $a \in Z_q^*$, the advantage for $\mathcal{A}$ to compute a in G_1 is negligble, which can be defined as

$$Adv_{G_1\mathcal{A}}^{DL} = Pr[\mathcal{A}(g, g^a) = a : a \overset{R}{\leftarrow} Z_q^*] \leq \varepsilon.$$

2) A variant of DL Assumption
Suppose G_1 is a multiplicative cyclic group. Given the tuple $(g_0, g_0^a) \in G_1$, where $a \in Z_q^*$, $g_0 \in G_1$. For any PPT algorithm $\mathcal{A}$, the advantage is negligble for $\mathcal{A}$ to compute a in G_1. It can be expressed as the following:

$$Adv_{G_1\mathcal{A}}^{DL'} = Pr[\mathcal{A}(g_0, g_0^a) = a : a \overset{R}{\leftarrow} Z_q^*, g_0 \overset{R}{\leftarrow} G_1] \leq \varepsilon.$$

C. COMPUTATIONAL DIFFIE-HELLMAN ASSUMPTION

Suppose G_1 is multiplicative cyclic group, where g is one of its generators. Given the tuple (g, g^a, g^b) with the unknown elements $a, b \in Z_q^*$, the advantage for a PPT algorithm $\mathcal{A}$ to compute g^{ab} in G_1 has no more than ε, which is negligible.

Computational Diffie-Hellman (CDH) assumption can be defined as

$$Adv_{G_1\mathcal{A}}^{CDH} = Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \overset{R}{\leftarrow} Z_q^*] \leq \varepsilon.$$

D. MERKLE HASH TREE

MHT is an authentication structure used to check the integrity of a set of elements. MHT is a binary tree, which leaf nodes store the hash value of the corresponding file block, and the parent node stores the aggregated hash value of the two-child nodes. As shown in Figure. 1, node G stores the hash value $H(G) = H(m_1)$, node C stores the hash value $H(C) = H(G\|I) = H(H(m_1)\|H(m_2))$. Calculating the hash value of each node in the whole tree structure from the bottom up, we finally get the value of the root node.

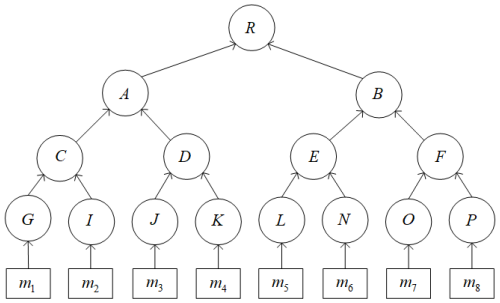

FIGURE 1. Merkle hash tree structure.

Through this tree structure, the integrity of the data block can be verified efficiently and safely. Taking the data block m_3 as an example, its auxiliary authentication information (AAI) is $\{H(C), H(K), H(B)\}$, which is composed of all sibling nodes on the internal node in the root-to-leaf path. The CSP sends m_3 and $proof = \{AAI, sig(H(R))\}$ to the verifier, and the verifier reconstructs MHT and calculates $H(R')$, as shown in Figure. 2 (The gray nodes are the information in AAI). Finally, the verifier checks whether $sig(H(R))$ is the signature of $H(R')$. If the check passes, the data block m_3 is proved to be complete.

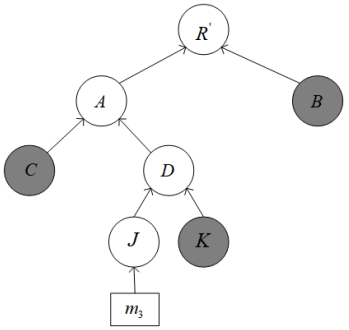

FIGURE 2. Integrity verification based on the MHT.

E. SYSTEM MODEL

The system comprises four kinds of different entities: the key generation center (KGC), the data owner, the CSP, and the designated verifier, which are described in detail as follows.

1) The KGC: This entity generates public parameters and the user's private key.
2) The CSP: This entity has a lot of storage and computing resources. The CSP stores outsourced data from the data owner. Moreover, the CSP generates a data integrity proof to the verifier based on the challenge message.
3) The data owner: This entity can outsource data files to the CSP. The data owner can also designate a verifier to perform the integrity checking task.
4) The designated verifier: This entity is designated by the data owner to check outsourced data.

Figure. 3 shows the system model for our scheme. We assume that the CSP is untrusted. The CSP may forge proof to pass data integrity checking when outsourced data is corrupted. The CSP may not honestly update the data owner's data, but use previous blocks and tags to deceive the data owner. We also assume the verifier is semi-trusted because he is honest-but-curious. On the one hand, the verifier honestly verifies the data integrity proof and returns the verification result correctly to the data owner. On the other hand, when

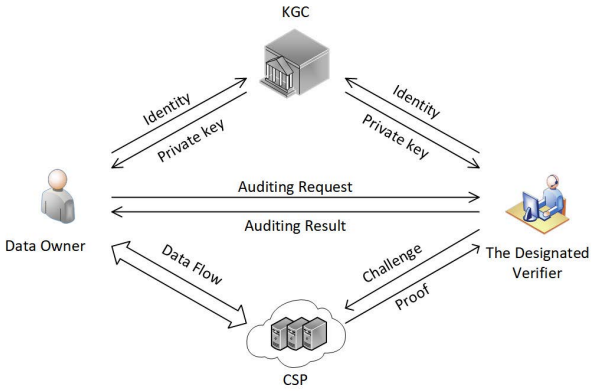

FIGURE 3. System Model.

verifying the integrity proof, the verifier may try to obtain the relevant content of the data, threatening user data security.

This scheme includes the nine polynomial algorithms.

1) $Setup(1^{\lambda}) \rightarrow (params, msk)$. The algorithm is executed by the KGC, which inputs a security parameter λ and outputs the master key msk and public system parameters $params$.
2) $Extract(ID, msk) \rightarrow sk_{ID}$. The algorithm is executed by the KGC, inputting user identity ID and outputting the user's private key sk_{ID}.
3) $TagGen(sk_O, F) \rightarrow T$. The algorithm is executed by the data owner. It inputs the private key sk_O of the data owner and the original file F and outputs the tag set T.
4) $Challenge(l) \rightarrow chal$. The algorithm is executed by the designated verifier, inputting the number l of challenge blocks and outputting the challenge message $chal$.
5) $ProofGen(chal, F, T) \rightarrow \Gamma$. The algorithm is executed by the CSP. It inputs challenge message $chal$, file F, and tag set T, and outputs a data integrity proof Γ.
6) $Verify(\alpha, chal, \Gamma) \rightarrow 1/0$. The algorithm is executed by the designated verifier, which inputs challenge message $chal$, the proof Γ and outputs 0 or 1. If the output is 1, it indicates that the data stored in the cloud is complete. Otherwise, the data is incomplete.
7) $UpdateGen() \rightarrow \xi$. The algorithm is executed by the data owner to generate an update request information ξ.
8) $UpdateExec(\xi, F, T) \rightarrow \delta$. The algorithm is executed by the CSP. It inputs the update request information ξ, and outputs an update proof δ.
9) $UpdateVerify(\delta) \rightarrow 0/1$. The algorithm is executed by the data owner to check the update proof δ by CSP. If δ is correct, it outputs 1 otherwise 0.

F. SECURITY MODEL

In this section, we define the security model for our scheme and consider the security requirements of the scheme in the following three areas.

1) This scheme can resist untrusted CSP. The data integrity proof can be verified successfully only if the CSP has fully stored the outsourced data.
2) This scheme is resistant to the undesignated verifier. Only the designated verifier can help data owners verify the integrity of outsourced data.
3) This scheme can solve the problem of verifier semi-trusted. The designated verifier cannot obtain the any data contents from the data integrity proof.

The first security requirement is defined as follows.

To formalize the security model, we use a security game $Game1$ to capture the first security requirement to show how the adversary $\mathcal{A}$ goes against the security of this scheme. The $Game1$ involves two roles, where the adversary $\mathcal{A}$ represents an untrusted CSP and the challenger $\mathcal{C}$ simulates the environment for $\mathcal{A}$. The game includes the following phases:

1) **Setup**: $\mathcal{C}$ runs the $setup$ algorithm to generate the master key msk and public parameters $params$. Then $\mathcal{C}$ sends $params$ to $\mathcal{A}$ and keeps msk secret.
2) **Queries**: $\mathcal{A}$ executes polynomial times of queries to $\mathcal{C}$, $\mathcal{C}$ responds to $\mathcal{A}$'s queries as follows:
   a) *Hash queries*: $\mathcal{A}$ performs the hash query adaptively to the $\mathcal{C}$. $\mathcal{C}$ computes the corresponding hash value and returns it to $\mathcal{A}$.
   b) *Extract queries*. $\mathcal{A}$ queries the private key for user identity ID_i. $\mathcal{C}$ runs $Extract$ algorithm to calculate its private key sk_{ID_i} and returns the sk_{ID_i} to $\mathcal{A}$.
   c) *Tag queries*: $\mathcal{A}$ queries the tag of data block with any user identity ID_i. $\mathcal{C}$ runs $TagGen$ algorithm to compute the corresponding tag and returns it to $\mathcal{A}$.
3) **ProofCheck**: $\mathcal{A}$ and $\mathcal{C}$ act as the CSP and the verifier, respectively. $\mathcal{C}$ runs $Challenge$ algorithm and sends $chal$ to $\mathcal{A}$. $\mathcal{A}$ performs $ProofGen$ algorithm to generate a integrity proof Γ and sends Γ to $\mathcal{C}$. Finally, $\mathcal{C}$ runs $Verify$ algorithm to obtain the verification result 0/1.
4) **Forgery**: $\mathcal{C}$ submits a new challenge message $chal^*$ to $\mathcal{A}$. $\mathcal{A}$ generates Γ^* for the data blocks indicated in $chal^*$. If Γ^* is not equal to the correct proof Γ and

$$Verify(\alpha, chal^*, \Gamma^*) \rightarrow 1,$$

$\mathcal{A}$ wins this game.

Definition 1: If any PPT adversary $\mathcal{A}$ can only win the $Game1$ with negligible probability, then this scheme is effective against the untrusted CSP.

We define the second safety requirement as follows.

We are still building a secure game $Game2$ played between a challenger $\mathcal{C}'$ and an adversary $\mathcal{A}'$.

1) **Setup**, **Queries** and **ProofCheck**: The interaction between $\mathcal{C}'$ and $\mathcal{A}'$ in "Setup" phase and "Queries" phase is similar to $Game1$. In addition, $\mathcal{A}'$ asks $\mathcal{C}'$ for $H_1(ID_O)$ and $H_1(ID_V)$. $\mathcal{C}'$ calculates them and responds to $\mathcal{A}'$, but sk_O and sk_V are confidential and unknown to $\mathcal{A}'$. $\mathcal{A}'$ cannot perform the query in "ProofCheck" phase since $\mathcal{A}'$ is the undesignated verifier.
2) **Forgery**: $\mathcal{A}'$ forges α^* with the response returned by the CSP. The true α is calculated from $H_1(ID_O)$ of the data owner and private key sk_V of the designated verifier. If α^* and α are equal, $\mathcal{A}'$ wins $Game2$.

Definition 2: In the case of authorizing the designated verifier, if the probability of the PPT adversary $\mathcal{A}'$ winning $Game2$ is negligible, then the scheme is safe and the undesignated verifier cannot perform checking tasks.

Finally, we define the definition of the third safety requirement. We are still building a secure game $Game3$ played between a challenger C'' and an adversary A''.

1) **Setup**: The interaction between $\mathcal{C}''$ and A'' is the same as that the interaction between $\mathcal{C}$ and A in $Game1$.
2) **Queries**: Expect $\mathcal{A}''$ cannot perform tag queries on $\mathcal{C}''$, the rest of the queries are the same as in $Game1$.
3) **ProofCheck**: In this phase, A'' acts as the designated verifier and C'' acts as the CSP. A'' generates the challenge message $chal$ and sends it to $\mathcal{C}''$. $\mathcal{C}''$ honestly executes the $proofGen$ algorithm to generate the proof Γ and sends it to A'', A'' performs the $Verify$ algorithm and returns the result value 0 or 1.

4) **Forgery**: A'' performs polynomial operations on the information in the proof Γ to retrieve its contents, and regenerates the proof Γ^*. If Γ^* can pass the *Verify* algorithm, then A'' succeeds in the above game.

Definition 3: The scheme can solve the problem that the verifier's semi-trusted if the probability of either the PPT adversary $\mathcal{A}$ win the *Game*3 is negligible.

III. OUR PROPOSED SCHEME

We give the concrete construction of our scheme in this section.

A. CONSTRUCTION OF SCHEME

$Setup(1^\lambda) \rightarrow (params, msk)$: On input a security parameter λ, the KGC randomly selects a big prime q. G_1 and G_2 are two multiplicative groups whose orders are both q, where g is a generator of G_1. $\pi: Z_q^* \times \{1, \ldots, n\} \rightarrow \{1, \ldots, n\}$ is a pseudorandom permutation (PRP) and $\phi: Z_q^* \times \{1, \ldots, n\} \rightarrow Z_q^*$ is a pseudorandom function (PRF). Then the KGC chooses a bilinear map $e : G_1 \times G_1 \rightarrow G_2$ and two secure hash functions H_1 and $H_2 : \{0, 1\}^k \rightarrow G_1$. H is a cryptographic hash function. Finally the KGC randomly selects a value $x \in Z_q^*$ as the master secret key and calculates $P_0 = g^x$ as the master public key.

The public system parameters are $params = (q, g, G_1, G_2, e, H_1, H_2, H, \pi, \phi, P_0)$ and the master secret key $msk = x$.

$Extract(ID, msk) \rightarrow sk_{ID}$: After receiving the user's identity ID, the KGC calculates user's private key

$$sk_{ID} = H_1(ID)^x$$

using msk and returns it to user.

The KGC obtains the private keys of the data owner and the designated verifier through the following calculations, Where ID_O and ID_V are their identities, respectively.

$$sk_O = H_1(ID_O)^x, sk_V = H_1(ID_V)^x.$$

$TagGen(sk_O, F) \rightarrow T$: Given the data file F named $Fid \in \{0, 1\}^\lambda$, the data owner splits F into n blocks, and then divides each block into s sectors. That is

$$F = \{m_{i,j}\}_{\substack{1 \le i \le n \\ 1 \le j \le s}},$$

in which m_{ij} represents the j-th sector of the i-th data block in the file F. The data owner continues to do the following:

1) Based on his private key sk_O and the designated verifier's identity $H_1(ID_V)$, the data owner calculates

$$\alpha = e(sk_O, g) \cdot e(H_1(ID_V), P_0).$$

2) The data owner randomly chooses $t \in Z_q^*$ and s elements $u_1, u_2, \cdots, u_s \in G_1$, computes $\chi = g^t$. Then the data owner generates the tag T_i for each data block m_i by using the following equation:

$$T_i = sk_O \cdot (H_2(\alpha \| Fid \| H(m_i)) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}})^t.$$

3) The data owner sets $\tau_0 = u_1 \| u_2 \| \cdots \| u_s \| \chi \| Fid$, and compute

$$\tau = \tau_0 \| IDS(\tau_0),$$

where IDS is an identity-based signature [31].

4) The data owner uses the hash value $H(m_i)$ of each data block m_i as leaf nodes, builds the MHT from the bottom up. Then he calculates the hash value $H(R)$ of the root node. The data owner uses an IDS algorithm to compute

$$sig(H(R)) = IDS(H(R)).$$

5) The data owner uploads

$$T = (F, \{u_j\}_{1 \le j \le s}, \tau, \chi, \{T_i\}_{1 \le i \le n}, sig(H(R)))$$

to the CSP, and deletes it from local storage.

$Challenge(l) \rightarrow chal$: The designated verifier randomly selects $l \in [1, n]$ as the total number of challenge blocks and $k_1, k_2 \in Z_q^*$ as seeds of PRF and PRP, then sends $chal = (l, k_1, k_2)$ as the challenge message to the CSP.

$ProofGen(chal, T) \rightarrow \Gamma$: After receiving $chal$, the CSP generates a proof Γ. The specific process is as follows:

1) The CSP first calculates set $S = \{(a_i, c_i) | i \in [1, l]\}$, where a_i is random parameter and c_i is the index of the i-th challenged block:

$$a_i = \phi(k_1, i), c_i = \pi(k_2, i).$$

2) The CSP randomly selects $w \in Z_q^*$,and computes

$$\sigma = \prod_{(a_i, c_i) \in S} T_{c_i}^{a_i}, M_j = \sum_{(a_i, c_i) \in S} a_i \cdot m_{c_i, j} + w$$

and $W_j = (u_j)^w$ for each u_j.

3) The CSP provides $\{H(m_i)\}_{c_1 \le i \le c_l}$ of each challenged block and their corresponding AAI $\{\Omega_i\}_{c_1 \le i \le c_l}$ in MHT, and sets $\Lambda = \{\{H(m_i), \Omega_i\}_{c_1 \le i \le c_l}\}$.

4) Finally, the CSP defines a data integrity proof

$$\Gamma = (\{u_j, W_j\}_{1 \le j \le s}, Fid, \tau, \sigma, \{M_j\}_{1 \le j \le s}, \Lambda, sig(H(R)))$$

and sends Γ to the designated verifier.

$Verify(\alpha, chal, \Gamma) \rightarrow 1/0$: The procedure for the designated verifier to verify the proof Γ is as follows:

1) The designated verifier generates $H(R')$ using $\{H(m_i), \Omega_i\}_{c_1 \le i \le c_l}$, and checks whether $sig(H(R))$ is the signature of $H(R')$. If it isn't, the designated verifier outputs 0, else continues to execute.

2) The designated verifier gets τ_0 and $IDS(\tau_0)$ from τ, then checks if $IDS(\tau_0)$ is a valid signature of τ_0. If it is not, the designated verifier output 0.

3) Based on his private key sk_V and the data owner's identity $H_1(ID_O)$, the designated verifier calculates

$$\alpha = e(sk_V, g) \cdot e(H_1(ID_O), P_0).$$

4) The designated verifier calculates $a_i = \phi(k_1, i)$, $c_i = \pi(k_2, i)$.

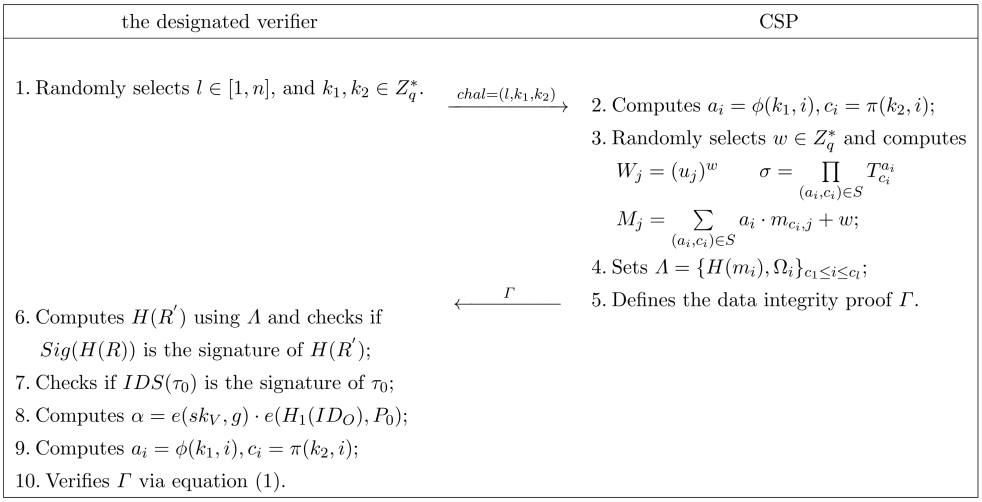

FIGURE 4. The process of integrity checking.

5) The designated verifier checks if the equation

$$\begin{aligned} & e(\sigma, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ & = e(H_1(ID_O)^{\sum_{(a_i,c_i)\in S} a_i}, P_0) \\ & \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \cdot \prod_{j=1}^{s} u_j^{M_j}, \chi). \quad (1) \end{aligned}$$

If the equation holds, it means that the data on the cloud is complete, then the designated verifier outputs 1, otherwise outputs 0.

Figure 4 shows the interaction process between the CSP and the designated verifier in the data integrity checking. The correctness of the equation (1) can be proved as follows:

$$\begin{aligned} & e(\sigma, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ & = e(\prod_{(a_i,c_i)\in S} T_{c_i}^{a_i}, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ & = e(\prod_{(a_i,c_i)\in S} (sk_O \cdot (H_2(\alpha \| Fid \| H(m_{c_i})) \cdot \prod_{j=1}^{s} u_j^{m_{c_i,j}})^t)^{a_i}, g) \\ & \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ & = e(\prod_{(a_i,c_i)\in S} (sk_O)^{a_i}, g) \cdot e(\prod_{(a_i,c_i)\in S} (\prod_{j=1}^{s} (u_j^{m_{c_i,j}}))^{a_i}, \chi) \\ & \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i}, \chi) \cdot e(\prod_{j=1}^{s} (u_j)^w, \chi) \\ & = e(\prod_{(a_i,c_i)\in S} H_1(ID_O)^{a_i}, g^x) \cdot e(\prod_{j=1}^{s} u_j^{\sum_{(a_i,c_i)\in S} a_i \cdot m_{c_i,j} + w}, \chi) \\ & \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i}, \chi) \\ & = e(H_1(ID_O)^{\sum_{(a_i,c_i)\in S} a_i}, P_0) \cdot e(\prod_{j=1}^{s} u_j^{M_j}, \chi) \\ & \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i}, \chi) \\ & = e(H_1(ID_O)^{\sum_{(a_i,c_i)\in S} a_i}, P_0) \\ & \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \cdot \prod_{j=1}^{s} u_j^{M_j}, \chi). \end{aligned}$$

B. DYNAMICS DATA OPERATIONS

When the data owner wants to update outsourced data, the traditional approach is to download all data from the cloud and then upload the locally updated data and tags to the cloud, incurring high computation and communication costs. In our scheme, we use MHT to support dynamic operations. It is assumed that the data blocks and the corresponding tags have been stored in the cloud, and the CSP holds $Sig(H(R))$ that the data owner's signature to the MHT root node. The specific dynamic operation is shown below.

1) DATA MODIFICATION

The data owner changes the specified data block that has been stored in the cloud to the new data block, the steps are shown in Figure 5.

1) $UpdateGen() \rightarrow \xi$. If the i-th data block m_i needs to be modified to m_i^*, the data owner first divides m_i^* into s sectors $m_i^* = (m_{i,1}^*, m_{i,2}^*, \cdots, m_{i,s}^*)$, and then uses the $TagGen()$ algorithm to generate tag T_i^*.

$$T_i^* = sk_O \cdot (H_2(\alpha \| Fid \| H(m_i^*)) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}^*})^t$$

the data owner		CSP
1. Generates tag for m_i^* $T_i^* = sk_O \cdot (H_2(\alpha \Vert Fid \Vert H(m_i^*)) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}^*})^t$;		
2. Sets $\xi = \{\mathbb{M}, i, m_i^*, T_i^*\}$.	$\xrightarrow{\xi}$	3. Replaces m_i, T_i with m_i^*, T_i^*, respectively; 4. Replaces $H(m_i^*)$ with $H(m_i)$ in MHT and computes R^*;
6. Computes R' using $\{\Omega_i, H(m_i)\}$ and check if $Sig(H(R))$ is $H(R')$'s signature; If not, output 0;	$\xleftarrow{\delta}$	5. Sets $\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\}$.
7. Computes R'' using $\{\Omega_i, H(m_i^*)\}$ and checks if $R^* = R''$, if so, computes $Sig(H(R^*))$.	$\xrightarrow{Sig(H(R^*))}$	8. Replaces $Sig(H(R))$ with $Sig(H(R^*))$.

FIGURE 5. The process of the data block modification.

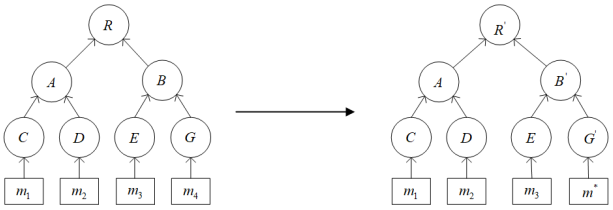

FIGURE 6. MHT update for modification operation.

Finally, the data owner generates $\xi = \{\mathbb{M}, i, m_i^*, T_i^*\}$ and sends it to the CSP, where $\mathbb{M}$ stands for the modification operation, and i represents the subscript of the data block that needs to be modified.

2) $UpdateExec(\xi, F, T) \rightarrow \delta$. After receiving ξ, the CSP performs the modification operation and generates an update proof δ.
   a) The CSP replaces the m_i, T_i with the m_i^* and T_i^*, respectively.
   b) The CSP replaces $H(m_i)$ with $H(m_i^*)$ in the MHT, and then updates the MHT to obtain the new root node R^* is obtained. The process is shown in Figure. 6, to modify the fourth data block, the stored hash value of the leaf node G is modified to $H(m^*)$.
   c) The CSP generates
   $$\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\},$$
   and sends it to the data owner, where Ω_i is the corresponding AAI to m_i.

3) $UpdateVerify(\delta) \rightarrow 0/1$. The data owner verifies δ sent by the CSP and checks whether the CSP updates the data file as required.
   a) The data owner uses $\{\Omega_i, H(m_i)\}$ to obtain root node R'. Then the data owner checks the $Sig(H(R)))$ is the signature of $H(R')$. If it is true, the data owner proceeds to do the following. Otherwise output 0.
   b) The data owner uses $H(m_i^*)$ and Ω_i to compute the new root node R'', and compares it with R^* in δ. If equal, the data owner uses IDS algorithm to compute $Sig(H(R^*))$ and outputs 1. Otherwise it outputs 0.
   c) The data owner deletes the
   $$\{m_i^*, T_i^*, Sig(H(R^*)), \delta\}$$
   of the local store, and sends $Sig(H(R^*))$ to CSP.

2) DATA INSERTION

The data owner inserts a new data block after the specific data block stored in the outsourced file. Figure. 7 describes the specific steps of data block insertion.

1) $UpdateGen() \rightarrow \xi$. The data owner generates tag T^* for the data block m^* to be inserted, and sends the update request information $\xi = \{\mathbb{I}, i, m^*, T^*\}$ to the CSP, where $\mathbb{I}$ stands for the insertion operation.
2) $UpdateExec(\xi, F, T) \rightarrow \delta$. The CSP inserts m^* and T^* after m_i and T_i, respectively. After the leaf node m_i, the CSP inserts a new leaf node, which stores $H(m^*)$. Then the CSP updates the MHT to obtain the new root node R^*. The example is described in Figure. 8. After inserting leaf node I storing $H(m^*)$ after leaf node G, MHT adds an intermediate node J, where J is the parent node of I and G and its hash value is $H(J) = H(H(m_4) \Vert H(m_{m^*}))$. Finally, the CSP sends
   $$\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\}$$
   to the data owner.
3) $UpdateVerify(\delta) \rightarrow 0/1$. First, the data owner computes R' with $\{\Omega_i, H(m_i)\}$, then verifies whether $Sig(H(R))$ is the signature of $H(R'))$. If it is not, output 0. Second, the data owner compares R' and R^*, where R' is calculated based on $H(m^*)$ and $\{\Omega_i, H(m_i)\}$, and if not equal, it outputs 0. Third, the data owner uses IDS to compute $Sig(H(R^*))$ and sends it to the CSP. Finally, the data owner deletes $\{m^*, T^*, Sig(H(R^*)), \delta\}$ and outputs 1.

the data owner		CSP
1. Generates tag for m^* $T^* = sk_O \cdot (H_2(\alpha \Vert Fid \Vert H(m^*)) \cdot \prod_{j=1}^{s} u_j^{m^*_{i,j}})^t$;		
2. Sets $\xi = \{\mathbb{I}, i, m^*, T^s*\}$.	$\xrightarrow{\xi}$	3. Inserts m^*, T^* after m_i, T_i respectively; 4. Uses $H(m^*)$ as the i+1-th leaf node to update MHT, and computes R^*;
6. Computes $H(R^{'})$ using $\{\Omega_i, H(m_i)\}$. Verifies $Sig(H(R))$ by $H(R^{'})$. If not, output 0;	$\xleftarrow{\delta}$	5. Sets $\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\}$.
7. Computes $R^{''}$ using $H(m^*)$ and $\{\Omega_i, H(m_i)\}$; Checks if $R^* = R^{''}$, if so, compute $Sig(H(R^*))$.	$\xrightarrow{Sig(H(R^*))}$	8. Replaces $Sig(H(R))$ with $Sig(H(R^*))$.

FIGURE 7. The process of the data block insertion.

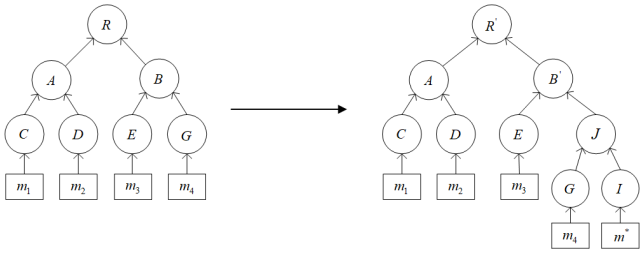

FIGURE 8. MHT update for insertion operation.

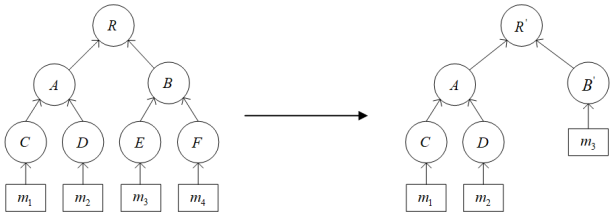

FIGURE 10. MHT update for deletion operation.

the data owner		CSP
1. Sets $\xi = \{\mathbb{D}, i\}$.		
	$\xrightarrow{\xi}$	2. Deletes the i-th leaf node in MHT and computes R^*;
4. Computes $H(R^{'})$ using $\{\Omega_i, H(m_i)\}$. Verifies $Sig(H(R))$ by $H(R^{'})$, if not, output 0 ;	$\xleftarrow{\delta}$	3. Sets $\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\}$.
5. Computes $R^{''}$ using Ω_i; Checks if $R^* = R^{''}$, if so, compute $Sig(H(R^*))$.	$\xrightarrow{Sig(H(R^*))}$	6. Replaces $Sig(H(R))$ with $Sig(H(R^*))$.

FIGURE 9. The process of the data block deletion.

3) DATA DELETION

The data owners deletes specific data blocks stored in the cloud. Furthermore, we show the operation flow of the CSP and the data owner during data block deletion in Figure. 9.

1) $UpdateGen() \rightarrow \xi$. The data owner sends the update request information $\xi = \{\mathbb{D}, i\}$ to the CSP, that $\mathbb{D}$ stands for the delete operation needs to be performed on the i-th data block.
2) $UpdateExec(\xi, F, T) \rightarrow \delta$. The CSP calculates Ω_i of the i-th leaf node. Then he deletes m_i, T_i and the i-th leaf node from F, T and MHT, respectively. The CSP updates MHT to obtain the new root node R^*, as in the example in Figure. 10. Finally the CSP sends
$$\delta = \{\Omega_i, H(m_i), Sig(H(R)), R^*\}$$
to the data owner.
3) $UpdateVerify(\delta) \rightarrow 0/1$. According $\{\Omega_i, H(m_i)\}$ in the δ, the data owner calculates R', and verifies whether $Sig(H(R))$ is the signature of $H(R')$). If the verification fails, it output 0. Then the data owner computes the new root node R'' according Ω_i. If R'' and R^* are not equal, the data owner output 0. Moreover, the data owner calculates $Sig(H(R^*))$ using IDS and sends it to the CSP. Finally, the data owner deletes $\{Sig(H(R^*)), \delta\}$ from the local storage and outputs 1.

C. SECURITY ANALYSIS

In this section, we analyze the security of our scheme against the untrusted CSP, the undesignated verifier, and the semi-trusted verifier.

Theorem 1: If the CDH problem is held in G_1, then our scheme can resist untrusted CSP in the random oracle.

Proof: Suppose an adversary $\mathcal{A}$ is a PPT algorithm attacking our scheme. The challenger $\mathcal{C}$ wants to solve the CDH problem, which is to calculate the value of g^{ab} given a random instance $(g, g^a, g^b) \in G_1$ where a and b are unknown. In the security game $Game1$, $\mathcal{A}$ and $\mathcal{C}$ interact in order to complete each other's task separately. $\mathcal{C}$ simulates the environment for $\mathcal{A}$ and relies on $\mathcal{A}$ as a subroutine to obtain the answer to solve the CDH problem after $\mathcal{A}$ breaches the scheme.

1) **Setup**: $\mathcal{C}$ generates the public parameters *params*, and sets $P_0 = g^a$, where a is the master key and is unknown. $\mathcal{C}$ sends *params* and P_0 to $\mathcal{A}$.
2) **Queries**: $\mathcal{A}$ adaptively executes polynomial queries to the $\mathcal{C}$.
   a) *Hash queries*: $\mathcal{A}$ adaptive executes Hash queries with any identity ID_i. $\mathcal{C}$ maintains a list $L = \{ID, H_1(ID)\}$. When $\mathcal{A}$ asks for the hash value

of the ID_i, $\mathcal{C}$ first finds whether $(ID_i, H_1(ID_i))$ in L. If it exists, $\mathcal{C}$ return $H_1(ID_i))$ to $\mathcal{A}$. Otherwise, $\mathcal{C}$ randomly selects $\theta \in Z_p^*$ and sets $H_1(ID_i) = g^\theta$, then $\mathcal{C}$ returns $H_1(ID_i)$ to $\mathcal{A}$ and stores $(ID_i, H_1(ID_i))$ in L. Note that $\mathcal{C}$ defines $H_1(ID_O)$ as g^b, Where b is a specific value of θ, but b is unknown.

b) *Extract queries*: $\mathcal{A}$ adaptive executes Extract queries with any identity ID_i. $\mathcal{C}$ computes

$$sk_{ID_i} = H_1(ID_i)^a = (g^\theta)^a = g^{a\theta} = P_0^\theta.$$

And $\mathcal{A}$ cannot directly ask for the ID_O of the private key, $\mathcal{C}$ implicitly sets $sk_O = (g^b)^a = g^{ab}$.

c) *Tag queries*: $\mathcal{A}$ executes Tag queries of any data block in file F of any identity ID_i. $\mathcal{C}$ first randomly selects Fid as the filename of F and splits $F = \{m_{i,j}\}_{\substack{1\le i\le n\\ 1\le j\le s}}$. Then $\mathcal{C}$ computes α by sk_O and $H_1(ID_V)$.

If $ID_i \neq ID_O$, $\mathcal{C}$ computes

$$T_i = sk_O \cdot (H_2(\alpha \| Fid \| H(m_i)) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}})^t.$$

If $ID_i = ID_O$, for each j, $1 \le j \le s$, $\mathcal{C}$ randomly chooses $\beta_j, \gamma_j, r \in Z_q^*$, and sets

$$u_j = g^{\beta_j} \cdot (g^b)^{\gamma_j},\ \chi = (g^a)^r.$$

Thus, $t = ar$ and

$$\prod_{j=1}^{s} u_j^{m_{i,j}} = \prod_{j=1}^{s} [g^{\beta_j} \cdot (g^b)^{\gamma_j}]^{m_{i,j}} = (g)^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot (g^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}}.$$

For each i, $1 \le i \le n$, $\mathcal{C}$ randomly chooses $\lambda_i \in Z_p$, and programs the random oracle at $\alpha \| Fid \| H(m_i)$ as

$$H_2(\alpha \| Fid \| H(m_i)) = (g^b)^{\alpha_i} \cdot g^{\lambda_i} / g^{\sum_{j=1}^{s} \beta_j m_{i,j}},$$

where $\alpha_i = (r \sum_{j=1}^{s} \beta_j m_{i,j} - 1)/r$. Now, $\mathcal{C}$ can compute T_i.

$$\begin{aligned} T_i &= sk_O \cdot (H_2(\alpha \| Fid \| H(m_i)) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}})^t \\ &= g^{ab} \cdot ((g^b)^{\alpha_i} \cdot g^{\lambda_i} / g^{\sum_{j=1}^{s} \beta_j m_{i,j}} \cdot g^{\sum_{j=1}^{s} \beta_j m_{i,j}} \\ &\quad \cdot (g^b)^{\sum_{j=1}^{s} \gamma_j m_{i,j}})^{ar} \\ &= g^{ab} \cdot (g^{ab})^{r\alpha_i} \cdot (g^{ab})^{r \sum_{j=1}^{s} \gamma_j m_{i,j}} \cdot (g^a)^{r\lambda_i} \\ &= (g^{ab})^{(1 + r\alpha_i + r \sum_{j=1}^{s} \gamma_j m_{i,j})} \cdot (g^a)^{r\lambda_i} \\ &= (g^a)^{r\lambda_i}. \end{aligned}$$

Finally, $\mathcal{C}$ uses an IDS algorithm to sign

$$\tau_0 = u_1 \| u_2 \| \cdots \| u_s \| \chi \| Fid$$

and generates $\tau = \tau_0 \| IDS(\tau_0)$. And $\mathcal{C}$ returns

$$T = (F, \{u_j\}_{1\le j\le l}, Fid, \tau, \chi, \{T_i\}_{1<i<n})$$

to $\mathcal{A}$.

3) **ProofCheck**: $\mathcal{C}$ generates the challenge message $chal = \{l, k_1, k_2\}$ and sends it to $\mathcal{A}$. After receiving $chal$, $\mathcal{A}$ generates proof Γ. $\mathcal{C}$ verifies Γ to get 0 or 1.

4) **Forgery**: $\mathcal{C}$ sends a new challenge message $chal^* = (l^*, k_1^*, k_2^*)$ to $\mathcal{A}$. The $proofGen(chal^*, T)$ algorithm generates a corresponding proof based on $chal$ is

$$\Gamma = (\{u_j, W_j\}_{1\le j\le s}, Fid, \tau, \sigma, \{M_j\}_{1\le j\le s}, \Lambda, sig(H(R))),$$

which $\Lambda = \{\{H(m_i), \Omega_i\}_{c_1\le i\le c_l}\}$. The correctness of $H(m_i)$ can be verified through Λ and $sig(H(R))$. $\mathcal{A}$ generates a forged proof Γ^* based on $chal^*$. Because of the authentication in MHT, the last two parts of Γ^* should be the same as Λ, $sig(H(R))$ in Γ. Suppose

$$\Gamma^* = (\{u_j, W_j\}_{1\le j\le s}, Fid, \tau, \sigma^*, \{M_j^*\}_{1\le j\le s}, \Lambda, sig(H(R))).$$

Hence, $(\sigma^*, \{M_j^*\}_{1\le j\le s}) \neq (\sigma, \{M_j\}_{1\le j\le s})$.

If both Γ^* and Γ can pass the verify algorithm, then there must be

$$\begin{aligned} &e(\sigma^*, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ &= e(H_1(ID_O)^{\sum_{(a_i,c_i)\in S} a_i}, P_0) \\ &\quad \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \cdot \prod_{j=1}^{s} u_j^{M_j^*}, \chi). \quad (2) \\ &e(\sigma, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ &= e(H_1(ID_O)^{\sum_{(a_i,c_i)\in S} a_i}, P_0) \\ &\quad \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \cdot \prod_{j=1}^{s} u_j^{M_j}, \chi). \quad (3) \end{aligned}$$

Use equation (2) divided by equation (3),

$$\begin{aligned} e(\sigma^*/\sigma, g) &= e(\prod_{j=1}^{s} u_j^{M_j^* - M_j}, \chi) \\ &= e(g^{\sum_{j=1}^{s} \beta_j (M_j^* - M_j)} \\ &\quad \cdot (g^b)^{\sum_{j=1}^{s} \gamma_j (M_j^* - M_j)}, \chi). \end{aligned}$$

Reduction to

$$\begin{aligned} &e(\sigma^* \cdot \sigma^{-1} \cdot P_0^{-\sum_{j=1}^{s} r\beta_j (M_j^* - M_j)}, g) \\ &\quad = e((g^{ab})^{\sum_{j=1}^{s} r \cdot \gamma_j (M_j^* - M_j)}, g). \end{aligned}$$

From the above formula, we see that we found the solution to the CDH problem,

$$g^{ab} = (\sigma^* \cdot \sigma^{-1} \cdot P_0^{-\sum_{j=1}^{s} r\beta_j (M_j^* - M_j)})^{\frac{1}{\sum_{j=1}^{s} r \cdot \gamma_j (M_j^* - M_j)}},$$

unless evaluating the exponent causes a divide-by-zero. However, at least one of $(M_j^* - M_j)$ is not 0, and r and γ_j are all randomly selected and theoretically hidden from $\mathcal{A}$. So,

$$\sum_{j=1}^{s} r \cdot \gamma_j (M_j^* - M_j) = 0$$

only with probability $1/q$,it is negligible.

Therefore, Theorem 1 is proved, Which shows that our scheme can resist untrusted CSP in the random oracle.

Theorem 2: If the DL problem is held in G_1, this scheme is effective for the undesignated verifier.

Proof: Suppose an adversary $\mathcal{A}'$ is a PPT algorithm in the security game *Game*2. The challenger $\mathcal{C}'$ wants to get a solution to the DL problem from the process of $\mathcal{A}'$ breaking the scheme. The query-response interaction process between $\mathcal{A}'$ and $\mathcal{C}'$ is similar to that in Theorem 1. The differences are as follows:

$\mathcal{C}'$ defines $H_1(ID_V) = g^c$, $sk_O = (g^b)^a = g^{ab}$ and

$$\begin{aligned}\alpha &= e(sk_O, g) \cdot e(H_1(ID_V), P_0) \\ &= e(g^{ab}, g) \cdot e(g^c, g^a) = e(g^{ab+ac}, g).\end{aligned}$$

Then $\mathcal{A}'$ performs an adaptive polynomial query on $\mathcal{C}'$, $\mathcal{C}'$ responds $H_1(ID_V)$ and $H_1(ID_O)$ to $\mathcal{A}'$. Note that sk_O is secret and unresponsive. Eventually $\mathcal{A}'$ forges α^* through known information. The correct α meets the equation

$$\begin{aligned}\alpha &= e(sk_V, g) \cdot e(H_1(ID_O), P_0) \\ &= e(g^{ac}, g) \cdot e(g^b, g^a) = e(g^{ab+ac}, g).\end{aligned}$$

If $\alpha = \alpha^*$, then $\mathcal{A}'$ wins the game, $\mathcal{C}'$ can also obtain the answer to the DL problems. Depending on the difficulty of the DL problem, the advantage for the PPT algorithm $\mathcal{A}'$ to compute $ab + ac$ in G_1 is negligible, we can know that our scheme is valid and can prevent the undesignated verifier from checking outsourced data. Theorem 2 is proved.

Theorem 3: If the variant of the DL problem and the one-way nature of cryptographic hash function is held in G_1, this scheme is effective in terms of resisting the semi-trusted verifier in the random oracle.

Proof: Suppose an adversary $\mathcal{A}''$ is a PPT algorithm in the security game *Game*3, and it wants to steal the relevant content of the data block in the following two ways during the integrity checking phase. The challenger $\mathcal{C}''$ wants to solve the variant of DL problem and crack the one-way of cryptographic hash function. Formally expressed as given a random instance $(g_1, g_1^x) \in G_1^2$ of the variant of DL problem, $\mathcal{C}''$ tries to calculate the value of x by relying on $\mathcal{A}''$. And for a given hash value h, $\mathcal{C}''$ wants to find m such that $H(m) = h$. $\mathcal{C}''$ simulates the environment for $\mathcal{A}''$ and uses $\mathcal{A}''$ as a subroutine to obtain its answers.

1) **Setup and Queries**: The operations of these two phases are performed by $\mathcal{C}''$ are similar to those performed by $\mathcal{C}$. However, $\mathcal{A}''$ cannot perform tag queries on $\mathcal{C}''$ because the verifier does not know the tag contents in the real situation.
Except that in the Tag Query phase, $\mathcal{C}''$ also randomly choose $\{d_j\}_{1 \le j \le s} \in Z_q^*$, and sets

$$u_1 = g^{d_1}, u_2 = g^{d_2}, \cdots, u_s = g^{d_s}.$$

2) **Forgery**: $\mathcal{A}''$ sends challenge message to $\mathcal{C}''$, and $\mathcal{C}''$ simulates the CSP to generate proof Γ and returns it to $\mathcal{A}''$. $\mathcal{C}''$ return the correct proof

$$\Gamma = (\{u_j, W_j\}_{1 \le j \le s}, Fid, \tau, \sigma, \{M_j\}_{1 \le j \le s}, \Lambda, sig(H(R)))$$

and it meets the

$$\begin{aligned}&e(\sigma, g) \cdot e(\prod_{j=1}^{s} W_j, \chi) \\ &= e(H_1(ID_O)^{\sum\limits_{(a_i,c_i)\in S} a_i}, P_0) \\ &\quad \cdot e(\prod_{(a_i,c_i)\in S} H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \cdot \prod_{j=1}^{s} u_j^{M_j}, \chi). \quad (4)\end{aligned}$$

If $\mathcal{A}''$ wants to obtain the data block m_{c_i}, there are two ways.

1) $\mathcal{A}''$ wants to use $H(m_{c_i})$ to get m_{c_i}. The cryptographic hash functions are one-way. It is easy to calculate its hash value from a pre-mapped value. However, it is computationally negligible to generate a pre-mapped value whose hash value is equal to a special value. When $\mathcal{A}''$ gets $H(m_{c_i})$, it outputs m'_{c_i} as a guess for m_{c_i}. If $m'_{c_i} = m_{c_i}$, $\mathcal{A}''$ breaks our scheme, then $\mathcal{C}''$ can rely on $\mathcal{A}''$ to obtain an answer that solves the one-way nature of the cryptographic hash function. But the one-way nature of cryptographic hash functions means that solving for m_{c_i} is computationally negligible. The two are contradictory. So $\mathcal{A}''$ cannot obtain the data block content from the cryptographic hash function value.
2) $\mathcal{A}''$ wants to obtain $m_{c_i,j}$ through a series of linear combinations of $M_j = \sum_{i=1}^{l} a_i \cdot m_{c_i,j} + x^*$. If $\mathcal{A}''$ gets $m_{c_i,j}$, it can generate $M_j^* = \sum_{i=1}^{l} a_i \cdot m_{c_i,j}$ directly. If M_j^* is correct, then there must be

$$\begin{aligned}e(\sigma, g) = e(H_1(ID_O)^{\sum\limits_{(a_i,c_i)\in S} a_i}, P_0) \\ \cdot e(\prod_{(a_i,c_i)\in S} (H_2(\alpha \| Fid \| H(m_{c_i}))^{a_i} \\ \cdot \prod_{j=1}^{s} u_j^{M_j^*}, \chi). \quad (5)\end{aligned}$$

So if we apply equations (4) and (5), we get

$$e(\prod_{j=1}^{s} W_j, \chi) = e(\prod_{j=1}^{s} u_j^{M_j^* - M_j}, \chi)$$

and can further imply that

$$\prod_{j=1}^{s} W_j = (g^{\sum_{j=1}^{s} d_j})^{x^*}.$$

Therefore, If $\mathcal{A}''$ can calculate $\prod_{j=1}^{s} W_j$, then $\mathcal{C}''$ can solve the a variant of DL problem. Because a variant of DL problems cannot be solved under a non-negligible probability, $\mathcal{A}''$ cannot calculate x^*.

Based on the one-way nature of the cryptographic hash function and the difficulty of a variant of DL problems, we know that our scheme is effective and prevents the verifier from obtaining the data content of the data owner. Therefore, Theorem 3 is proved.

IV. PERFORMANCES ANALYSIS

In this section, we compare the computational cost and communication cost of our scheme with other schemes. Finally, we evaluate the performance of this scheme through experiments.

In related work, we use Table 1 to present a functionality comparison between our scheme and existing schemes. For performance comparison, we should choose schemes that as more than one same function as our scheme. Therefore, we initially choose the schemes in [22], [23], [25], [27], [29] and [30]. However, schemes in [25] and [27] have been proved to have design flaws, which are detailed in the relevant work section, while the scheme in [29] uses the verifier to sign the encrypted data, which reduces security and flexibility. Therefore, we compare our scheme with two identity-based RDIC schemes with privacy protection and a PKI-based PDP scheme with privacy protection and dynamic updating, which were designed in [22], [23] and [30], respectively.

A. COMPUTATIONAL COST

For simplicity, we define the following notations to denote the operations in our scheme. Let T_{exp}, T_{mul} and T_P denote the computational cost of one exponentiation operation, one multiplication operation and one pairing operation on group G_1, respectively. n is the total number of data blocks. l is the number of challenged data blocks. T_{IDS} and T_{ver} represent the computational cost of the IDS algorithm to generate and verify signatures, respectively. T_{ver} and T'_{ver} represent the computational cost of generating and verifying signatures in the PKI-based signature algorithm such as BLS signature, respectively.

In our scheme, a data block is divided into multiple sectors, effectively reducing the storage cost [10]. However, the [22], [23] and [30] schemes only divide the file F into n blocks and do not go on to divide the blocks into s sectors. To make a fair comparison, we set $s = 1$. Computationally expensive operations, such as pairing and exponentiation, mainly determine our scheme's computational cost.

Computationally expensive operations mainly determine our scheme's computational cost, such as pairing and exponentiation. While other operations such as hash function and addition have negligible costs, they are ignored in subsequent analyses. We ignore their computational costs since the *Setup*, *Extract*, and *Challenge* algorithm has little impact on scheme performance. In the *TagGen* and *Verify* algorithm, the data owner and the designated verifier need to calculate α. However, α only needs to be calculated once in advance and stored locally. Its computational cost and required storage space are are minimal. Therefore, in the following discussion, the computational cost of α is not considered.

The *TagGen* algorithm needs to compute χ, n times T_i and two IDS signatures τ and $sig(H(R))$. When we set $s = 1$, the computational cost is

$$(2n + 1) \cdot T_{exp} + 2n \cdot T_{mul} + 2 \cdot T_{IDS}.$$

The *ProofGen* algorithm should compute σ, s times W_j and M_j. Its computational cost is

$$(l + 1) \cdot T_{exp} + (l - 1) \cdot T_{mul}.$$

In the *Verify* algorithm, we need to validate τ and $sig(H(R))$, and then check equation 1. Its computational cost is

$$4 \cdot T_p + (l + 2) \cdot T_{exp} + (l + 2) \cdot T_{mul} + 2 \cdot T_{ver}.$$

In Table 2, we give the computational cost comparison between our scheme and the schemes in [22], [23] and [30]. From Table 2, we can get that in the *TagGen* algorithm, our scheme has two IDS signature operations, while other schemes have only one IDS signature or PKI-based signature operation. Furthermore, our scheme has n more multiplications than the scheme in [22], and one more exponentiation and n multiplications than the scheme in [30]. In the *ProofGen* algorithm, our scheme is the same as in the scheme in [23], reducing two pairing operations compared with the scheme in [22] and one pairing operation compared with the scheme in [30]. In the *Verify* algorithm, our scheme requires four pairing operations and two verification operations of the IDS algorithm, the scheme in [23] requires three pairing operations and one verification operation of the PKI-based signature algorithm. The scheme in [30] requires two pairing schemes and one verification operation of the PKI-based signature algorithm. However, the pairing operation of the scheme in [22] is linear with the number of challenge blocks, and this scheme requires more pairing operations and one verification operation of the IDS algorithm.

B. COMMUNICATION COST

The communication between the data owner, the CSP, and the designated verifier is mainly in three phases: data owner transfers T to the CSP, the designated verifier sends *chal* to CSP, and CSP returns Γ to the designated verifier, namely "Initialization", "Challenge" and "Response". So next, we mainly discuss the communication cost of these three phases.

TABLE 2. Comparison of computational cost.

Schemes	TagGen	ProofGen	Verify
Scheme in [22]	$(2n+1)\cdot T_{exp}+n\cdot T_{mul}+T_{IDS}$	$2\cdot T_P+(l+1)\cdot T_{exp}+l\cdot T_{mul}$	$l\cdot T_P+(l+1)\cdot T_{exp}+(l-1)\cdot T_{mul}+T_{ver}$
Scheme in [23]	$(2n+1)\cdot T_{exp}+2n\cdot T_{mul}+T_S$	$(l+1)\cdot T_{exp}+(l-1)\cdot T_{mul}$	$3\cdot T_P+(l+3)\cdot T_{exp}+(l+2)\cdot T_{mul}+T_{ver}^{'}$
Scheme in [30]	$2n\cdot T_{exp}+n\cdot T_{mul}+T_S$	$T_P+(l+1)\cdot T_{exp}+l\cdot T_{mul}$	$2\cdot T_P+(l+3)\cdot T_{exp}+(l+1)\cdot T_{mul}+T_{ver}^{'}$
Our scheme	$(2n+1)\cdot T_{exp}+2n\cdot T_{mul}+2\cdot T_{IDS}$	$(l+1)\cdot T_{exp}+(l-1)\cdot T_{mul}$	$4\cdot T_P+(l+2)\cdot T_{exp}+(l+2)\cdot T_{mul}+2\cdot T_{ver}$

TABLE 3. Comparison of communication cost.

Schemes	Initialization	Challenge	Response
Scheme in [22]	$(n+1)\cdot\|G_1\|+\|\tau\|$	$\|G_1\|+\|G_2\|+l\cdot(\|n\|+\|Z_q^*\|)+\|pf\|$	$\|G_1\|+\|G_2\|+\|\tau\|$
Scheme in [23]	$(n+2)\cdot\|G_1\|+\|\tau\|$	$\|n\|+2\cdot\|Z_q^*\|$	$4\cdot\|G_1\|+\|\tau\|+\|Z_q^*\|$
Scheme in [30]	$n\cdot\|G_1\|+\|\tau\|$	$l\cdot\|n\|+l\cdot\|Z_q^*\|$	$(l+1)\cdot\|G_1\|+\|G_T\|+\|\tau\|+\|\Lambda\|$
Our scheme	$(n+2)\cdot\|G_1\|+\|\tau\|+\|sig(H(R))\|$	$\|n\|+2\cdot\|Z_q^*\|$	$4\cdot\|G_1\|+\|Z_q^*\|+\|\tau\|+\|\Lambda\|+\|sig(H(R))\|$

For the sake of brevity of theoretical analysis, we use $|n|$ to denote the size of the elements in the set $[1, n]$ $|G_1|$ and $|Z_q^*|$ to denote the size of an element in $|G_1|$ and Z_q^*, respectively.

In the "Initialization" phase, the data owner uploads T to the CSP. When we set $s = 1$, the communication cost of this phase is

$$(n+2)\cdot|G_1|+|\tau|+|sig(H(R))|.$$

In the "Challenge" phase, the designated verifier sends $chal = \{l, k_1, k_2\}$ to the CSP. Its communication cost is

$$|n|+2\cdot|Z_q^*|.$$

In the "Response" phase, the CSP returns Γ to the designated verifier. The communication cost required by the CSP to transmit Γ is

$$4\cdot|G_1|+|Z_q^*|+|\tau|+|\Lambda|+|sig(H(R))|.$$

Furthermore, we compared the communication costs of our scheme with schemes in [22], [23] and [30], as shown in Table 3.

C. EXPERIMENT RESULTS

In this section, we conduct experiments on the proposed scheme to evaluate its performance. We implemented our scheme based on the "Charm" framework. The experiment was carried out on the VMware Workstation 10 with the configured of 4-core CPU, 4G memory, and 40G disk. The workstation runs on the host computer of laptop-Q9GS1UTR with the Win10 operating system, 8G Ram, and an AMD Ryzen 7 PRO 4750U with a Radeon Graphics 1.70ghz processor. The operation system is Ubuntu-20.04. For a full comparison, we also implemented the schemes in [22], [23] and [30] in the same library and environment.

We first evaluate the computational cost of generating tags for files in our scheme and the schemes in [22], [23] and [30]. First, we create a file with a size of 20M. In this experiment, the number of data blocks the file is set to 200, 400, 600, 800, 1000. Since these schemes all require an IDS algorithm or PKI-based signature algorithm, we choose the IDS algorithm in [31] or the BLS algorithm in [32], respectively,

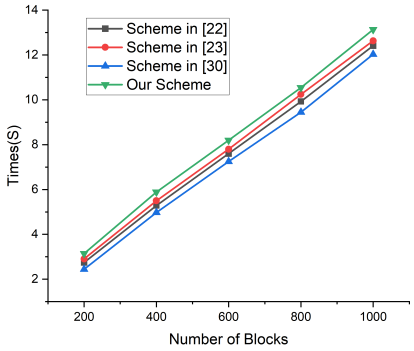

FIGURE 11. Computation cost of tag generation.

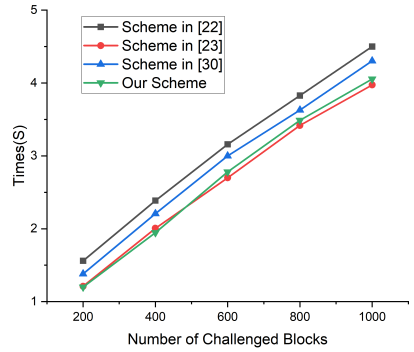

FIGURE 12. Computational cost of proof generation.

to implement the signature operation. The results are shown in Figure 11. The computation cost of tag generation is high, and the time spent increases almost linearly with the number of data blocks. Moreover, our scheme takes more time to generate tags than the schemes in [22], [23] and [30].

We compare the computational cost of generating data integrity proofs between our scheme and the schemes in [22], [23] and [30]. We increase the counter of challenged blocks from 200 to 1000 with an increment of 200 in each experiment. The result is shown in Figure. 12. Our scheme and the scheme in [23] have almost the same computational cost and better performance than the schemes in [22] and [30]. Finally,

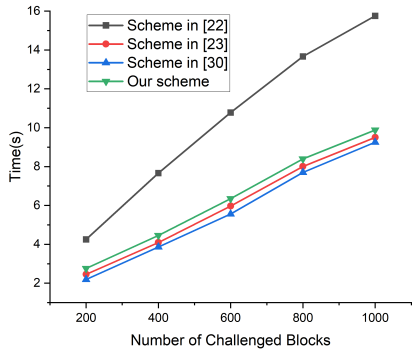

FIGURE 13. Computational cost for verification.

as the number of challenge blocks increases, we evaluate the performance of the three schemes for verifying the data integrity proof. The results are shown in Figure 13, and the time cost of all three schemes increases with the number of challenge blocks. Moreover, our scheme takes slightly more time than schemes in [23] and [30], and lower than scheme in [22].

According to the experimental results, compared with other schemes, the cost of our scheme is slightly higher. However, our scheme achieves dynamic data operations and allows only the designated verifier to perform verification tasks, better meeting the data owner's needs. Moreover, it can ensure that untrusted CSP cannot forge and delete data content, and semi-trusted verifiers cannot obtain the data owner's data content. Therefore, our scheme is effective and feasible for real application.

V. CONCLUSION

This paper proposes an identity-based remote data integrity checking scheme with a designated verifier. This scheme can also solve the semi-trusted verifier and realize data privacy protection. Meanwhile, our scheme uses MHT to support dynamic operations such as data insertion, modification and deletion. Furthermore, based on the DL assumption and the CDH assumption, we prove the scheme's security. Finally, the experimental analysis proves that our scheme is effective and more suitable for real-life application scenarios.

ACKNOWLEDGMENT

The authors thank the anonymous reviewers for their careful work and thoughtful suggestions that have helped improve this paper substantially.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Commun. ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] D. Zissis and D. Lekkas, "Addressing cloud computing security issues," *Future Gener. Comput. Syst.*, vol. 28, no. 3, pp. 583–592, Mar. 2012.

[3] J. Lu, F. Nan, Y. Huang, C.-C. Chang, Y. Du, and H. Tian, "Privacy-preserving public auditing for secure data storage in fog-to-cloud computing," *J. Netw. Comput. Appl.*, vol. 127, pp. 59–69, Dec. 2018.

[4] Y. Deswarte, J.-J. Quisquater, and A. Saidane, "Remote integrity checking," in *Proc. Work. Conf. Integrity Internal Control Inf. Syst.*, Cham, Switzerland: Springer, 2003, pp. 1–11.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur. (CCS)*, 2007, pp. 598–609.

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

[7] Y. Feng, Y. Mu, G. Yang, and J. K. Liu, "A new public remote integrity checking scheme with user privacy," in *Proc. Australas. Conf. Inf. Secur. Privacy*. Berlin, Germany, Springer, 2015, pp. 377–394.

[8] H. Yan, J. Li, and Y. Zhang, "Remote data checking with a designated verifier in cloud storage," *IEEE Syst. J.*, vol. 14, no. 2, pp. 1788–1797, Jun. 2020.

[9] 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.

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

[11] Y. Ren, J. Xu, J. Wang, and J.-U. Kim, "Designated-verifier provable data possession in public cloud storage," *Int. J. Secur. Appl.*, vol. 7, no. 6, pp. 11–20, Nov. 2013.

[12] S.-T. Shen and W.-G. Tzeng, "Delegable provable data possession for remote data in the clouds," in *Proc. Int. Conf. Inf. Commun. Secur.*, Berlin, Germany, Springer, 2011, pp. 93–111.

[13] H. Wang, "Proxy provable data possession in public clouds," *IEEE Trans. Services Comput.*, vol. 6, no. 4, pp. 551–559, Oct./Dec. 2013.

[14] H. Wang, "Identity-based distributed provable data possession in multi-cloud storage," *IEEE Trans. Services Comput.*, vol. 8, no. 2, pp. 328–340, Mar./Apr. 2015.

[15] J. Chang, H. Wang, F. Wang, A. Zhang, and Y. Ji, "RKA security for identity-based signature scheme," *IEEE Access*, vol. 8, pp. 17833–17841, 2020.

[16] Y. Chen and J.Chang, "Identity-based proof of retrievability meets with identity-based network coding," *Cluster Comput.*, early access, 2022, doi: 10.1007/s10586-022-03545-y.

[17] J. Zhao, C. Xu, F. Li, and W. Zhang, "Identity-based public verification with privacy-preserving for data storage security in cloud computing," *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 96, no. 12, pp. 2709–2716, 2013.

[18] Y. Ji, B. Shao, J. Chang, and G. Bian, "Privacy-preserving certificateless provable data possession scheme for big data storage on cloud, revisited," *Appl. Math. Comput.*, vol. 386, Dec. 2020, Art. no. 125478.

[19] X. Yang, M. Wang, T. Li, R. Liu, and C. Wang, "Privacy-preserving cloud auditing for multiple users scheme with authorization and traceability," *IEEE Access*, vol. 8, pp. 130866–130877, 2020.

[20] 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.

[21] J. Zhang and Q. Dong, "Efficient ID-based public auditing for the outsourced data in cloud storage," *Inf. Sci.*, vols. 343–344, pp. 1–14, May 2016.

[22] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Security*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

[23] J. Li, H. Yan, and Y. Zhang, "Identity-based privacy preserving remote data integrity checking for cloud storage," *IEEE Syst. J.*, vol. 15, no. 1, pp. 577–585, Mar. 2021.

[24] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur.*, Berlin, Germany, Springer, 2009, pp. 355–370.

[25] T. Shang, F. Zhang, X. Chen, J. Liu, and X. Lu, "Identity-based dynamic data auditing for big data storage," *IEEE Trans. Big Data*, vol. 7, no. 6, pp. 913–921, Dec. 2021.

[26] X. Li, S. Liu, R. Lu, and X. Zhang, "On security of an identity-based dynamic data auditing protocol for big data storage," *IEEE Trans. Big Data*, vol. 7, no. 6, pp. 975–977, Dec. 2021.

[27] Y. Sun, Q. Liu, X. Chen, and X. Du, "An adaptive authenticated data structure with privacy-preserving for big data stream in cloud," *IEEE Trans. Inf. Forensics Security*, vol. 15, pp. 3295–3310, 2020.

[28] S. Li, Y. Zhang, C. Xu, and K. Chen, "Cryptoanalysis of an authenticated data structure scheme with public privacy-preserving auditing," *IEEE Trans. Inf. Forensics Security*, vol. 16, pp. 2564–2565, 2021.
[29] J. Liu, X. A. Wang, Z. Liu, H. Wang, and X. Yang, "Privacy-preserving public cloud audit scheme supporting dynamic data for unmanned aerial vehicles," *IEEE Access*, vol. 8, pp. 79428–79439, 2020.
[30] Y. Wu, Z. L. Jiang, X. Wang, S. M. Yiu, and P. Zhang, "Dynamic data operations with deduplication in privacy-preserving public auditing for secure cloud storage," in *Proc. 7 IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 562–567.
[31] F. Hess, "Efficient identity based signature schemes based on pairings," in *Proc. Int. Workshop Sel. Areas Cryptogr.*, Berlin, Germany, Springer, 2002, pp. 310–324.
[32] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, Berlin, Germany, Springer, 2001, pp. 514–532.

GENQING BIAN received the Ph.D. degree from the School of Management, Xi'an University of Architecture and Technology (XAUAT), Shaanxi, China. He is currently a Professor with XAUAT. His research interests include information security, cloud computing security, and data analysis. He is a member of the China Computer Federation (CCF) and the Association for Computing Machinery (ACM).

RUI ZHANG is currently pursuing the M.S. degree with the School of Information and Control Engineering, Xi'an University of Architecture and Technology, Shaanxi, China. Her research interests include cloud computing security and privacy protection.

BILIN SHAO received the M.S. degree from the School of Management, XAUAT, Shaanxi, China. He is currently a Professor with XAUAT. His research interests include information security, information management technology, cloud computing security, and VANETS security. He is a member of CCF.