Computers & Security 124 (2023) 103002

Contents lists available at ScienceDirect

Computers & Security

journal homepage: www.elsevier.com/locate/cose

DIVRS: Data integrity verification based on ring signature in cloud storage

Miao Tian[a], Yushu Zhang[a,*], Youwen Zhu[a], Liangmin Wang[b], Yong Xiang[c]

[a] College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 211106, China
[b] School of Cyber Science and Engineering, Southeast University, Nanjing, 211100, China
[c] School of Information Technology, Deakin University, Victoria 3125, Australia

ARTICLE INFO

Article history:
Received 14 March 2022
Revised 5 September 2022
Accepted 31 October 2022
Available online 2 November 2022

Keywords:
Cloud storage
Data audit
Lattice-based cryptography
Ring signature
Post-quantum security

ABSTRACT

Nowadays, cloud service is becoming an indispensable part of people's lives and a growing number of users store data on servers. Although this way can reduce storage and maintenance costs, data owners lose the control and possession of the data, which will result in some additional security issues, such as data corruption. It means that cloud server providers may delete or tamper with the infrequently used data to economize on storage space. Therefore, it is essential to provide an audit service to verify integrity of data stored on the cloud. Most of the existing audit schemes are based on the difficult problems of large integer decomposition and discrete logarithms. With the development of quantum computers, these schemes will face the security threat, because quantum computers can easily solve these difficult problems. In short, it is essential to design a data audit scheme that is resistant to quantum attacks. Therefore, we design a novel data audit scheme to achieve post-quantum security, in which the ring signature based on the problem of learning with errors is employed. In addition, two data possession games are designed to prove the security of the DIVRS. Experiment results show that the communication and computing costs are acceptable, and the time spent in each phase of the DIVRS is shorter than that of the comparison schemes.

1. Introduction

Data owners prefer to store large amounts of data on cloud platforms that provide both storage and computing services, e.g., Apple iCloud, Baidu Cloud, Tencent Cloud, and Ali Cloud, etc. In this way, on the one hand, data owners can make full use of the abundant software, hardware, and bandwidth resources of the cloud, and access their data anywhere via the internet; on the other hand, the cloud can obtain economic benefits by collecting rents from data owners. Another benefit for data owners comes from the fact that a cloud platform has high availability and low cost compared with complex local storage management. Although data owners can gain great benefits from storing data in clouds, this way also brings serious security problems, i.e., cloud server providers (CSP) may delete or tamper with the infrequently used data to economize on storage space. In addition, once the data are outsourced to the cloud, the data owner cannot exercise physical control.

As a way to verify data integrity, provable data possession (PDP) (Ateniese et al., 2007) allows users to label each block of data and select part of the block for validation based on the challenge information. However, the data cannot be recovered if they are corrupted. Therefore, proof of retrievability (POR) model (Juels and Kaliski, 2007) was proposed to solve this problem, which not only ensures that the server stores the data correctly through sampling and error-correcting techniques, but also enables users to retrieve the data they need. However, some schemes (Ateniese et al., 2007; Shacham and Waters, 2008; Tian et al., 2021; Xu et al., 2021) only consider static data storage, which is not suitable for general scenarios. To support data updates, including modification, insertion and deletion, some schemes based on classic Merkle hash tree and bilinear aggregate signature were designed. There is a computational burden on data owners to verify data integrity themselves, thus researchers put forward a great deal of schemes (Jin et al., 2016; Li et al., 2016; Liu et al., 2015; Shen et al., 2017; Shi et al., 2013; Wang et al., 2013a; 2013b) to achieve public validation and batch processing. Afterwards, the identity-based PDP schemes (Li et al., 2022; 2017; Shen et al., 2019; Yu et al., 2016b) were presented, which support multiple complex storage environments. However, those schemes bring heavy computational cost and need large system public parameters.

* Corresponding author.
E-mail address: yushu@nuaa.edu.cn (Y. Zhang).

https://doi.org/10.1016/j.cose.2022.103002
0167-4048/

The majority of integrity verification schemes utilize conventional asymmetric cryptographic primitives, and difficult problems used to construct these schemes comes from large integer decomposition or discrete logarithm. However, with the development of quantum computers, traditional cryptography may no longer be secure, which will threaten the integrity verification schemes. Lattice-based cryptography (Micciancio and Regev, 2006) is a mature research and application field. The potential resistance of lattice cryptography to attack from quantum computer provides a new impetus for the research and implementation of related algorithm protocols. Famous lattice-based cryptography designs include Ajtai-Dwork lattice cryptosystem (Ajtai and Dwork, 1997), Goldreich-Goldwasser-Halevi public key cryptography and signature system (Goldreich et al., 1997), NTRU system (Hoffstein et al., 1998), cryptosystem based on SIS (Ajtai, 1996) and LWE (Regev, 2009), and fully-homomorphic-encryption system (Gentry, 2009), etc. The cryptography based on lattice maintains security under the quantum environment, which provides the great promise for cryptography of post-quantum, as it realizes high efficiency and strong security. To sum up, it is necessary to design a data audit scheme utilizing lattice cryptography in the quantum computing environment.

In this paper, we propose an efficient integrity verification scheme based on ring signature constructed from a lattice cipher, which is secure under the quantum environment. And the contributions of this work are elaborated as follows.

- A DIVRS scheme is proposed to realize data audit against quantum attack, which effectively solves the security problems of cloud data storage. The DIVRS scheme not only verifies whether the CSP deletes or tampers with the data of owner, but also guarantees the security of verification in quantum environment.
- The security of the DIVRS scheme is analyzed in terms of correctness of verification, unforgeability of tag, and anonymity. In view of the characteristics of anti-quantum attack and the adopted cryptology mechanism of lattices, the DIVRS scheme enables fair interacting between owners and the server, as well as deals with the security problem in quantum environment and achieves fast and efficient verification.
- We evaluate the performance of the DIVRS scheme on six randomly generated files, which not only illustrates the smaller communication and computational costs, but also indicates that the proposed scheme is faster than that of previous schemes in stage of TagGen, ProGen, and Verify. Meanwhile, the experimental results show that the DIVRS scheme is feasible and efficient.

The rest of this paper is organized as follows. In Section 2, We review the related work, and preliminary background is introduced in Section 3. In addition, the system and threat models are described in Section 4. In Section 5, we describe the proposed data integrity verification scheme followed by the security analysis. The theoretical and actual performance of this scheme are evaluated in Section 7 and the scheme is concluded in the last section.

2. Related work

2.1. Provable data possession

Ateniese et al. (2007) first introduced the model of PDP, which allows the data owner to verify the data integrity based on RSA algorithm. However, this model did not consider dynamic operations and rigorous security proof. Afterwards, they designed the provable secure PDP scheme using symmetric key cryptography to realize dynamic operations (Ateniese et al., 2008). However, this scheme only supports block updating, deletion, and appending operations rather than true (physical) block insertion operation. Furthermore, Wang et al. (2010) presented the public cloud data auditing method by constructing the classic MHT for data authentication to satisfy full dynamic operations. In addition, this extended scheme makes use of bilinear aggregate signature to implement multiple audit tasks for different users simultaneously. These schemes above can verify data integrity but cannot recover data.

2.2. Proofs of retrievability

Juels and Kaliski (2007) first defined and proved the proofs of retrievability (POR) technique, which can verify data integrity and meanwhile recover data in its entirety by retaining and transmitting reliable file data. However, the data owner has to bear a heavy communication and verification burden because public verification is not involved in this scheme. Cash et al. (2017) took advantage of the oblivious RAM algorithmic to construct a novel scheme, in which the protocol is run between the user and the server to perform arbitrary reads/writes. However, the oblivious RAM is employed in this scheme, resulting in increasing practical expenditure. Shacham and Waters (2008) put forward two POR schemes based on BLS signatures and pseudorandom functions respectively and provided proof of security. While supporting public verification, they only considered static data files. Armknecht et al. (2018) built the formal framework of outsourcing proofs of retrievability (OPOR) and presented three instantiations of OPOR, in which the user outsources the task of implementing and validating the POR to the auditor. However, data update is not implemented in this scheme.

2.3. Public auditing and dynamic update scheme

The above two kinds of schemes both make data owners bear a heavy burden. Wang et al. (2013a) designed an efficient data verification scheme using proxy re-signature, in which the verifier can check the data integrity by partial data blocks. Liu et al. (2015) solved a new problem by utilizing regenerating code, in which the burden on data owners is reduced. However, this scheme cannot proceed dynamic data operations.

Shen et al. (2017) introduced a protocol of sampling blockless verification, which supports data dynamic operation by utilizing the location array and the doubly-linked-info-table. Moreover, this protocol achieves both public audit and batch audit. Li et al. (2016) put forward two lightweight protocols, which adopt offline and online signatures to allow batch validation and data dynamics. Shi et al. (2013) designed a novel scheme to balance user storage. However, the audit time of this scheme is slow. Erway et al. (2015) proposed a fully dynamic data verification scheme, which allows verifying the integrity of file data and file system. Yu et al. (2016a) designed a cloud data audit scheme with verifiable outsourcing, in which the secret keys of TPA are encrypted using the homomorphic blind encryption algorithm. Afterwards, Zhu et al. (2013) provided a service that supports dynamic audit by utilizing the table of index-hash and random sampling, which takes advantage of provable data update and anomaly detection. Jin et al. (2016) not only achieved efficient handling of data dynamics by index switcher, but also solved any possible dispute fairly by using fair arbitration protocols. Ren et al. (2018) exploited the ASBB encryption for enhanced dynamic proofs of retrievability, which possesses the characteristics of recoverability and public auditability.

2.4. Certificateless public audit and identity-based data integrity verification

To avoid security risks caused by certificate management, the first certificateless public auditing mechanism was proposed

(Wang et al., 2013b), in which the auditor does not need to manage the users certificate because of signatures of homomorphic certificates. To solve the problem of requiring complex key management, Yu et al. (2016b) designed an identity-based verification scheme using homomorphic technology, and this protocol achieved zero knowledge privacy. Afterwards, Li et al. (2017) presented the concept of fuzzy identity-based data auditing and constructed a scheme in which fuzzy identities are represented by biometrics. Shen et al. (2019) constructed a scheme for secure sharing with hiding sensitive information. Nevertheless, this proposal suffers from a key escrow problem. Li et al. (2022) devised a multicopy audit scheme in the multi-cloud environment to take advantage of the homomorphic verifiable tag. However, data dynamic operation and batch verification (Gudeme et al., 2021) are not satisfied in this scheme.

2.5. Data integrity verification for multiple own- ers and multiple clouds

To satisfy multiple owners and multiple clouds, Yang and Jia (2013) constructed an auditing protocol using the bilinear pairing. By this scheme, some characteristics, such as the data dynamic operations and batch auditing, are introduced to reduce the burden of users. Afterwards, Barsoum and Hasan (2015) presented a multicopy dynamic possession scheme based on map to verify multiple data distributed across multiple cloud centers, which can be against colluding servers and realizes public verifiability. Nevertheless, the verification process also brings a huge computing overhead to the user side. Zhu et al. (2012) realized cooperative PDP in multi-cloud storage by adopting hash index, which not only provides perfect security proof by zero-knowledge, but also satisfies dynamic data operations. Nayak and Tripathy (2021) proposed an efficient PDP method to cater for multi-owners and batch operation. Although these schemes have a lower computational cost, they fail to guarantee the storage correction (Yu and Hao, 2021).

3. Preliminaries

In this section, we take a brief look at the related techniques including lattice-based cryptography, sigma protocols, and ring signature.

3.1. Lattice-based cryptography

Definition 1. Let $\mathbf{B} = \{\boldsymbol{b_1}, \boldsymbol{b_2}, \cdots, \boldsymbol{b_m}\}$ be a $m \times m$-dimensional matrix, which consists of m vectors of linearly independent. The full-rank lattice $\mathbf{\Lambda} = \{y = \sum_{i=1}^{m} x_i \boldsymbol{b_i}, (x_1, x_2, \cdots, x_m) \in Z^m\}$ is generated by $\mathbf{B}$, where $\mathbf{B}$ is the basis of the lattice $\mathbf{\Lambda}$. A lattice can be represented by different lattice bases.

In fact, lattice theory was first applied in cryptography as an analysis tool. Many security analyses of lattice-based cryptography systems can be reduced to solving difficult problems in lattices. These studies explored the security analysis methods of classical mathematical problems based on public key cryptography, so that people realize the importance of lattice theory research on public key cryptography.

Definition 2. The LWE problem is defined as follows. Let $(\boldsymbol{A}, \boldsymbol{v})$ be the input pairs, the LWE search problem is described as: the process of finding a vector $\boldsymbol{s} \in Z_q^n$ such that $\boldsymbol{v} = \boldsymbol{As} + e$, where e is an error and it is chosen from distribution $\mathcal{X}^m$. The LWE decision problem is described as: determining whether the vector $\boldsymbol{v}$ is uniformly taken from Z_q^n or calculated from $\boldsymbol{v} = \boldsymbol{As} + e$, where $\boldsymbol{A}$ is chosen uniformly from $Z_q^{m \times n}$, $\boldsymbol{v}$ is chosen uniformly from Z_q^m, and m, n, q are positive integers.

The LWE problem has attracted wide attention in lattice-based cryptography. In particular, it has important applications in the design of lattice cryptosystem.

3.2. Sigma protocol

Definition 3. The sigma protocol is an interactive process that relates the prover and the verifier. Firstly, the prover creates a commitment and sends it to the verifier. After obtaining the commitment, the verifier obtains a random challenge in the domain of challenge and sends it to the prover. Afterwards, the prover creates a response and returns the response to the verifier. In the end, the response is verified by the verifier and the result is outputted.

3.3. Ring signature

Definition 4. A ring signature scheme (Rivest et al., 2001) consists of four probabilistic polynomial time algorithms, which are detailed as follows:

- **Setup**$((1^k) \rightarrow pp)$ is an algorithm of public parameter generation, in which the public parameter pp is obtained by the security parameter 1^k.
- **KeyGen**$(pp \rightarrow (pk, sk))$ is an algorithm of key generation, in which a pair of public and secret keys (pk, sk) is generated according to the public parameter pp.
- **Sign**$((sk, M, R) \rightarrow \sigma)$ is a signature algorithm, in which the signature σ is generated according to the secret key sk, the message M, and the list of public keys $R = \{pk_1, pk_2, \cdots, pk_V\}$.
- **Verify**$((R, M, \sigma) \rightarrow 1/0)$ is a verification algorithm that takes a ring $R = \{pk_1, pk_2, \cdots, pk_V\}$, a message M, and a signature σ as inputs, and outputs either 1 or 0.

We require a ring signature scheme to satisfy the following properties:

- **Correctness**: If the message is signed according to the correct signing steps and the signature is not tampered with during propagation, then the ring signature satisfies the signature verification equation.
- **Anonymity**: Even if the attacker illegally obtains the secret keys of all possible signers, the probability that he can identify the real signer is no more than $1/N$, where N is the number of all possible signers.
- **Unforgeability**: Even if an external attack can get the signature of any message M from a random prophet that generates the ring signature without knowing any member's private key, the probability of a legitimate signature being successfully forged is negligible.

4. Problem statement

4.1. System model

As shown in Fig. 1, there are three entities in the system model of DIVRS : third party auditor, cloud server provider, and data owner. Their roles are elaborated as follows.

- **Data owner (DO)**. The data owner first divides the data into data block in the process of tag generation. Afterwards, he/she generates the tag of data block according to secret key to verify the data integrity. Finally, data and tags are uploaded by the data owner to the cloud to save the costs of data storage and query services. When the data owner verifies the integrity of the cloud data, he/she issues an audit request to the TPA.

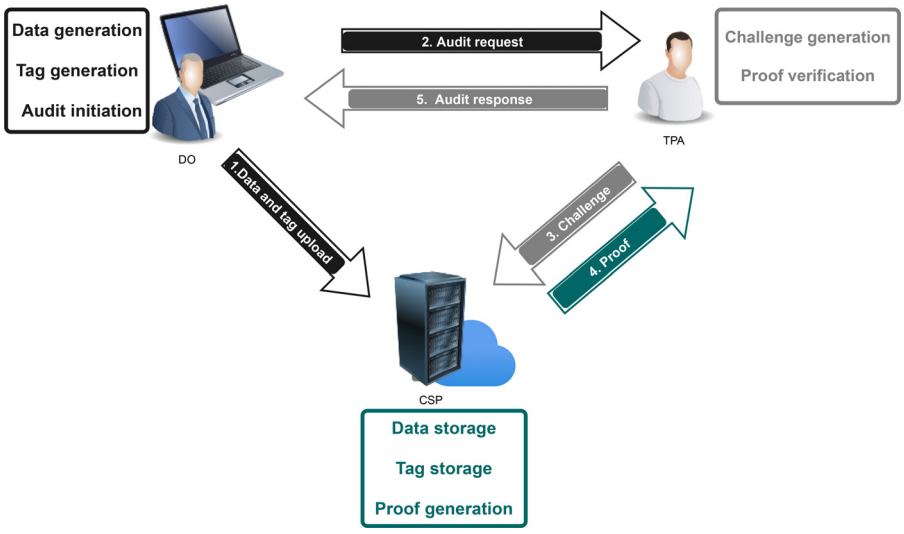

Fig. 1. System model.

- **Cloud server provider (CSP)**. The cloud server provider stores the data and the data tags because it has sufficient computational and storage resources. However, the cloud server provider is considered malicious and therefore have the potential to delete or tamper with the data of data owner. In the phase of proof generation, the cloud server provider calculates the proof information and sends it to TPA when a challenging message is obtained from TPA.
- **Third party auditor (TPA)**. As a trusted third party, the TPA can replace the data owner to verify the integrity of cloud data, as it has strong computing power. In the stage of challenge, a challenge message is generated by the third party auditor and it is sent to the server. In addition, the third party auditor checks the data by information of proof and the public key, and then he/she returns the verification result in the phase of verification.

4.2. Threat model

In the DIVRS scheme, the cloud servers are considered as semi-trusted entities. Specifically, the cloud server provider may delete or tamper data to save computing and bandwidth resources. In addition, with the development of quantum computers, most of the schemes will face security threat. Based on the above analysis, the following two threat scenarios are concluded.

- **The malicious cloud server provider**. After the data are uploaded to the server, a malicious CSP may delete data that have rarely accessed to reclaim storage space in the cloud storage for its own benefit. Furthermore, some CSPs may tamper with data, which brings tremendous economic losses to data owners.
- **The quantum environment**. Quantum computers can easily solve traditional difficult problems. Therefore, with the development of quantum computers, most traditional asymmetric cryptographic primitives may become insecure.

4.3. The core goals of DIVRS

To solve the threats mentioned above, we design a DIVRS scheme that not only quickly verifies the integrity of the cloud data, but also has the security in the quantum environment. The design goals are as follows:

- **Public auditing**. Any TPA can replace the data owner to verify the integrity of cloud data without downloading entire outsourced data, thus public verification is an essential requirement, which reduces computation and storage overhead for the data owner.
- **Storage correctness**. To achieve some purposes, data stored in the cloud servers may be corrupted or lost in some circumstances. Therefore, the result of the third party auditor verified was 1 if and only if the cloud keeps the data intact.
- **Blockless Verification**. By checking the integrity of a randomly selected block, auditors can verify the integrity of all blocks in the DIVRS scheme, which aims to reduce computational costs and bandwidth consumption.
- **Unforgeability**. It must be computationally infeasible for the cloud to forge a response in the auditing phase, otherwise the scheme is no longer secure.
- **Post-quantum security**. We design a data auditing scheme based on lattice cipher in this paper, which can resist the attack of quantum computer.

5. The DIVRS scheme

5.1. Overview of the DIVRS scheme

The DIVRS framework consists of the following six algorithms, **Setup, KeyGen, TagGen, Challenge, ProofGen**, and **Verify**. A summarization of these algorithms is presented in Table 1.

Table 1
Overview of the scheme.

DO side	$pp \leftarrow$ **Setup**(λ) is an algorithm of setup, in which public parameter pp is obtained according to the security parameter λ. $(pk, sk) \leftarrow$ **KeyGen**(pp) is an algorithm of key generation, in which the public and secret key pair (pk, sk) are generated by the public parameter pp. $T \leftarrow$ **TagGen**(pk, sk, F) is an algorithm of tag generation, in which the data tag T is generated according to the public and secret key pair (pk, sk) and the data set F.
CSP side	$P \leftarrow$ **ProGen**$(pk, \sigma, chal)$ is the proof generation algorithm, in which the proof P is generated according to the public key pk, the signature σ, and the challenge information $chal$.
TPA side	$chal \leftarrow$ **Challenge**(pk) is the challenge generation algorithm, in which the challenge information $chal$ is generated by the public key pk. $0\backslash 1 \leftarrow$ **Verify**(pk, u) is the data integrity verification algorithm, in which the verification result $0\backslash 1$ is obtained according to the public key pk and the proof u.

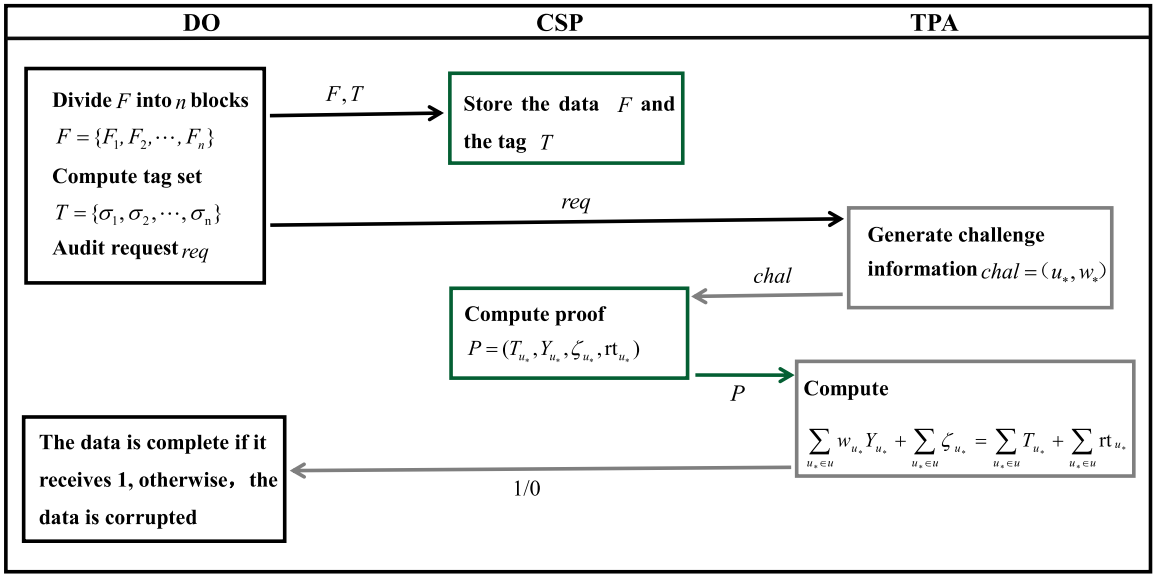

Fig. 2. The protocol of DIVRS.

First of all, the data owner runs **Setup** to generate a public parameter set pp, and then runs **KeyGen** to generate public and secret key pair (pk, sk) using public parameters. Meanwhile, the DO divides data F into n blocks, i.e., $F = \{F_1, F_2, \cdots, F_n\}$. Secondly, the data owner calculates the signature σ_i using ring signature algorithm for each data block F_i, and then the tag $T = (\sigma_1, \sigma_2, \cdots, \sigma_n)$ and the data $F = \{F_1, F_2, \cdots, F_n\}$ are uploaded to the CSP. Afterwards, the DO issues an audit request to the TPA.

After receiving the audit request from the DO, the TPA generates the challenge information $chal$ by **TagGen** is shown below. A subset $u = (u_1, u_2, \cdots, u_v)$ is chosen randomly by the TPA from the set $(1, 2, \cdots, n)$, and then the TPA chooses a random value $w_* \in Z_p^*$ as coefficient. Therefore, the challenge message $chal = (u_*, w_*)$ is generated. At the same time, $chal$ is sent to the CSP. After receiving the challenge information, the CSP runs **proofGen** to generate the proof P and it is sent to the TPA. Finally, the TPA runs **Verify** to verify the cloud data. The equation of verification is calculated by the TPA, and it returns the verification result 1/0.

5.2. The scheme of DIVRS

The proposed scheme based on ring signature in cloud has six stages, which are defined as follows. Figure 2 presents the details of the protocol of DIVRS.

- **Setup**. This algorithm is used to generate public parameters. Let $G := R_p^m \times R_p^r \times R_p^r$ be an additive group, where $p = 5$ mod 8 and m, r are integers. Let G_1, G_2 be two symmetric subsets of G, and $Q_{\mathcal{K}}$ be the uniform distribution over $R_p^{r\times m}$. A set of group actions Δ: $G \times \mathcal{K} \to \mathcal{K}$ is uniquely defined by a matrix $\mathcal{K} \in R_p^{r\times m}$. Finally, $(G, \mathcal{K}, G_1, G_2, Q_{\mathcal{K}}, \Delta)$ and a fixed $K_0 \in \mathcal{K}$ are outputted as the public parameter pp.
- **Key generation**. This algorithm is used to produce a pair of secret and public keys (sk, pk), which is generated by the DO. More specifically, the DO first chooses a random value g from the finite set G_1, and calculates a value $k = g \times K_0$. For $b \in (0, 1)$, we define

$$G_b := \{(g, e, \tilde{e}) \in G \mid ||g||_\infty, ||e||_\infty, ||\tilde{e}||_\infty \le D_b\} \tag{1}$$

and

$$l_i = Lc_i + e_i, \tag{2}$$

where c_i, e_i are drawn uniformly from G_1. The secret key is $sk = c_i$, and the public key is $pk = (l_1, l_2, \cdots, l_N)$.
- **Tag generation**. The aim of this algorithm is to generate the tag of data by utilizing ring signature. To this end, the data owner first computes a commitment com. The calculation details are shown below. First of all, the file F held by the data owner consists of n blocks. i.e., $F = \{F_1, F_2, \cdots, F_n\}$. For notational simplicity, $(sd_1, sd_2, \ldots, sd_{\mathrm{M}})$ is obtained by algorithm $\mathrm{SdTree}^{\mathcal{O}'}(sd_{rt}, \mathrm{M})$ (Ward et al., 2020), where $sd_{rt} \in \{0,1\}^\lambda$, and we set

$$\mathcal{O}'(png||sd_h||h) := \mathcal{O}(st_i||png||sd_h||h), \tag{3}$$

where $\mathcal{O}$ is a random oracle, png is a pseudorandom number generator, and $st_i \in \{0,1\}^{2\lambda}$. For i from 1 to M, $\mathcal{O}_i(png||sd_h||h)$ is computed by the following equation:

$$\mathcal{O}_i(png||sd_h||h) := \mathcal{O}(st_i||i||png||sd_h||h), \tag{4}$$

$$\mathcal{O}(png||sd_i) = (s_i, \alpha_{1i}, \alpha_{2i}, \cdots, \alpha_{Ni}), \tag{5}$$

where s_i is a probability distribution over R_p, and α_{ji} is chosen uniformly from $\{0,1\}^\lambda$. After that, for j from 1 to N, the data owner calculates

$$\mathrm{L}_j = \lfloor Ls_i + l_j \rceil \in R^r \tag{6}$$

and

$$\mathrm{C}_{ji} = \mathcal{O}_i(\mathrm{L}_j||\alpha_{ji}), \tag{7}$$

and then the merkle tree $\mathrm{MT_i}$ is constructed by $(\mathrm{C}_{i1}, \mathrm{C}_{i2}, \cdots, \mathrm{C}_{iN})$, the rt_i is the root of the tree MT_i. Finally, the data owner calculates

$$\zeta = (st_i, \zeta_1, \zeta_2, \cdots, \zeta_{\mathrm{M}}) \tag{8}$$

and

$$Y_* = \mathcal{H}(F_*, R, \zeta), \tag{9}$$

where $\zeta_i = \mathrm{rt_i}$, R is a ring, and $\mathcal{H}$ is the hash function SHA256. For notational simplicity, let $Y_* = (y_1, y_2, \cdots, y_\omega)$, If y_* is 0, the data owner computes $\phi = s_i + c_i$, where B_1, B_2 are integers such that $B_1 < B_2 < q$, $s_i \in [-B_2, B_2]^m$ and $\phi \in [-(B_2 - B_1), (B_2 - B_1)]^m$. Afterwards, the data owner calculates $\eta_i = (\phi, \tau, \alpha_i)$, where τ is obtained by algorithm getMerklePath

(Ward et al., 2020). If y_* is 1, the data owner calculates $\eta_i = sd_i$ and $\eta_* = (sd_\rho, \{\eta_i\})$, where sd_ρ is obtained by algorithm ReleaseSds$^{\mathcal{O}'}$ (Ward et al., 2020). For each data block F_*, the DO computes a tag

$$\sigma_* = (st_*, Y_*, \eta_*, \zeta_*). \tag{10}$$

For notational simplicity, we denote the set of tags by $T = (\sigma_1, \sigma_2, \cdots, \sigma_n)$. Finally, the DO sends the tags T and the data $F = \{F_1, F_2, \cdots, F_n\}$ to the cloud server provider. The overall process of tag generation is summarized in Algorithm 1.

- **Challenge**. Upon receiving the audit request, the TPA randomly chooses a subset $u = (u_1, u_2, \cdots, u_v)$ from the set $(1, 2, \cdots, n)$, in which $c \ll n$. In addition, the TPA randomly chooses a value $w_* \in Z_p$ as coefficient for each u_*, the challenge message $chal = (u_*, w_*)$ is generated. Afterwards, the TPA sends $chal$ to the CSP for integrity auditing.
- **Proof generation**. After receiving challenge information $chal$ from the TPA, the CSP calculates $T_{u_*} = w_{u_*}\mathcal{H}(F_{u_*}, R, \zeta)$, and $\widetilde{\mathrm{rt}}_{u_*}$ is obtained by algorithm ReconstructRoot (Ward et al., 2020), where $u_* \in u$. Afterwards, the CSP sends $P = (T_{u_*}, Y_{u_*}, \zeta_{u_*}, \widetilde{\mathrm{rt}}_{u_*})$ to the TPA as the proof.
- **Verify**. Upon receiving the responses from the CSP, the TPA first calculates $\varpi_i = w_{u_i} Y_{u_i}$, then he/she calculates $\varkappa = \sum_{u_* \in u} \varpi_i + \sum_{u_* \in u} \zeta_{u_i}$. Finally, the proof P is verified according to the following formula:

$$\varkappa = \sum_{u_* \in u} T_{u_*} + \sum_{u_* \in u} \widetilde{\mathrm{rt}}_{u_*}. \tag{11}$$

6. Security analysis

Theorem 1 **(Correctness)**: The proof can pass the TPAs verification if the TPA and the DO are honest in obeying the specified procedures.

Proof. Through the above scheme description, we can calculate:

$$\begin{aligned} \widetilde{\mathrm{C}}_{u_*} &= \mathcal{O}_i(\widetilde{\mathrm{R}}_{u_*} || \alpha_{u_*}) \\ &= \mathcal{O}_i(\lfloor L\phi_i \rceil || \alpha_{u_*}) \\ &= \mathcal{O}_i(\lfloor L(s_i + c_i) \rceil || \alpha_{u_*}) \\ &= \mathcal{O}_i(\lfloor (Ls_i + Ac_i) \rceil || \alpha_{u_*}), \end{aligned} \tag{12}$$

$$\begin{aligned} \mathrm{C}_{ji} &= \mathcal{O}_i(R_j || \alpha_{ji}) \\ &= \mathcal{O}_i(\lfloor Ls_i + l_j \rceil || \alpha_{ji}) \\ &= \mathcal{O}_i(\lfloor Ls_i + Lc_j + e_j \rceil || \alpha_{ji}). \end{aligned} \tag{13}$$

Obviously, $\widetilde{\mathrm{rt}}_{u_*}$ obtained by algorithm ReconstructRoot (Ward et al., 2020) and ζ_{u_*} constructed by C_{ji} are equal. Moreover, $||\phi||_\infty \le B_2 - B_1$, $L\phi \notin Border_{q,d,B_1}$. Therefore,

$$\begin{aligned} \sum_{u_* \in u} w_{u_*} Y_{u_*} + \sum_{u_* \in u} \zeta_{u_*} &= \sum_{u_* \in u} w_{u_*} \mathcal{H}(F_{u_*}, R, \zeta) + \sum_{u_* \in u} \zeta_{u_*} \\ &= \sum_{u_* \in u} T_{u_*} + \sum_{u_* \in u} \widetilde{\mathrm{rt}}_{u_*}. \end{aligned} \tag{14}$$

Theorem 1 (Unforgeability of tag)**.** *: The scheme satisfies tag unforgeability, if any PPT adversary $\mathcal{A}$ has negligible advantage against a challenger $\mathcal{C}$ in the following game.*

Setup. A challenger $\mathcal{C}$ executes the initialization of system to generate public parameters and obtains the key pair (pk_i, sk_i) by the **KeyGen** algorithm for all $i \in N$ using random coins rr_i. It sets $PK = \{pk_i\}$ and initializes two empty sets: ES_1 and ES_2. Finally, the challenger provides pp and PK to $\mathcal{A}$.

Sign Query. The challenger $\mathcal{C}$ checks if $pk_i \in R$, and if so it generates the tag for F_i by performing **TagGen**. The challenger provides the tag value to $\mathcal{A}$ and adds (i, F_i, R_i) to ES_1.

Corruption Query. The challenger $\mathcal{C}$ adds pk_i to ES_2 and returns rr_i to $\mathcal{A}$.

Forgery. $\mathcal{A}$ outputs the corresponding public key R_i', a messages F_i', and a tag σ_i'.

If $R_i' \subset PK \backslash ES_2$, $(\cdot, F_i', R_i') \notin ES_1$, and **Verify**$(R_i', F_i', \sigma_i') = 1$, we say the adversary $\mathcal{A}$ wins.

Proof. We prove that the advantage that $\mathcal{A}$ wins the game is negligible. Firstly, we assume that an adversary outputs a forgery (R^*, F^*, σ^*) successfully with this challenger, where σ^* is a proof constructed of the ith query to the random oracle. Afterwards, confirming the randomness of the adversary up to the ith random oracle query, the adversary will output a forgery (R'^*, F'^*, σ'^*) with non-negligible probability if the adversary run again and we answer the random oracle queries using new randomness, where σ'^* is again a proof constructed in the ith query to the random oracle. We respond to random oracles in the same way and use the same randomness against adversary. Therefore, we have $F^* = F'^*, R^* = R'^*$, and $com^* = com'^*$. In particular, we obtain two valid transcripts on the same commitment com. However, this cannot happen with non-negligible probability. □

Theorem 2 ((Anonymity))**.** *: The scheme satisfies anonymity, if any PPT adversary $\mathcal{A}$ has negligible advantage against a challenger $\mathcal{C}$ in the following game.*

Proof. A challenger $\mathcal{C}$ obtains public parameters and the key pair (pk_i, sk_i) for all $i \in N$ using random coins rr_i by **Setup** and **KeyGen**, respectively. Then a random bit $b \in (0, 1)$ is sampled. Finally, the challenger provides PP and $\{rr_i\}$ to $\mathcal{A}$.

The adversary $\mathcal{A}$ outputs a challenge (R, F_i, i_0, i_1) to $\mathcal{C}$, where the ring R must contain pk_{i_0} and pk_{i_1}. Afterwards, the challenger $\mathcal{C}$ runs **Sign** to generate σ^*, and provides it to $\mathcal{A}$. Finally, $\mathcal{A}$ outputs a guess b^*. If $b^* = b$, we say the adversary $\mathcal{A}$ wins. Since the proof is generated independently from the secret keys sk_{i_0} and sk_{i_1}, anonymity holds. □

7. Performance analysis

For our comparative analysis, we evaluate the performance of the DIVRS scheme in terms of computation and communication costs. In addition, actual performances of the DIVRS scheme are compared in experiments. All the experiments are conducted on a computer, it is equipped with Windows 10 system, an Intel (R) Core (TM) i7-6500 CPU. We implement the signing and verification in Dilithium NIST on a Xilinx Zynq-7000 FPGA platform. The performance parameters of simulation platform are presented in Table 2.

7.1. Theoretical analysis

The communication and computation costs are analyzed in the stage of tag generation, proof generation, and verification in the DIVRS scheme and other six schemes, and Table 3 shows some definitions of theoretical analysis.

7.1.1. Communication cost

The TPA sends (i, v_i) to the cloud server as a challenge in the scheme of SEPDP (Nayak and Tripathy, 2021). Therefore, the communication cost is $v(p + a)$. Furthermore, the CSP returns (α, γ, R)

Table 2
Simulation platform.

Operating system	CentOS 7.8. x86 64
CPU	Intel(R) Core (TM) i7-6500
Memory	8GB RAM
Program language	C

Table 3
Notation of operations.

Symbol	Operation
Hash	hash operation
$Mult_p$	point multiplication
Add	addition
$Mult_m$	matrix-vector multiplication
Exp	exponential operation
Pair	pairing operation
Inv	inverse operation

(Nayak and Tripathy, 2021) and (σ, λ) (Shen et al., 2019) as the response, respectively. Therefore, the communication cost in the proof generation stage is $3d$ and $p+v$, respectively, where d is the size of element in G. In Li et al. (2022), the TPA submits the challenge information $(c, k_1, k_2)_{k_1,k_2 \in Z_p}$ to the cloud organizer. Therefore, the communication cost of the challenge stage is $logv + 2p$. In addition, the cloud organizer returns the integrity proof $(\sigma, \{M_k\}, R, \{u_k\})_{1 \le k \le f}$ to TPA, hence the communication cost of proof stage is $op + (2+f)g_1$, where o is the number of sectors of each data block.

The TPA sends (id_{x_i}, v_i) to the cloud in the stage of challenge (Hahn et al., 2022). Therefore, the communication cost is $v(p+t)$, where t is the size of each index. In addition, the cloud sends (σ, u) to the TPA in the stage of proof, where $\sigma_i \in G_1$. Let g_1 be the size of an element of G_1, the communication cost is $p + g_1$.

In the stage of challenge, the TPA sends $chal = (u_*, w_*)$ to the CSP for integrity auditing. Let v represents the number of challenged blocks for each auditing, a and p are the size of an element of set $(1, n)$ and Z_p. Therefore, the communication cost of the DIVRS in the stage of challenge is $v(p+a)$. In the stage of proof generation, the CSP sends $(l_{u_*}, Y_{u_*}, \zeta_{u_*}, \widetilde{rt}_{u_*})$ to the TPA as proof, the communication cost of our DIVRS in the proof generation stage is $2\lambda(v+1)+v$. From the above analysis, the communication costs in the stage of challenge and proof generation are acceptable. A comparison of the communication costs of the seven schemes is shown in Table 4.

7.1.2. Computation cost

As shown in Table 5, the computational cost is evaluated in terms of tag generation, proof generation and proof verification in the DIVRS scheme and other six schemes.

In the stage of TagGen, it is obvious that the overhead of computation is independent of the sector number (Hahn et al., 2022; Liu et al., 2015; Nayak and Tripathy, 2021; Shen et al., 2019), while those two schemes (Li et al., 2022; Yang and Jia, 2013) have increased linearly. In addition, our scheme does not involve pair and exponential operations. In the process of ProGen, the number of challenge blocks is the main factor affecting the computational overhead of the five schemes (Hahn et al., 2022; Liu et al., 2015; Nayak and Tripathy, 2021; Shen et al., 2019; Yang and Jia, 2013). However, in addition to the number of challenge blocks, the replica count is another factor in the scheme Li et al. (2022). In the DIVRS scheme, the auditing proof is generated by the CSP and the proof is verified by the TPA. Therefore, DO and TPA have relatively small computational cost in our scheme, and the computational cost of CSP is acceptable. In brief, the evaluation of theoretical performances pointed out that the DIVRS scheme has acceptable overheads in computation and communication.

7.2. Experimental result

7.2.1. Setup

We run these experiments on a Windows 10 operating system with Intel Core i7- 6500 CPU, 8GB RAM, using C language. We set the security level as 128 bits and use 128-bit *sd* and 256-bit commitment and hash values. The data block size is considered 512 bits with the total block number less than 3000. The ring signature algorithm is implemented based on the Faafl implementation by Ward et al. (2020). We use the ring $R_q = Z_q[X]/(X^{256}+1)$ and the coefficients of the LWE secrets are sampled uniformly from $[-6, 6]$. In additional, the experiment is performed on six randomly generated files, in which each file consists of $n \in \{500, 1000, 1500, 2000, 2500, 3000\}$ blocks. For ProGen and verification, the performance depends on the amount of challenged blocks $v \in \{50, 100, 150, 200, 250, 300\}$. In the experiments, we set the key size to be 1024 bits. We set the number of data blocks ranging from 50 to 3000 and set the number of challenged blocks ranging from 50 to 300 when $n = 3000$ in our experiments. All the results of the evaluation are average results over 10 runs.

7.2.2. Tag generation time

In the stage of tag generation, the data owner first computes a commitment, and then the signature is generated for each data block according to the commitment and the merkle tree. Finally, the signatures of all data blocks are defined as the set of tags. As the tag generation time is mostly depended on some group mathematical operations and the selected signature algorithms, Fig. 3 demonstrates the time costs in the stage of tag generation for the four schemes, in which the horizontal and vertical axes represent the number of data blocks and the time of consumption, respectively. In these schemes (Hahn et al., 2022; Li et al., 2022; Shen et al., 2019), the tag generation time will also increase significantly with the number of data blocks. The DIVRS scheme is much faster than other methods (Hahn et al., 2022; Li et al., 2022; Shen et al., 2019) because its computation complexity in relation to tag generation depends on the ring signature.

7.2.3. Proof generation time

In the stage of proof generation, after receiving challenge information form the TPA, the CSP calculates the proof and sends it to the TPA. In addition, the time of proof generation mainly depends on the size of the subset and the number of data blocks that need to be checked. We evaluate the time of proof generation by varying the number of challenged blocks. In both schemes (Shen et al., 2019; Yang and Jia, 2013), the proof generation time will also increase significantly with the number of challenged blocks. However, it is obvious that the proof generation time increases slowly with the number of challenged blocks in schemes (Shen et al., 2019; Yang and Jia, 2013). Figure 4 shows the time consumption for the stage of proof generation. The DIVRS scheme does not use bilinear pairings, the time of proof generation time is shorter than other four schemes (Hahn et al., 2022; Li et al., 2022; Shen et al., 2019; Yang and Jia, 2013).

Table 4
Communication cost comparison.

	Liu et al. (2015)	Shen et al. (2019)	Li et al. (2022)	Yang and Jia (2013)	Nayak and Tripathy (2021)	Hahn et al. (2022)	DIVLS
Challenge stage	$(v+2)p$	$v(p+a)$	$(\log v + 2p)$	$v(q+a)$	$v(p+a)$	$v(p+t)$	$v(p+a)$
ProofGen stage	$(2f+v)p$	$p+v$	$op + (2+f)g_1$	$(g_1+a)p + vq$	$3d$	$p+g_1$	$2\lambda(v+1)+v$

Table 5
Computational cost comparison.

	TagGen(DO)	ProGen(CSP)	Verify(TPA)
Liu et al. (2015)	$(n+2)Pair + Mult_p + Hash$	$(2v+3x+1)Mult_p + (v+2x-3)Add$	$(1+n)Exp + nMult_p$
Shen et al. (2019)	$n(Hash + 2Mult_p + 2Exp)$	$(2v-1)Mult_p + vExp + (v-1)Add$	$4Pair + (v+l_{ID}+2)Mult_p + 2cAdd$ $-Add + (l_{ID}+v+3)Exp + vHash$
Li et al. (2022)	$2n(Exp + Mult_p) + (o+1)Exp$	$v(Exp + Mult_p)$	$3Pair + (vN_c + o + 3)Exp + (vN + o)Mult_p$
Yang and Jia (2013)	$2sExp + (o+n-1)Mult_p + nHash$	$(2v-1)Mult_p + (v+s-2)Add + (o+v)Exp + oPair$	$vExp + Mult_p + vHash + (v-1)Add + 2Pair$
Nayak and Tripathy (2021)	$Exp + 2nMult_p + nHash + Inv$	$3Exp + (3v+1)Mult_p + vHash$	$5Exp + (2n+3v+1)Mul_p + (n+v)Hash + Inv$
Hahn et al. (2022)	$n(3Hash + Exp)$	$v(Mult + Hash + Exp)$	$(2v+1)Hash + 2Exp + 2Mult_p + 2Pair$
DIVRS	$nAdd + nMult_m + nHsah$	$v(Mult_p + Hsah)$	$vMult_p + (4v+2)Add$

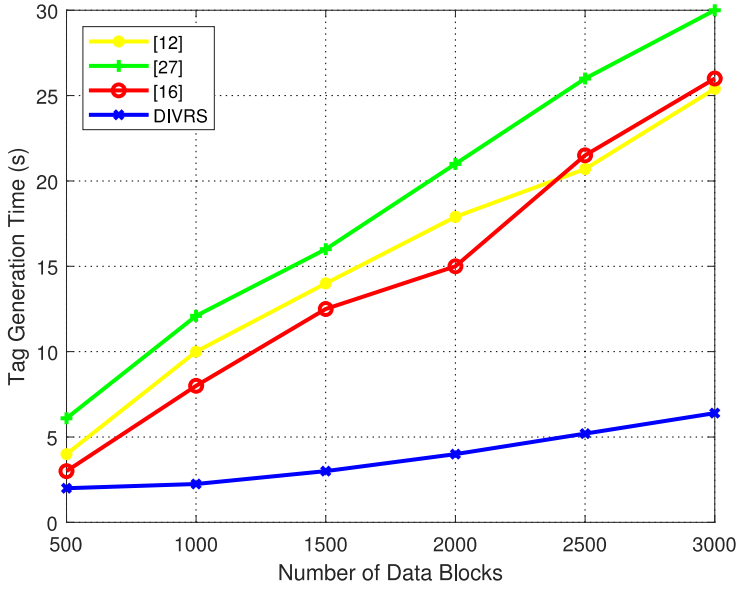

Fig. 3. Tag Generation Time.

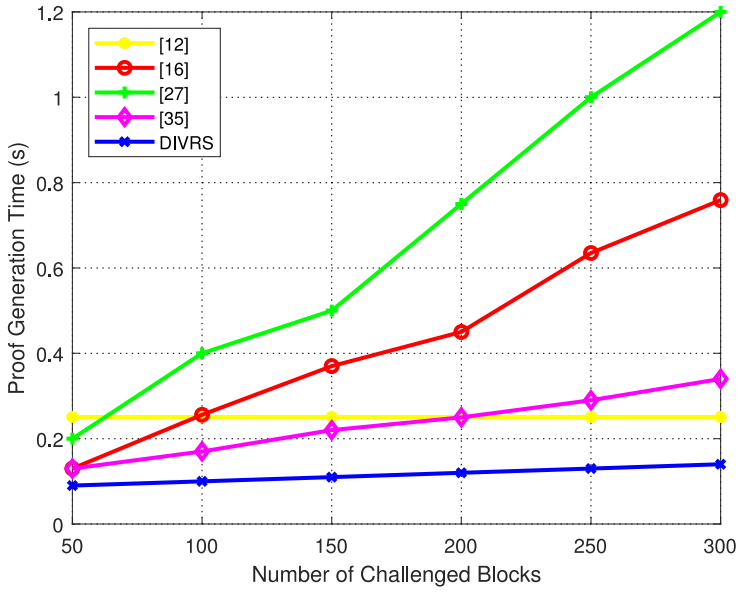

Fig. 4. Proof Generation Time.

7.2.4. Verification time

Let us now consider the verification phase. Upon receiving the responses from the CSP, the TPA verifies the proof according to Eq. (11), the data is complete if Eq. (11) is true, otherwise, the data is corrupted. Finally, the data owner receives the auditing result from the TPA. To improve the efficiency of verification, the algorithm based on ring signature is used in our scheme. To visually show the comparison of the verify time of our proposed scheme and the other three schemes (Yang and Jia, 2013), we run the experiment of the verify algorithm. Figure 5 displays the time consumption for the stage of proof verification. It can be observed that the verification time increases with the number of challenged

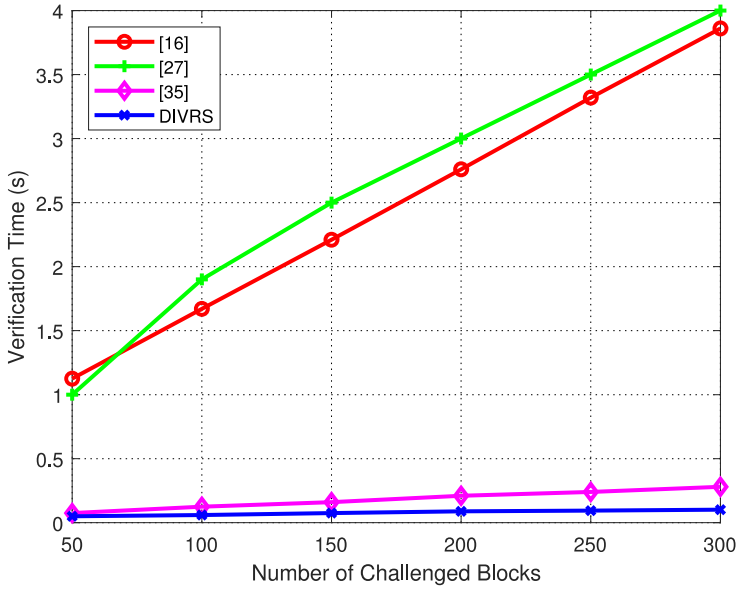

Fig. 5. Verification Time.

Algorithm 1 Tag Generation.

Input: $F_1, F_2, \cdots, F_n$, sk, R
Output: $\sigma_1, \sigma_2, \cdots, \sigma_n$

Run ReleaseSds$^{\mathcal{O}'}$ to obtain $(sd_1, sd_2, \ldots, sd_{\mathrm{M}})$;
Run SdTree$^{\mathcal{O}'}$ (sd_{rt}, M) to obtain sd_ρ;
for $i = 1$ to n **do**
  for $j = 1$ to N **do**
    $\mathrm{C}_{ji} = \mathcal{O}_i(\mathrm{L}_j || \alpha_{ji})$
  end for
end for
Construct the merkle tree MT_i by $(\mathrm{C}_{i1}, (\mathrm{C}_{i2}, \cdots, (\mathrm{C}_{iN})$
for $i = 1$ to n **do**
  $st_i \in \{0,1\}^{2\lambda}$
  $\zeta_i = \mathrm{rt}_i$, where rt_i is the root of the tree MT_i
  $Y_i = \mathcal{H}(F_i, R, st_i, \{\zeta_i\})$
  $\eta_i = (sd_\rho, \{sd_i\})$
  $\sigma_i = (st_i, Y_i, \eta_i, \zeta_i)$
end for
return $\sigma_1, \sigma_2, \cdots, \sigma_n$

Algorithm 2 Proof Verification.

Input: $\{T_{u_i}\}, \{Y_{u_i}\}, \{\zeta_{u_i}\}, \{\widetilde{rt}_{u_i}\}, \{w_i\}$
Output: $1/0$

for $i = 1$ to v **do**
  $\varpi_i = w_{u_i} Y_{u_i}$
  $\varkappa = \sum_{i=0}^{v} \varpi_i + \sum_{i=0}^{v} \zeta_{u_i}$
end for
if $\varkappa = \sum_{i=0}^{v} T_{u_i} + \sum_{i=0}^{v} \widetilde{rt}_{u_i}$ **then**
  Output 1;
else
  Output 0.
end if
return Verification result $1/0$

blocks. As shown in Fig. 5, the DIVRS scheme exhibits the best performance, and it is slightly faster than (Yang and Jia, 2013). The performance of Shen et al. (2019) and Li et al. (2022) are the worst due to the greater number of exponential computations. Therefore, the DIVRS scheme is capable of effectively verifying the data integrity of data owners.

8. Conclusion

In this paper, a novel data integrity verification scheme is presented based on ring signature in cloud storage, which achieves post-quantum security. Specifically, the data are divided into data blocks by the DO and signatures are calculated by the ring signature algorithm for each data block. Meanwhile, the set of signature and the data are uploaded to the CS. Afterwards, the DO issues an audit request to the TPA. After receiving the audit request from the data owner, the challenge information is generated by the third party auditor and it is sent to the CSP. Moreover, the proof is generated by the cloud server provider according to challenge information and it is sent to the TPA. Finally, the TPA checks the integrity of the cloud data by the verification equation. The proposed scheme not only verifies the integrity of cloud data, but also ensures the security in the quantum environment. In addition, we evaluate the performance of the DIVRS scheme in terms of computation and communication costs, and the analysis shows the effectiveness of this scheme compared with the other six data audit schemes. We also compare the tag generation time, proof generation time, and verification time of DIVRS scheme with the other schemes in experiments. The experimental results demonstrate that the DIVRS scheme is feasible in cloud storage.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Miao Tian: Conceptualization, Methodology, Formal analysis, Investigation, Writing – original draft. **Yushu Zhang:** Resources, Data curation, Funding acquisition, Writing – review & editing. **Youwen Zhu:** Project administration. **Liangmin Wang:** Software, Validation, Visualization. **Yong Xiang:** Supervision.

Acknowledgments

The work was supported by National Key R D Program of China (2020YFB1005500).

References

Ajtai, M., 1996. Generating hard instances of lattice problems. In: Proceedings of the 28th Annual ACM Symposium on Theory of Computing, pp. 99–108.
Ajtai, M., Dwork, C., 1997. A public-key cryptosystem with worst-case/avarage-case equivalence. 29th ACM Symposium on Theory of Computing.
Armknecht, F., Bohli, J.-M., Karame, G., Li, W., 2018. Outsourcing proofs of retrievability. IEEE Trans. Cloud Comput. 9 (1), 286–301.
Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D., 2007. Provable data possession at untrusted stores. In: Proc. of CCS., pp. 598–609.
Ateniese, G., Di Pietro, R., Mancini, L.V., Tsudik, G., 2008. Scalable and efficient provable data possession. In: Proceedings of the 4th Int. Conf. Security Privacy Communication Netowrks, pp. 1–10.
Barsoum, A.F., Hasan, M.A., 2015. Provable multicopy dynamic data possession in cloud computing systems. IEEE Trans. Inf. Forensics Secur. 10 (3), 485–497.
Cash, D., Küpçü, A., Wichs, D., 2017. Dynamic proofs of retrievability via oblivious ram. In: Proc. 36nd Int. Conf. Theory Appl. Cryptogr. Tech. (EUROCRYPT), pp. 22–57.
Erway, C.C., Küpçü, A., Papamanthou, C., Tamassia, R., 2015. Dynamic provable data possession. ACM Trans. Inf. Syst. Secur. 17 (4), 1–29.
Gentry, C., 2009. Fully homomorphic encryption using ideal lattices. In: Proceedings of the 41st Annual ACM Symposium on Theory of Computing., pp. 169–178.
Goldreich, O., Goldwasser, S., Halevi, S., 1997. Public-key cryptosystems from lattice reduction problems. In: Proceedings of Advances in Cryptology (CRYPTO), Vol. 1294.
Gudeme, J., Pasupuleti, S., Kandukuri, R., 2021. Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage - sciencedirect. Comput. Secur. 103.
Hahn, C., Kwon, H., Kim, D., Hur, J., 2022. Enabling fast public auditing and data dynamics in cloud services. IEEE Trans. Serv. Comput. 15 (4), 2047–2059.
Hoffstein, J., Pipher, J., Silverman, J.H., 1998. NTRU: A ring-based public key cryptosystem. Algorithmic Number Theory. 10 (1007), 267–288.
Jin, H., Jiang, H., Zhou, K., 2016. Dynamic and public auditing with fair arbitration for cloud data. IEEE Trans. Cloud Comput. 6 (3), 680–693.
Juels, A., Kaliski Jr., B.S., 2007. Pors: proofs of retrievability for large files. In: Proceedings of the 14th ACM Conf. Computer Communications Security, pp. 584–597.
Li, J., Yan, H., Zhang, Y., 2022. Efficient identity-based provable multi-copy data possession in multi-cloud storage. IEEE Trans. Cloud Comput. 10 (1), 356–365.
Li, J., Zhang, L., Liu, J.K., Qian, H., Dong, Z., 2016. Privacy-preserving public auditing protocol for low-performance end devices in cloud. IEEE Trans. Inf. Forensics Secur. 11 (11), 2572–2583.
Li, Y., Yu, Y., Min, G., Susilo, W., Ni, J., Choo, K.-K.R., 2017. Fuzzy identity-based data integrity auditing for reliable cloud storage systems. IEEE Trans. Depend. Secure Comput. 16 (1), 72–83.
Liu, J., Huang, K., Rong, H., Wang, H., Xian, M., 2015. Privacy-preserving public auditing for regenerating-code-based cloud storage. IEEE Trans. Inf. Forensics Secur. 10 (7), 1513–1528.
Micciancio, D., Regev, O., 2006. Lattice-based cryptography. In: Proceedings of Advances in Cryptology (CRYPTO), pp. 131–141.
Nayak, S.K., Tripathy, S., 2021. SEPDP: secure and efficient privacy preserving provable data possession in cloud storage. IEEE Trans. Serv. Comput. 14 (3), 876–888.
Regev, O., 2009. On lattices, learning with errors, random linear codes, and cryptography. Proc. Annu. ACM Symp. Theory Comput. 56 (6).
Ren, Z., Wang, L., Wang, Q., Xu, M., 2018. Dynamic proofs of retrievability for coded cloud storage systems. IEEE Trans. Serv. Comput. 11 (4), 685–698.
Rivest, R.L., Shamir, A., Tauman, Y., 2001. How to leak a secret. In: International Conf. Theory Application of Crypto. Information Security (ASIACRYPT)., pp. 552–565.
Shacham, H., Waters, B., 2008. Compact proofs of retrievability. In: International Conf. the Theory Application Cryptology Inf. Security, pp. 90–107.
Shen, J., Shen, J., Chen, X., Huang, X., Susilo, W., 2017. An efficient public auditing protocol with novel dynamic structure for cloud data. IEEE Trans. Inf. Forensics Secur. 12 (10), 2402–2415.
Shen, W., Qin, J., Yu, J., Hao, R., Hu, J., 2019. Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. IEEE Trans. Inf. Forensics Secur. 14 (2), 331–346.
Shi, E., Stefanov, E., Papamanthou, C., 2013. Practical dynamic proofs of retrievability. In: Proc. 20th ACM Conf. Comput. Commun. Secur. (CCS), pp. 325–336.
Tian, J., Wang, H., Wang, M., 2021. Data integrity auditing for secure cloud storage using user behavior prediction. Comput. Secur. 105.
Wang, B., Li, B., Li, H., 2013. Panda: Public auditing for shared data with efficient user revocation in the cloud. IEEE Trans. Serv. Comput. 8 (1), 92–106.
Wang, B., Li, B., Li, H., Li, F., 2013. Certificateless public auditing for data integrity in the cloud. In: 2013 IEEE Conf. Communications Network Security, pp. 136–144.
Wang, Q., Wang, C., Ren, K., Lou, W., Li, J., 2010. Enabling public auditability and data dynamics for storage security in cloud computing. IEEE Trans. Parallel Distrib. Syst. 22 (5), 847–859.
Ward, B., Shuichi, K., Federico, P., 2020. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: International Conf. Theory Application of Crypto. Information Security (ASIACRYPT), pp. 464–492.
Xu, G., Han, S., Bai, Y., Feng, X., Gan, Y., 2021. Data tag replacement algorithm for data integrity verification in cloud storage. Comput. Secur. 103 (3), 102205.
Yang, K., Jia, X., 2013. An efficient and secure dynamic auditing protocol for data storage in cloud computing. IEEE Trans. Parallel Distrib. Syst. 24 (9), 1717–1726.
Yu, J., Hao, R., 2021. Comments on SEPDP: secure and efficient privacy preserving provable data possession in cloud storage. IEEE Trans. Serv. Comput. 14 (6), 2090–2092.
Yu, J., Ren, K., Wang, C., 2016. Enabling cloud storage auditing with verifiable outsourcing of key updates. IEEE Trans. Inf. Forensics Secur. 11 (6), 1362–1375.
Yu, Y., Au, M.H., Ateniese, G., Huang, X., Susilo, W., Dai, Y., Min, G., 2016. Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage. IEEE Trans. Inf. Forensics Secur. 12 (4), 767–778.
Zhu, Y., Ahn, G.-J., Hu, H., Yau, S.S., An, H.G., Hu, C.-J., 2013. Dynamic audit services for outsourced storages in clouds. IEEE Trans. Serv. Comput. 6 (2), 227–238.
Zhu, Y., Hu, H., Ahn, G.-J., Yu, M., 2012. Cooperative provable data possession for integrity verification in multicloud storage. IEEE Trans. Parallel Distrib. Syst. 23 (12), 2231–2244.

Miao Tian received her MS degree from the School of Computer Science and Engineering, Northwest Normal University, China, in 2019 and she is currently pursuing her PhD degree in the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. Her research interest includes multimedia security, cloud computing security, and big data security.

Yushu Zhang received the PhD degree in computer science and technology from the College of Computer Science, Chongqing University, Chongqing, China, in 2014. He held various research positions with Southwest University, Chongqing, China, City University of Hong Kong, Hong Kong, China, University of Macau, China, and Deakin University, Victoria, Australia. He is currently a Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He is currently an Associate Editor for the Information Sciences and Signal Processing. He has authored or coauthored more than 100 refereed journal articles and conference papers in these areas. His current research interests include multimedia security, blockchain, and artificial intelligence.

Youwen Zhureceived the BE and PhD degrees in computer science from the University of Science and Technology of China, Hefei, China, in 2007 and 2012, respectively. From 2012 to 2014, he was a JSPS Postdoctoral with the Kyushu University, Fukuoka, Japan. He is currently an Associate Professor with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, China. He has authored or coauthored more than 40 papers in refereed international conferences and journals, and has served as program committee member in several international conferences. His current research interests include identity authentication, information security and data privacy.

Liangmin Wang received his PhD degree in Cryptology from Xidian University, Xi'an, China. He is a full professor in the School of Cyber Science and Engineering, Southeast University, Nanjing, China. He has been honored as a "Wan-Jiang Scholar" of Anhui Province since Nov. 2013. His research interests include data security & privacy. He has published over 60 technical papers in premium journals and conferences, like IEEE/ACM TON, INFOCOM, IoT Journal, and TITS. He has served as a TPC member of many IEEE conferences, such as ICC, HPCC, TrustCom. He is an associate editor of Security and Communication Networks, a member of IEEE, ACM, and a senior member of CCF.

Yong Xiang received the PhD degree in electrical and electronic engineering from The University of Melbourne, Australia. He is currently a Professor with the School of Information Technology, Deakin University, Australia. He has published six monographs, more than 190 refereed journal articles, and numerous conference papers in these areas. His current research interests include information security and privacy, signal and image processing, data analytics and machine intelligence, the Internet of Things, and blockchain. He has served as an Honorary Chair, a Program Chair, a TPC Chair, a Symposium Chair, and a Track Chair for a number of international conferences.