electronics

MDPI

Article

Public Authentic-Replica Sampling Mechanism in Distributed Storage Environments

Jiale Ye [1], Yongmei Bai [2], Jiang Xu [1], Shitao Huang [1], Zhaoyang Han [3] and Wei Wan [4,*]

[1] School of Computer Science, School of Cyber Science and Engineering, Engineering Research Center of Digital Forensics, Ministry of Education, Nanjing University of Information Science and Technology, Nanjing 210044, China; 202212490273@nuist.edu.cn (J.Y); 001136@nuist.edu.cn (J.X.); 202312490514@nuist.edu.cn (S.H.)
[2] College of Computer Science and Technology, Shanghai University of Finance and Economics, Shanghai 200433, China; baiyongmei2024@outlook.com
[3] School of Software, Shandong University, Jinan 250100, China; nuaahanzy@gmail.com
[4] State Grid Zaozhuang Power Supply Company, Zaozhuang 277899, China
* Correspondence: wanwei@vip.163.com

Abstract: With the rapid development of wireless communication and big data analysis technologies, the storage of massive amounts of data relies on third-party trusted storage, such as cloud storage. However, once data are stored on third-party servers, data owners lose physical control over their data, making it challenging to ensure data integrity and security. To address this issue, researchers have proposed integrity auditing mechanisms that allow for the auditing of data integrity on cloud servers without retrieving all the data. To further enhance the availability of data stored on cloud servers, multiple replicas of the original data are stored on the server. However, in existing multi-replica auditing schemes, there is a problem of server fraud, where the server does not actually store the corresponding data replicas. To tackle this issue, this paper presents a formal definition of authentic replicas along with a security model for the authentic-replica sampling mechanism. Based on time-lock puzzles, identity-based encryption (IBE) mechanisms, and succinct proof techniques, we design an authentic replica auditing mechanism. This mechanism ensures the authenticity of replicas and can resist outsourcing attacks and generation attacks. Additionally, our schemes replace the combination of random numbers and replica correspondence tables with Linear Feedback Shift Registers (LFSRs), optimizing the original client-side generation and uploading of replica parameters from being linearly related to the number of replicas to a constant level. Furthermore, our schemes allow for the public recovery of replica parameters, enabling any third party to verify the replicas through these parameters. As a result, the schemes achieve public verifiability and meet the efficiency requirements for authentic-replica sampling in multi-cloud environments. This makes our scheme more suitable for distributed storage environments. The experiments show that our scheme makes the time for generating copy parameters negligible while also greatly optimizing the time required for replica generation. As the amount of replica data increases, the time spent does not grow linearly. Due to the multi-party aggregation design, the verification time is also optimal. Compared to the latest schemes, the verification time is reduced by approximately 30%.

Keywords: cloud storage; data integrity auditing; data security; data availability; multi-copy storage

check for updates

Citation: Ye, J.; Bai, Y.; Xu, J.; Huang, S.; Han, Z.; Wan, W. Public Authentic-Replica Sampling Mechanism in Distributed Storage Environments. *Electronics* **2024**, *13*, 4167. https://doi.org/10.3390/electronics13214167

Academic Editor: Carlo Mastroianni

Received: 19 September 2024
Revised: 17 October 2024
Accepted: 19 October 2024
Published: 23 October 2024

Copyright: © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

Storage as service [1–3] allows for consumption-based storage facilities to store and process massive amounts of data through cloud storage. Cloud storage relieves clients of the burden of storage management [4], simplifying data maintenance and management and reducing reliance on specialized IT personnel. Additionally, cloud storage facilitates universal data access across independent geographical locations, enabling users to easily access and share data regardless of their location. Moreover, cloud storage eliminates the

substantial costs associated with purchasing hardware, software, and personnel maintenance [5], providing a cost-effective solution that significantly reduces the total cost of ownership, enhancing operational efficiency and competitiveness for enterprises. However, cloud service providers are not entirely trustworthy, and clients need to ensure that their data stored on the servers are not subject to integrity or other security threats [6–8]. Therefore, effective mechanisms are needed to ensure data integrity in cloud storage. Integrity auditing technology is considered an effective means for clients to verify the correctness of data stored in the cloud. In 2007, Ateniese et al. [9] proposed Provable Data Possession (PDP), and Jules and Kaliski [10] introduced Proofs of Retrievability (PORs). Both methods provide clients with guarantees of the integrity and availability of outsourced data. Subsequently, more and more experts began researching how to ensure data integrity for clients in cloud storage. In schemes where clients interact with cloud storage servers [11,12], only the original data are stored, making it difficult to recover in case of damage to the original data. However, in multi-replica storage, any damaged data can be correctly recovered using other replicas [13–15], refs. [16–18], which has been widely applied in industries such as finance, education, and healthcare, where data availability is critical.

However, in multi-replica schemes, servers may still engage in fraudulent behavior, i.e., not actually storing all data replicas. This lack of assurance regarding the genuine storage of data replicas not only leads to the inability to recover damaged data but also significantly undermines the integrity and availability of the stored data, which is especially critical in industries involving key business operations. Therefore, there is an urgent need for a genuine replica sampling mechanism that can resist server fraud to ensure data security and availability, protecting user interests and trust. Moreover, we specifically define server fraud as outsourcing attacks and generation attacks [19].

In an outsourcing attack, a malicious server attempts to deceive the verifier into passing the audit. During the integrity audit challenge, the malicious server, upon receiving a challenge from the verifier, quickly retrieves the corresponding data from another storage provider and generates the proof. The malicious server appears to be storing the data continuously but, in reality, it does not bear the responsibility of storing it. This behavior allows the malicious server to deceive the client not only about the physical space allocated to the data but also about its storage capacity, even claiming to store data beyond its limit. This behavior not only affects data reliability but also imposes significant economic losses and risks of data leakage for the client. Furthermore, since the data retrieved from external storage lack security guarantees, the overall security of the data are greatly compromised, increasing the risk of data leakage and loss, further exacerbating the trust crisis.

Secondly, in a generation attack, the malicious server uses certain algorithms or predefined methods to quickly and dynamically generate replicas and produce proofs during a challenge without actually storing real data. This allows the malicious server to deceive the client, reducing the actual storage space required, thus saving costs, but posing a serious threat to data security. Therefore, to enhance data integrity and availability, effective mechanisms must be developed to defend against these fraudulent behaviors and maintain user trust and interests.

In data integrity auditing schemes, the client and the server, as two interacting entities, have different evaluation criteria to consider from their respective perspectives [15,20,21]. Existing schemes often focus on different evaluation criteria, attempting to make the scheme more favorable to either the server or the client. However, considering only one party's evaluation criteria will lead to an imbalanced scheme. Therefore, it is necessary to trade off between the client and the server, meaning that the scheme must consider how to meet the evaluation criteria of both parties. Armknecht et al. proposed a Mirror algorithm [22], which shifts the replica generation process to the server, reducing the client's computational burden and significantly saving bandwidth resources. At the same time, the server also benefits from reduced communication transmission, saving bandwidth resources. Additionally, although the server regains the initiative in replication, it is almost impossible for it to engage in improper behavior. This scheme is client-friendly while

imposing as little burden as possible on the server. However, this trade-off still faces the issue of linear storage cost increase with the number of replicas when the client generates and uploads copy parameters. Moreover, it does not meet the criteria for public verifiability compared to other schemes. In 2023, the user-friendly scheme [23] satisfies the public verifiability feature of client generation and uploading of copy parameters under a single cloud server, where the generation and uploading of copy parameters remain constant as the number of copies increases and the copy parameters are recoverable.

Although the aforementioned schemes are excellent, they have some obvious limitations. First, these schemes typically rely on a single-cloud storage environment, making the system vulnerable to single points of failure. If the cloud provider experiences an outage, data may become inaccessible or be permanently lost. Second, in a single-cloud environment, the tasks of replica generation [13,19] and storage are all handled by a single server, which not only increases the computational and storage burden but can also lead to performance bottlenecks when dealing with large-scale data. Furthermore, single-cloud schemes are less capable of defending against outsourcing attacks and generation attacks, where a malicious server can deceive the verifier by using external resources or dynamically generating replicas. In contrast, a multi-cloud environment distributes storage and processing tasks, improving the system's security and stability while also reducing verification overhead. Lastly, the public verifiability of single-cloud schemes is limited; the verifier must rely on a single cloud service provider, lacking the transparency of multi-party collaboration. Multi-cloud storage, however, can enhance the reliability of the audit process through collaboration among different cloud providers. Unfortunately, the above schemes have not been implemented in a multi-cloud environment, and both public verifiability and constant-level copy parameter construction will be called into question.

To address the aforementioned challenges, we propose a public authentic-replica sampling mechanism. This mechanism is a security sampling scheme designed to tackle data integrity and availability issues in a multi-cloud storage environment. Our primary goal is to ensure that the data replicas stored on cloud servers genuinely exist and have not been tampered with, thus maintaining the integrity and availability of the data. To achieve this goal, we designed an auditing method based on time-lock puzzles, succinct proofs, and identity-based encryption to ensure the authenticity of replicas. Additionally, recoverable copy parameters are employed, allowing any third party to verify the data, achieving public verifiability. Based on this sampling mechanism, we have instantiated a public authentic-replica sampling scheme. During the designated replica generation process, by utilizing a Ref-Table and random seeds, we achieve the distributed storage of data replicas across multiple clouds with negligible overhead while also reducing the burden on the client. The overall design cleverly allows for proof aggregation, further reducing verification overhead, resulting in a scheme that is both efficient and secure. The main contributions can be summarized as follows:

1. To address the issue of servers falsely storing replica data, this paper designs an authentic-replica sampling mechanism that periodically checks the server's replica storage. We formally define the security model for the authentic-replica sampling mechanism and design a publicly verifiable authentic-replica sampling scheme for distributed storage environments. Our scheme distributes data replicas across multiple clouds, ensuring low verification overhead while distributing the pressure of replica generation. Through security analysis, we demonstrate the security of this scheme, ensuring that the replicas stored on the server occupy actual storage space.
2. This scheme is based on time-lock puzzles and employs identity-based encryption mechanisms and succinct proof techniques to ensure the authentic replica. It is designed to resist generation attacks and outsourcing attacks, avoiding the associated security threats.
3. To address the issue of linear storage cost increase when the client generates and uploads copy parameters as the number of replicas increases, this paper combines random seeds and reference tables. This approach ensures that even as the number of

stored replicas grows, the cost for the client to generate and upload copy parameters remains constant, which greatly reduces the computational overhead on the client side. Additionally, by using publicly recoverable copy parameters, the scheme achieves public verifiability. Security analysis demonstrates the scheme's security.

We organize our paper as follows: We review the preliminaries required in our paper in Section 2. The system model and security model of the public authentic-replica sampling mechanism are described in Section 3. In Section 4, we give the detailed construction of the public authentic-replica sampling scheme and the security analysis. We give the performance of our scheme in Section 5, and finally the conclusion is in Section 6.

Related Work

Distributed storage systems combine network and storage technologies to allow clients to store data remotely and provide various services such as archiving, publishing, federation, and anonymity. As networking technology has evolved, new types of distributed storage systems have emerged [24]. These systems can be categorized into two main types of architectures: client–server and peer-to-peer. In a client–server architecture [25], each entity is either a client or a server with the server responsible for authentication, data replication, backup, and handling client requests. This architecture is commonly used in distributed storage systems. In contrast, in a peer-to-peer architecture [25,26], each participant can be both a client and a server.

In distributed storage systems [27–30], storing data is essentially a data outsourcing problem for the client. Consequently, ensuring the integrity of the data that has been outsourced is of critical importance in terms of security in order to ensure the integrity of outsourced data on cloud servers, Ateniese [9] et al. proposed the Provable Data Possession (PDP) concept and constructed the PDP scheme, which serves the purpose of verifying the integrity of remote data by checking whether the server has certain blocks through Homomorphic Verifiable Tag (HVT) in the form of probability-based sampling. Secure PDP (S-PDP) is the first sampling mechanism-based PDP, which strongly guarantees the possession of data with its added robustness but at the cost of huge computation on the client side. For this reason, Efficient PDP (E-PDP) [31] has been proposed again, which is used to reduce the computation on the client side, but they all suffer from high communication cost and a linear relationship between the client's computation and the replica. In 2013, Hanser and Slamanig provided a PDP based on the elliptic curve cryptosystem [32], which enables the same tag to be verified both publicly and privately by identifying the vectors of each data block, which in turn achieves the public verifiability of the scheme, enabling the client and third party to audit the data remotely at the same time. In 2013, Chen proposed algebraic signatures to effectively check data possession in cloud storage [33]. Algebraic signatures have less computational overhead than homomorphic cryptosystems, which makes the scheme guarantee data possession while reducing the overhead, but it suffers from an upper limit on the number of verifications and relies on probabilistic guarantees. Compared to PDP, Juels and Kaliski [10] proposed the Proofs of Retrievability (PoRs), which is a form of encrypted Proof of Knowledge (PoK) that ensures the integrity of outsourced data in untrusted cloud storage while also enabling the recoverability of slightly damaged data through the use of Forward Error-correcting Codes (FECs), but it has the limitation of having an upper limit on the number of verifications. Subsequently, Shacham and Waters achieved data retrievability through coding techniques in 2008 [34], and through homomorphic authenticators, they achieved an infinite number of integrity verifications. In 2013, Yuan and Yu designed a new PoR scheme, which generates polynomials with short strings as a constant-size polynomial commitment technique through the use of constant-size polynomials as a proof, which makes the scheme have a communication cost of constant magnitude. Subsequently, more and more researchers have devoted themselves to this topic and produced a large number of research results.

However, even PoR schemes that can have a weak recovery capability can only repair minor damage. If the source data are damaged to a greater extent, the data will not be

recovered, and at the same time, data retrievability will be damaged at the same time. In order to make up for this shortcoming, many researchers have begun to focus on replica storage. Replica [35] is used to ensure data availabilit. Basic principles are used in the server to store different replicas, so that even if the data and more than one replica are corrupted, the remaining complete replica can restore the damaged data, to a large extent, to ensure the availability of data. In the specific scheme to realize multiple replicas, the first thing that researchers think of is to make the replicas directly become additional copies of data. This approach is simpler to implement, but it does not provide strong proof that the server is actually storing multiple copies; the server can be challenged without the client submitting a sufficient number of copies, and it will create copies when challenged, which in turn achieves the effect of spoofing. Thus, the copies that the client expects the server to store become difficult to prove. These schemes also do not guarantee the authenticity, integrity, and availability of the data copies on the server. In 2008, Curtmola et al. made the first attempt at Multiple-Replica Provable Data Possession (MR-PDP) [36], in which the client is able to securely confirm that the server stores multiple unique replicas. In addition, because multiple replicas can be checked in a batch, the overhead of verifying n replicas is less than that of verifying one replica n times. It succeeds in securely networking [37] and storing multiple replicas to create uniquely distinguishable copies of data, but MR-PDP only supports private authentication. In 2009, Bowers et al. proposed a new cloud storage scheme that is based on PoR and utilizes both within-server redundancy and cross-server redundancy, allowing for the testing and reallocation of storage resources when failures are detected. In 2010, Barsoum and Hasan proposed another scheme [36]. There are two versions: one is at the cost of higher storage overhead on the client and server, and the other one is with probabilistic detection to reduce client and server overhead. This scheme is able to delegate the auditing task to a trusted third party using a BLS homomorphic linear authenticator, but the client must re-encrypt and upload the entire copy to the server. Subsequently, multi-copy integrity auditing on dynamic data updates was also proposed. In 2014, Barsoum and Hasan proposed a scheme [38], which supports multi-copy storage in addition to the dynamic operations on the number of outsourcing.

In 2016, Armknecht et al. proposed the Mirror scheme [22], which is based on the difficult problem of Linear Feedback Shift Registers (LFSRs) with Rivest, where mirroring shifts the overhead of building replicas to the cloud servers, which makes the cloud servers reduce the bandwidth overhead. Subsequently, Guo et al. improved on Mirror [39] in order to resist the replacement forgery attacks caused by the extensibility of data labels. These schemes reduce the computation of the client and make the replica generation part realized by the server; however, these schemes still have the problem that the client has a linear relationship with the copy parameters, and the client still has a relatively large amount of computation. At the same time, compared with the previous schemes, it cannot satisfy the public verifiability characteristic. In 2023, the user-friendly scheme [23] satisfies the public verifiability feature of client generation and uploading of copy parameters under a single cloud server, where the generation and uploading of copy parameters remain constant as the number of copies increases and the copy parameters are recoverable. However, if all cloud servers store one more layer of copy parameter generation in a multi-cloud environment, in this environment, the constant and copy parameter generation cannot be satisfied, and the public verifiability characteristic will be questioned. None of the existing integrity auditing schemes can resist the two attacks: the generation attack and the outsourcing attack.

Although existing schemes have made significant efforts in ensuring data integrity and availability, most are confined to single-cloud environments. In a multi-cloud environment, these schemes fail to guarantee client-friendliness and cannot achieve public verifiability. Verification is limited to a single cloud, and whether efficient verification can be achieved in a multi-cloud setting remains uncertain. In contrast, the scheme proposed in this paper overcomes these limitations. By adopting techniques such as time-lock puzzles, succinct proofs, and identity-based encryption, it ensures the authenticity and security

of replicas, effectively resisting outsourcing attacks and generation attacks. In a multi-cloud environment, our scheme significantly reduces the overhead of replica generation by combining Ref-Table and random seeds. Additionally, it introduces public verifiability in the replica verification process, greatly enhancing the security and efficiency of the system.

2. Preliminaries

In this section, we will outline the key techniques used in the solution, including time-lock puzzles and succinct proofs for hidden order groups.

2.1. Time-Lock Puzzles

A time-lock puzzle is an encryption technique used to unlock data after a specific time. The basic concept is to encrypt a message so that it can only be decrypted after a set time delay. Rivest et al. introduced an RSA-based time-lock puzzle to encrypt a message [40]; the puzzle can be solved to reveal the key or message only after a certain amount of time. Subsequently, Freitag et al. introduced Non-Malleable Time-Lock Puzzles, which not only require the solver to decrypt after a specified time but also ensure that the puzzle's solution cannot be forged. These puzzles often have a wide range of applications in areas such as electronic auctions, corporate contract signing, etc. [41–43]. Rivest's time-lock puzzle is related to the RSA cryptosystem in a way, where Alice has to encrypt the message m for T seconds, which works as follows.

1. $\text{Setup}(\lambda, k, m) \rightarrow (N, u, t)$
   - Generate two large prime numbers p, q at random.
   - Calculate the Euler function $\phi(N) = (p-1)(q-1)$ for modes $N = pq$.
   - Determine the number S of modulo N-squared operations that the solver *Bob* can perform per second and compute $t = T \cdot S$.
   - Encrypt m with secret key K.
   - Randomly select $u \leftarrow \mathbb{Z}_N^*$ and encrypt K as $C_K = K + u^{2^t} \, mod \, N$.
   - For C_K, output the time-lock puzzle (N, u, t).
2. $\text{Solve}(N, u, t) \rightarrow (y)$. When *Bob* receives the puzzle, without knowing p, q, it can only be solved for y by t successive squared mode computations.
3. $\text{Verify}(u, x, y, \phi(N)) \rightarrow \{0, 1\}$. Alice knows that $\phi(N)$, with $r = 2^t \, mod \, \phi(N)$, can obtain the y result quickly and validate the solver Bob's result y'. If the $y = y'$, output 1; otherwise, output 0.

2.2. Succinct Proofs for Hidden Order Groups

In succinct proofs for hidden order groups, there are two types of succinct proofs: one is the succinct proof of exponentiation (POE), and the other is the succinct proof of knowledge of a discrete-log (POKD).

In POE, given a group $\mathbb{G}$ of unknown order, $\mathbb{Z}_N$, and prime numbers $Primes(\lambda)$ in $\left[0, 2^\lambda\right]$, both the prover and the verifier are given the triad $(u, w, x), u, w \in \mathbb{G}, x \in \mathbb{Z}$, and it takes the prover to convince the verifier that there is a $u^x = w$ in the group $\mathbb{G}$. Define this relationship as $\mathcal{R}_{POE} = \{((u, w \in \mathbb{G}, x \in \mathbb{Z}); \perp) : w = u^x \in \mathbb{G}\}$, where x can be much larger than the order of the group $\mathbb{G}$ [44].

- $\mathcal{R}_{POE}.Prove(u, w, x, N) \rightarrow \mathcal{P}$. The prover receives the verifier's prime $N \leftarrow Primes(\lambda)$ and computes the $q = \left[\frac{x}{N}\right]$ and the r, $x = qN + r$, and it sends $\mathcal{P} \leftarrow u^q$ to the verifier.
- $\mathcal{R}_{POE}.Verify(\mathcal{P}, w, x, N) \rightarrow \{0, 1\}$. The verifier computes $r \leftarrow x \, mod \, N$ and verifies that $\mathcal{P}^N u^r = w$ holds in the group, returning 1 if it does and 0 otherwise.

POE can be adapted to provide POKD, given a group $\mathbb{G}$ of unknown order, and the generator g, $\mathbb{Z}_N$ and prime numbers $Primes(\lambda)$ in $[0, 2^\lambda]$. Define this relationship as $\mathcal{R}_{POKD} = \{(w \in \mathbb{G}, x \in \mathbb{Z}) : w = g^x \in \mathbb{G}\}$.

- $\mathcal{R}_{POKD}.Prove(w, x, N) \rightarrow \mathcal{P}$. The prover receives the verifier's prime $N \leftarrow Primes(\lambda)$, computes the $q \in \mathbb{Z}$ and the r, $x = qN + r$, and sends pair $(\mathcal{P} \leftarrow g^N, r)$ to the verifier.

- $\mathcal{R}_{POKD}.Verify\big((\mathcal{P} \leftarrow g^N, r), w, N\big) \rightarrow \{0,1\}$. The verifier verifies that $\mathcal{P}^N u^r = w$ and $r \in [N]$ holds in the group, returning 1 if it does and 0 otherwise.

3. Syetem Model and Security Model of Public Authentic-Replica Sampling Mechanism

In this section, we will introduce the specific content of the public authentic-replica sampling mechanism, including the formal definition, system model, and security model.

3.1. Notations

In Table 1, we introduce the main notations of our scheme.

Table 1. Notations.

Notation	Description
λ	Security parameter
k	Safety factor
$\mathbb{G}_1$	Multiplicative cyclic group
$\mathbb{G}_T$	Multiplicative cyclic group
p	Multiplicative cyclic group $\mathbb{G}_1$ of order
e	Bilinear pair
S_l	Cloud server
$S_{l_{Rep}}$	Collection of replicas stored in S_l
f	Pseudorandom sequence
φ	Pseudorandom function
π	Pseudorandom sequence
H_i	Hash function
C	Client
$\mathcal{N}$	Total number of replicas stored on all cloud servers
N	RSA's model
$\mathcal{T}$	Ref-Table
m_i	Coded data block
σ_i	The tag corresponding to the m_i data block
$m_i^{(S_l,j)}$	The j-th replica about data block m_i is stored in the S_l
i_{β}	Challenged data blocks
j_{ε}	Challenged replicas
C_1	Collection of challenged data blocks
$Chal$	Challenged collections
$\widetilde{R_{S_l}}$	Proof of replica generated by S_l
$\widetilde{M_{S_l}}$	Proof of data generated by S_l
$\mathbb{F}$	Original encoding file
C_{S_l}	S_l Generated ciphertext
n	Number of blocks in the original coded file cut
$name$	Unique identifier for source code files
NS_l	Unique identifier for S_l

3.2. Overview of Our System Model

The system model of our scheme is represented in Figure 1. It contains four kinds of entities: several cloud servers (CSs), a client (C), a cloud organizer (CO), and a third-party auditor (TPA).

- Client: The client, as the data owner, has a significant volume of data that needs to be stored on a cloud server.
- CS: CS is a cloud server that provides enough storage space for the client and is always ready to take on challenges.
- CO: The CO is used to aggregate all the proofs generated in the CS and sends the results of this aggregated proof to the TPA.
- TPA: The TPA accepts the client's delegation to check the integrity of the data.

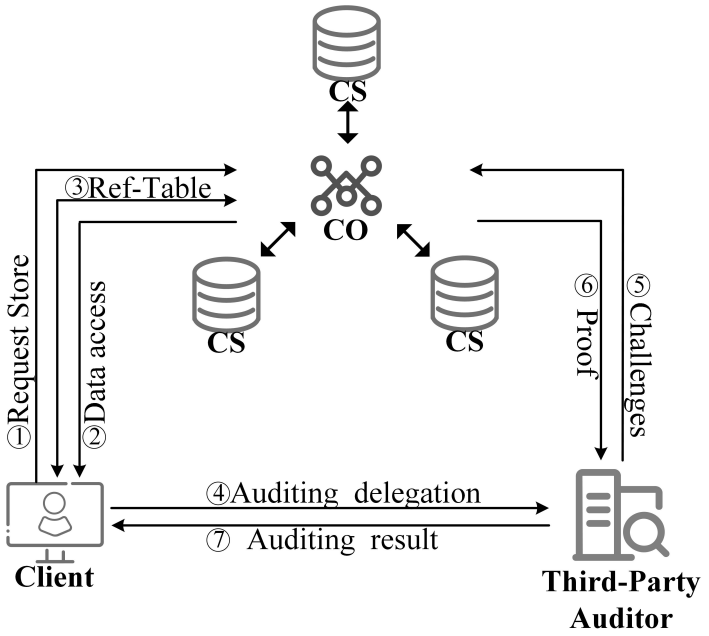


Figure 1. System model of public authentic-replica sampling mechansim.

The system works as follows.

1. Request Store: The client initiates the data outsourcing process by sending a formal request to the cloud storage (CS). The client sends the already encrypted and encoded data to the CS, requesting the storage of these data.
2. Data Access: Upon receiving the request, the cloud storage evaluates it to ensure it can meet the client's needs. Once it confirms its ability to fulfill the request, the cloud storage accepts the request and asks the client to provide the necessary parameters for storing their data.
3. Ref-Table: The client prepares a Ref-Table that details the requirements for the distribution of data replicas. This table specifies which specific data (or data blocks) should be stored in which cloud storage providers to ensure redundancy and fault tolerance. The client sends this table to the cloud operator (CO). Upon receiving the Ref-Table, the CO must relay the corresponding information to each cloud provider. Once all clouds are aware of their necessary information based on the Ref-Table, they will respond to the messages, which will ultimately be verified by the client. For the client, data storage can only be considered distributed across multiple clouds after a successful verification process.
4. Auditing Delegation: After the client ensures that its data are stored in a multi-cloud environment, it delegates the responsibility of data verification to a third-party auditor (TPA). The TPA acts as an independent entity to verify the integrity of the client's data in the multi-cloud storage.
5. Challenges: The TPA randomly selects challenges and sends them to each cloud storage. Specific data integrity challenges are directed to each cloud storage. These challenges are designed to verify whether the data are stored correctly and remain intact.
6. Proof: Each cloud storage, upon receiving the challenge, needs to generate an audit proof for its data and its replicas by utilizing the necessary parameters from the Ref-Table in conjunction with the received challenge. Each cloud storage submits its proof to the CO, which then uses an aggregation algorithm to combine all proofs into a single proof. This proof represents the collective data integrity status of all relevant cloud storages.
7. Auditing Result: After receiving the proofs, the TPA uses its verification algorithm to validate the responses from the cloud storage. Once the verification is complete,

the TPA returns the results to the client, indicating whether the data integrity has been maintained across the multi-cloud environment.

3.3. The Formal Definition of Public Authentic-Replica Sampling Mechanism

Definition 1. *The public authentic-replica sampling mechanism contains the following seven algorithms: Setup, KeyGen, Store, Table, Replicate, Challenge and Verify.*

1. $Setup(\lambda, k) \rightarrow pp$*: Input the security parameter* λ, k *to generate the public parameter* pp*.*
2. $KeyGen(pp) \rightarrow \{(sk_c, pk_c), (sk_{S_l}, pk_{S_l})\}$*: Input the public parameter* pp *to generate the public–private key pair* (sk_c, pk_c) *for the client, and for each server* S_l*, generate the public–private key pair* (sk_{S_l}, pk_{S_l})*; thus, generate all the server public–private key pair records as* $((sk_{S_l}, pk_{S_l}))$*.*
3. $Store(\mathbb{F}, name, sk_c, pk_c, pp) \rightarrow (\{\sigma_i\}, \Gamma, \mathcal{T})$*: Input source coded file* $\mathbb{F}$*, source coded file unique identifier* $name$*, client private–public key pair* (sk_c, pk_c) *and public parameter* pp*, preprocessing for the source coded file* $\mathbb{F}$*, output the tag* σ_i *corresponding to the data block* m_i*, the file tag* Γ *and the server Ref-Table* $\mathcal{T}$*.*
4. $Table(\{S_l\}, \{ID_{S_l}\}, c, ID_c, pp, \mathcal{T}) \rightarrow \{0, 1\}$*: The client decides on the replica allocation, inputs the Ref-Table* $\mathcal{T}$*, encrypts and transmits the number of replicas required to be stored by the corresponding cloud server, and the corresponding server decrypts them, encrypts them again, and hands them over to the client for verification. Only after the client has verified is the Ref-Table made public.*
5. $Replicate(\mathbb{F}, \mathcal{N}, \xi, sk_{S_l}, pp, \mathcal{T}, NS_l) \rightarrow \{m_i^{(S_l,j)}\}$*: Each server* S_l *is required to execute this* $Replicate$ *with input source encoding file* $\mathbb{F}$*, replica number* $\mathcal{N}$ *random seed* ξ *sent by the client, Ref-Table* $\mathcal{T}$*, and the server's own secret key* sk_{S_l}*, the public parameter pp, the server's own unique identifier* NS_l*. Generate the replicas that the server should store by itself* $\{m_i^{(S_l,j)}\}$*.*
6. $Challenge\left(\{Chal_{S_l}\}, m_i, \sigma_i, \{m_i^{(Rep)}\}, pp\right) \rightarrow (P)$*: Each server* S_l *will be challenged with its respective corresponding challenge* $Chal_{S_l}$*, the challenge data block, the tag corresponding to the data block, the corresponding replica stored by that server and the public parameters. For the challenge, the proofs are output, and finally the proofs from the server are handed over to the TPA for aggregation, and the TPA hands over the aggregated proof P to the client.*
7. $Verify(Chal, P, pk_c, pk_{s_l}, pp) \rightarrow \{0, 1\}$*: After receiving the final proof P, the client verifies it using its own public key. If the verification is successful, it outputs 1; otherwise, it outputs 0.*

3.4. Serurity Model

To achieve the security goal, consider a game between an adversary $\mathcal{A}$ and a simulation environment E where the environment E acts as an honest client, an honest verifier, and a partially honest server:

1. Environment E runs KeyGen to generate the key pair $(sk_c,\ pk_c)$ and gives pk_c to the adversary.
2. Adversary $\mathcal{A}$ selects the encoded file $\mathbb{F}$ while interacting with environment E to store the encoded file with the store random oracle. Environment E performs the *Store* algorithm and returns the $\{\sigma_i\}$ to adversary $\mathcal{A}$.
3. Environment E executes the Table algorithm and returns the Ref-Table $\mathcal{T}_A$ to the adversary $\mathcal{A}$.
4. The adversary $\mathcal{A}$ runs Replicate to generate and store the $\mathcal{T}_A$ replicas$\{m_i^{(\mathcal{A},j)}\}$ generated by file $\mathbb{F}$, while the environment E itself honestly generates and stores $\mathcal{N}$ - $\mathcal{T}_A$ replicas $\{m_i^{(E,j)}\}$.
5. For any encoded file $\mathbb{F}$ block for which the adversary $\mathcal{A}$ has performed a store query, the adversary $\mathcal{A}$ executes the Challenge algorithm to generate proofs for its own stored replica, while the environment E executes the Challenge algorithm to generate honest proofs for all remaining replicas. Both the proof of the adversary $\mathcal{A}$ and the honest proof generated by environment E is handed over to environment E to execute

the Verify algorithm for verification. After each execution of the protocol, environment E hands over the results of each verification to adversary $\mathcal{A}$.

6. Adversary $\mathcal{A}$ selects the file obtained from the return of some store random oracle and outputs a description of the storage service provider.

For a cheating server S_c, we say that a cheating server is ϵ-admissible if the cheating server is able to complete integrity verification convincingly with probability no less than ϵ, where

$$\Pr\left[\begin{array}{l} m_i^{(S_c,j)} \leftarrow S_c.Replicate(\mathcal{T}_{S_c}) \\ \wedge\, P_{S_c} \leftarrow S_c.Challenge(\mathcal{T}_{S_c}, \mathbb{F}_i) \\ \wedge\, Verify(P_{s_c} \cup P_{trust}) = 1 \end{array} \middle| \; Condition \right] \geq \epsilon \tag{1}$$

where the condition is

$$Condition = \begin{cases} (sk_c, pk_c) \leftarrow \mathrm{KeyGen}(pp), \\ oracle \leftarrow S_c(\mathbb{F}) \\ \mathcal{T}_{S_c} \leftarrow Table \\ m_i^{(Trust,j)} \leftarrow Replicate(\mathcal{N} - \mathcal{T}_{S_c}) \\ \mathbb{F}_i \in oracle \\ P_{trust} \leftarrow Challenge(\mathcal{N} - \mathcal{T}_{S_c}, \mathbb{F}_i) \end{cases} \tag{2}$$

Definition 2 (**Authentic Replica**). *We introduce the authentic-replica property, which allows the prover P to commit to storing n distinct replicas of the data D, all of which have actual storage space, and for the prover to be able to give a proof that convinces the verifier.*

We formalize two challenges: outsourcing attacks and generation attacks. An authentic-replica sampling mechanism design scheme that can overcome both attacks will distinguish itself from other integrity design schemes.

Given a multi-copy integrity auditing algorithm PoS, an honest server provider SP, a malicious adversary $\mathcal{A}$, an honest verifier $\mathcal{V}$, and specific data D, assume that the adversary $\mathcal{A}$ has the following data storage capacity:

- *The adversary $\mathcal{A}$ is only able to truthfully physically store the number of copies of n_1.*
- *The adversary $\mathcal{A}$ is able to interact with an honest server provider SP where SP has unlimited storage space.*

1. *The honest verifier $\mathcal{V}$ generates n copies $(n > n_1)$ by means of the PoS algorithm, which are handed over to the adversary $\mathcal{A}$ for storage. (Here, in order to ensure that these n copies are real, they are generated by an honest verifier.)*
2. *Honest validator $\mathcal{V}$ for generating a series of challenges c_i to hand over to the adversary $\mathcal{A}$.*
3. *The adversary $\mathcal{A}$ generates a series of proofs for these challenges π_i.*
4. *An honest verifier $\mathcal{V}$ verifies a series of challenges generated by an adversary $\mathcal{A}$ and gives the verification results.*
5. *If there is $P[\mathcal{V}.PosVerify(\pi_i) = 1 | \pi_i = \mathcal{A}.PosProve(c_i)] > 1 - \xi$, then adversary $\mathcal{A}$ wins the game; otherwise, adversary $\mathcal{A}$ loses.*

If no adversary can win the game with non-negligible probability, the scheme is characterized by the authentic-replica property.

Definition 3 (**Extractability**). *If any probabilistic polynomial-time (PPT) adversary plays the aforementioned game and manages to successfully output an S_c, that achieves ϵ-admissible. It can be demonstrated that an extraction algorithm exists which, with overwhelming probability, can recover the challenged data block.*

Definition 4 (**Storage allocation**). *For any rational adversary can successfully output a service provider in the above game that is ϵ-acceptable and satisfies authentic-replica (η), specifically, that allocates replica storage for only an extreme η ratio of the source data, where $0 < \eta < 1$, with a*

negligible probability of acceptance by the verifier, then we say that the protocol is (η, ϵ)*-storage allocation satisfying.*

Definition 5 (**Detectability**). *If the probability of detecting the corruption of all stored data is no less than* ψ *when the corrupted fraction of the source data is* ζ_0 *and the corrupted fraction of each replica is* ζ_j*, then we say that the protocol is* $(\zeta_0, \zeta_1, \cdots, \zeta_t, \psi)$ *detectable.*

Definition 6. *If the q-DABDHE problem is hard, then the Ref-Table transmission process is secure under our scheme.*

Definition 7 (**Outsourcing attack**). *Upon receiving the challenge c, adversary* $\mathcal{A}$ *quickly obtains the corresponding data D from another storage provider* P^* *and produces a proof, thus giving the illusion that adversary* $\mathcal{A}$ *itself actually stores the data D.*

Definition 8 (**Generation attack**). *If adversary* $\mathcal{A}$ *is able to identify the data D that it needs to store,* $\mathcal{A}$ *can lock that data D, allowing* $\mathcal{A}$ *to regenerate D on demand. On receiving a challenge c,* $\mathcal{A}$ *generates data D on demand and then generates a proof from that newly generated data D, thus giving the illusion that adversary* $\mathcal{A}$ *itself actually stores the data D.*

4. Our Proposed Scheme

In this section, we have provided specific implementation steps for the public authentic-replica sampling scheme and demonstrated its security through security analysis.

4.1. Construction of the Public Authentic-Replica Sampling Scheme

The scheme is as follows:

- $Setup(\lambda, k) \rightarrow pp$
  KGC generates an RSA modulus $N = q_0 \cdot q_1, q_0 = 2q_0' + 1, q_1 = 2q_1' + 1, q_0, q_1$ containing large prime numbers known only to *C*. The unique cyclic group $\mathbb{QR}_N$ of order $q_0' q_1' \in \mathbb{Z}_N^*$, obtains u_0 from $\mathbb{Z}_N^*$, where $gcd(u_0 \pm 1, N) = 1$, and obtains the generating element $u = u_0^2$ of the group. Choose the multiplicative cyclic group $\mathbb{G}_1$,$\mathbb{G}_T$, with prime p, the order of $\mathbb{G}_1$, g, the generator of $\mathbb{G}_1$, and $e: \mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$. Choose the following hash function, two pseudorandom sequences, and a pseudorandom function to randomly select $\overline{\alpha}, \overline{\beta}_1, \overline{\beta}_2, \overline{\beta}_3 \in \mathbb{Z}_p$, and compute $mpk = (g_1, g_2, g_3, g_4) = (g^{\overline{\alpha}}, g^{\overline{\beta}_1}, g^{\overline{\beta}_2}, g^{\overline{\beta}_3}), msk = (\overline{\alpha}, \overline{\beta}_1, \overline{\beta}_2, \overline{\beta}_3)$.

$$\begin{cases} H_0{:}\{0,1\}^* \rightarrow \mathbb{Z}_p \\ H_1{:}\{0,1\}^* \rightarrow \mathbb{QR}_N \\ H_2{:}\{1,2,\cdots,n\} \rightarrow \{0,1\}^\kappa \\ H_3{:}\{0,1\}^* \rightarrow \mathsf{Prm}(2^\lambda) \\ H_4{:}\{0,1\}^* \rightarrow \mathsf{Prm}(2^\lambda) \\ f{:}\mathbb{Z}_p \times \{1,2,\cdots,n\} \rightarrow \mathbb{Z}_p \\ \varphi{:}\mathbb{Z}_p \times \mathbb{Z}_p \rightarrow \mathbb{Z}_p \\ \pi{:}\mathbb{Z}_p \times \{1,2,\cdots,n\} \rightarrow \mathbb{Z}_p \end{cases} \tag{3}$$

  Obtain public parameters pp, where
  $pp = \{N, u, \mathbb{G}_1, \mathbb{G}_T, g, \{H_i\}, f, \varphi, \pi, mpk\}, i \in [0,4]$.

- $KeyGen(pp, msk, mpk) \rightarrow ((sk_c, pk_c), \{sk_s, pk_s\})$
  C puts his identity $ID_c \in \{0,1\}^*$ to KGC; KGC takes C's identity, master key, and public parameters as input and randomly selects three random numbers $r_1, r_2, r_3 \in \mathbb{Z}_p$ to compute C's partial private key sk_{pc}, $sk_{pc} = \left(r_1, g^{\frac{(\bar{\beta}_1 - r_1)}{(\bar{\alpha} - ID_c)}}, r_2, g^{\frac{(\bar{\beta}_2 - r_2)}{(\bar{\alpha} - ID_c)}}, r_3, g^{\frac{(\bar{\beta}_3 - r_3)}{(\bar{\alpha} - ID_c)}} \right)$. Similarly, for each server S_l, its partial private key is also calculated based on its identity ID_{S_l}, $sk_{pS_l} = \left(r_1^{S_l}, g^{\frac{\left(\bar{\beta}_1 - r_1^{S_l}\right)}{\left(\bar{\alpha} - ID_{S_l}\right)}}, r_2^{S_l}, g^{\frac{\left(\bar{\beta}_2 - r_2^{S_l}\right)}{\left(\bar{\alpha} - ID_{S_l}\right)}}, r_3^{S_l}, g^{\frac{\left(\bar{\beta}_3 - r_3^{S_l}\right)}{\left(\bar{\alpha} - ID_{S_l}\right)}} \right)$. C picks a large prime number α and calculates $\alpha' = \alpha^{-1} \ mod \ q_0' q_1'$; $sk_c = (sk_{c1} = \alpha', sk_{pc})$, $pk_c = (pk_{c1} = \alpha, ID_c)$. For each store on S_l, choose $\beta_{S_l 1}, \beta_{S_l 2} \in \mathbb{Z}_p$ as the private key $(sk_{S_l 1}, sk_{S_l 2})$ and compute the public key $(g^{\beta_{S_l 1}}, g^{\beta_{S_l 2}})$. S_l has $sk_{S_l} = \left(\left(sk_{S_l 1}, sk_{S_l 2} \right), sk_{pS_l} \right)$, $pk_{S_l} = \left(pk_{S_l 1}, pk_{S_l 2}, ID_{S_l} \right)$
- $Store(\mathbb{F}, name, sk_c, pk_c, pp) \rightarrow (\{\sigma_i\}, \Gamma, \mathcal{T})$

$$\begin{cases} \gamma = H_4(sk_{c1} || name) \\ \gamma' = \gamma^{-1} \bmod q_0' q_1' \\ \hat{\gamma} = \gamma \cdot pk_{c1} \bmod q_0' q_1' \end{cases} \tag{4}$$

Calculate the file tag $\Gamma = (\gamma, \gamma')$, calculate to obtain the tag corresponding to data block m_i, $\sigma_i = (H_1(name \mid i))^{\gamma'} \cdot {m_i}^{sk_{c1}}$. Given a Ref-Table $\mathcal{T}$, $\mathcal{T}$ denotes the delegated replica servers and their delegated replicas:

$$\mathcal{T} = \left(S_{1_{Rep}}, S_{2_{Rep}}, S_{3_{Rep}}, ..., S_{k_{Rep}} \right)^T = \begin{bmatrix} 1, 2, 4 \\ 3, 5 \\ ... \\ ... \end{bmatrix} \tag{5}$$

for a total of $\mathcal{N}$ replicas and k servers

$$\sum_{i=1}^{k} S_{i_{Rep}} = \mathcal{N} \tag{6}$$

- $Table.Enr(\{S_l\}, \{ID_{S_l}\}, c, ID_c, pp, \mathcal{T}) \rightarrow \{0,1\}$. For each server $S_l, l \in [1,k]$, the number of replicas commissioned as well as the original encoded file need to be stored according to $\mathcal{T}$. By using IBE, client C will ensure that the Ref-Table is known to the individual service providers.
  - $C.Enr\left(\mathcal{T}_{S_l}, ID_{S_l}, s, pp\right) \rightarrow CT_{S_l}$. Client C chooses a random number $s \in \mathbb{Z}_p$, $H_{sig} : \{0,1\}^* \rightarrow \mathbb{G}_1$. Calculate $CT_{S_l} = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3 \, \mathbb{C}_4)$, $(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3) = \left((g_1 g^{-ID_{S_l}})^s, e(g,g)^s, e(g_4, g)^s \cdot \mathcal{T}_{S_l} \right)$, and $\mathbb{C}_4 = e(g_2, g)^s \cdot e(g_3, g)^{sw}$, where $w = H_{sig}(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3)$.
  - $C.Dec\left(CT_{S_l}, sk_{p_{S_l}}, pp\right) \rightarrow \mathcal{T}_{S_l}$. When the server receives the ciphertext of the Ref-Table, it decrypts the plaintext of the Ref-Table based on a portion of its own partial private key and obtains its own Ref-Table that it should need to store.

$$\mathcal{T}_{S_l} = \frac{\mathbb{C}_3}{e(\mathbb{C}_1, g^{\frac{\left(\bar{\beta}_3 - r_3^{s_l}\right)}{\left(\bar{\alpha} - ID_{S_l}\right)}}) \cdot \mathbb{C}_2^{g^{\frac{\left(\bar{\beta}_3 - r_3^{s_l}\right)}{\left(\bar{\alpha} - ID_{S_l}\right)}}}} = \frac{e(g_4, g)^s \cdot \mathcal{T}_{S_l}}{e\left(g^{\left(\bar{\alpha} - ID_{S_l}\right) \cdot s}, g^{\frac{\bar{\beta}_3 - r_3^{S_l}}{\bar{\alpha} - ID_{S_l}}}\right) \cdot e(g, g)^{s \cdot r_3^{S_l}}} \tag{7}$$

And the server needs to convince the client that it already knows which copies it needs to store and for which purpose. It also needs to encrypt the transmission of its own results:

- $S.Enr\left(\mathcal{T}_{S_l}, ID_c, s, pp\right) \rightarrow CT_{S_l}$;
- $S.Dec\left(CT_{S_l}, sk_{p_c}, pp\right) \rightarrow \mathcal{T}_{S_l}$.

After both the client and the server have confirmed completion, if the Ref-Table was accepted correctly, output 1 and expose the Ref-Table $\mathcal{T}$; otherwise, output 0.

- $Replicate(\mathbb{F}, \mathcal{N}, \xi, sk_{S_l}, pp, \mathcal{T}, S_l, pk_c) \rightarrow \{m_i^{(S_l,j)}\}$

For each replica $j \in [1, \mathcal{N}] \cap S_{l_{Rep}}$, obtain the random seed ξ and compute

$$\begin{cases} r_{S_l,j} = f_{\xi}(H_0(name||j||NS_l)) \\ \dot{r}_{S_l,j} = \frac{sk_{S_l 2} - r_{S_l,j}}{H_0(name||r_{S_l,j})} - sk_{S_l 1} \\ R_{S_l,j} = g^{r_{S_l,j}} \end{cases} \tag{8}$$

Then, each data block m_i is encrypted to generate a replica $m_i^{(S_l,j)}$ based on an RSA time-lock puzzle.

$$\begin{cases} \mu_{S_l,i,j} = \mu_{S_l,i-1,j}^{\prod_{\theta \in \vartheta_i} H_2(\theta||R_{S_l,j})}, \mu = \mu_{S_l,0,j} \\ m_i^{(S_l,j)} = m_i \cdot \mu_{S_l,i,j} \end{cases} \tag{9}$$

where ϑ_i

$$\vartheta_i = \begin{cases} [i, i+\delta] & i \in [1, n-\delta] \\ [1, n] - [i, i+\delta] & i \in [n-\delta+1, n] \end{cases} \tag{10}$$

- $Challenge\left(Chal, m_i, \sigma_i, \{m_i^{(S_l,j)}\}, pp\right) \rightarrow P$
  - $Challenge\left(Chal_{S_l}, m_i, \sigma_i, \{m_i^{(S_l,j)}\}, pp\right) \rightarrow P_{S_l}$

  TPA randomly selects two values $rs_1, rs_2 \in \mathbb{Z}_q$, where rs_1 is the random seed of the π and rs_2 is the random seed of the ϕ. Together with the c challenge block numbers to be checked, the TPA sends $Chal = (\{c, cp\}, rs_1, rs_2)$; C searches the Ref-Table $\mathcal{T}$ and sends the challenge $Chal_{S_l}$ to the corresponding S_l.

  Upon receiving $Chal_{S_l}$, the server S_l responds to the query block m_i and its corresponding tag σ_i as well as the copy $m_i^{(S_l,j)}$ as follows:

  1. S_l generates its own challenge set; after accepting the challenge, $Chal_{S_l}$ generates the challenge set $C_{S_l} = (C_1, C_{S_l 2}) = \left(\left\{\left(i_{\dot{\beta}}, v_{\dot{\beta}}\right)\right\}, \{j_{\varepsilon}\}\right)$, $i = \pi(rs_1, \dot{\beta})$, $v_{\dot{\beta}} = \phi(rs_2, \dot{\beta}), j_{\varepsilon} = \pi(rs_1, \varepsilon), \dot{\beta} \in [1, c], \varepsilon \in [1, cp] \cap S_{l_{Rep}}, C = (C_1, C_2 = \sum_{l=1}^{k} C_{S_l 2})$

2. S_l computes data proofs $\widetilde{M_{S_l}}$, tag proofs $\widetilde{\sigma_{S_l}}$ and replica proofs $\widetilde{R_{S_l}}$, where $j_\varepsilon \in ([1, cp] \cap S_{l_{Rep}})$.

$$\begin{cases} \widetilde{M}_{S_l} = \prod_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} m_{i_{\dot{\beta}}}^{v_{\dot{\beta}}} \\ \widetilde{\sigma}_{S_l} = \prod_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} \sigma_{i_{\dot{\beta}}}^{v_{\dot{\beta}}} \\ \widetilde{R}_{S_l} = \prod_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} \left(\prod_{j_\varepsilon} m_i^{(S_l, j_\varepsilon)} \right)^{v_{\dot{\beta}}} \end{cases} \tag{11}$$

3. S_l generates itself honestly and correctly generates copies of the proof U_{S_l}

$$\begin{cases} U_{S_l} & = \frac{\widetilde{R_{S_l}}}{\widetilde{M_{S_l}}^{|[1,cp] \cap S_{l_{Rep}}|}} \\ \widetilde{v_{S_l}} & = \sum_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} \left(v_{\dot{\beta}} \sum_{j_\varepsilon \in ([1,cp] \cap S_{l_{Rep}})} \left(\prod_{\dot{k} \in [1, i_{\dot{\beta}}]} \prod_{\theta \in \vartheta_{\dot{k}}} H_2(\theta \| R_{S_l, j_\varepsilon}) \right) \right) \end{cases} \tag{12}$$

– $Aggressive(\{P_{S_l}\}, \mathcal{T}, pp) \rightarrow P_{agg}$

1. CO verifies all the $\widetilde{\sigma_{S_l}}$, and if successful, it aggregates all the proofs of S_l. Since all the $\widetilde{\sigma_{S_l}}$ are challenged for the same chunks of data, it stands to reason that all of the $\widetilde{\sigma_{S_l}}$ would have the same result.

$$\begin{cases} \tilde{\sigma} = \sigma_{S_l}, l \in [1, k] \\ U = \prod_{l=1}^{k} U_{S_l} \\ \hat{v} = \sum_{l=1}^{k} v_{S_l} \\ \tau \leftarrow H_3(\{v_i\}_{i \in c}, u) \\ \omega \leftarrow [\tilde{v}/n] \\ \Omega = u^{\omega} \\ \tilde{M} = \prod_{l=1}^{k} \widetilde{M_{S_l}} \\ \bar{R} = \prod_{l=1}^{k} \widetilde{R_{S_l}} \end{cases} \tag{13}$$

2. Give the aggregated proof $P = \left(\widetilde{M}, \widetilde{\sigma}, \widetilde{R}, U, \Omega \right)$ to V, and let V validate the integrity verification of the file $\mathbb{F}$ and the copy.

- $Verify(Chal, P, pk_c, pk_{s_l}, pp) \rightarrow \{0, 1\}$

1. The verifier recovers for each replica the corresponding $R_{S_l, j}$

$$R_{S_l, j} = pk_{S_l 1}^{-1} \cdot (pk_{S_l 2} \cdot g^{-r_{S_l, j}})^{\frac{1}{H_0(name || r_{S_l, j})}} \tag{14}$$

2. Generation of necessary parameters

$$\begin{cases} \tilde{v} = \sum_{l=1}^{\dot{k}} \sum_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} v_{\dot{\beta}} \cdot \sum_{j_\varepsilon \in ([1,cp] \cap S_{l_{Rep}})} \left(\prod_{\dot{k} \in [1, i_{\dot{\beta}}]} \prod_{\theta \in \vartheta_{\dot{k}}} H_2(\theta || R_{S_{l,j_\varepsilon}}) \right) \\ \tau \leftarrow H_3\left(\{v_i\}_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1}, u \right) \\ w \leftarrow \tilde{v} \bmod \tau \end{cases} \tag{15}$$

3. Verify

$$\begin{cases} U = \Omega^{\tau} \cdot u^{w} \\ \tilde{R} = \tilde{M} \cdot U \\ \tilde{\sigma}^{\gamma} = \left(\prod_{(i_{\dot{\beta}}, v_{\dot{\beta}}) \in C_1} H_1(name || i_{\dot{\beta}})^{v_i} \right)^{pk_{c1}} \cdot \tilde{M}^{\gamma} \end{cases} \tag{16}$$

4.2. Security Analysis

Theorem 1. *The proof can pass final validation if all participants honestly follow the specified process.*

Proof. Proof 1: $U = \Omega^{\tau} \cdot u^{w}$

$$
\begin{aligned}
U &= \prod_{l=1}^{k} U_{S_l} = \prod_{l=1}^{k} \frac{\widetilde{R_{S_l}}}{\widetilde{M_{S_l}}^{\left|[1,cp]\cap S_{l_{Rep}}\right|}} \\
&= \prod_{l=1}^{k} \frac{\widetilde{R_{S_l}}}{\prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left(m_{i_{\dot{\beta}}}^{\nu_{\dot{\beta}}}\right)^{\left|[1,cp]\cap S_{l_{Rep}}\right|}} \\
&= \prod_{l=1}^{k} \frac{\prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} \left(\prod_{j_\varepsilon\in\left([1,cp]\cap S_{l_{Rep}}\right)} m_{i_{\dot{\beta}}}^{(S_{l_{Rep}},j_\varepsilon)}\right)^{\nu_{\dot{\beta}}}}{\prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} \left(m_{i_{\dot{\beta}}}^{\nu_{\dot{\beta}}}\right)^{\left|[1,cp]\cap S_{l_{Rep}}\right|}} \\
&= \prod_{l=1}^{k} \frac{\prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left(\prod_{j_\varepsilon\in\left([1,cp]\cap S_{l_{Rep}}\right)} \left(m_{i_{\dot{\beta}}} \cdot \mu_{S_l,i_{\dot{\beta}},j_\varepsilon}\right)\right)^{\nu_{\dot{\beta}}}}{\prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left(m_{i_{\dot{\beta}}}^{\nu_{\dot{\beta}}}\right)^{\left|[1,cp]\cap S_{l_{Rep}}\right|}} \\
&= \prod_{l=1}^{k} \prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left(\prod_{j_\varepsilon\in\left([1,cp]\cap S_{l_{Rep}}\right)} \left(\mu_{S_l,i_{\dot{\beta}},j_\varepsilon}\right)\right)^{\nu_{\dot{\beta}}} \\
&= \mu^{\widetilde{v}} \\
&= \Omega^{\tau} \cdot u^{w}
\end{aligned}
$$

Proof 2: $\tilde{R} = \tilde{M} \cdot U$

$$
\begin{aligned}
\widetilde{R} &= \prod_{l=1}^{k} \widetilde{R_{S_l}} \\
&= \prod_{l=1}^{k} \prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left(\prod_{j_\varepsilon\in\left([1,cp]\cap S_{l_{Rep}}\right)} m_{i_{\dot{\beta}}}^{(S_l,j_\varepsilon)}\right)^{\nu_{\dot{\beta}}} \\
&= \prod_{l=1}^{k} \prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} \left(\prod_{j_\varepsilon\in\left([1,cp]\cap S_{l_{Rep}}\right)} m_{i_{\dot{\beta}}} \cdot \mu_{S_l,i_{\dot{\beta}},j_\varepsilon}\right)^{\nu_{\dot{\beta}}} \\
&= \widetilde{M} \cdot \prod_{l=1}^{k} \mu^{\nu_{S_l}} = \widetilde{M} \cdot \mu^{\widetilde{v}} = \widetilde{M} \cdot U
\end{aligned}
$$

Proof 3: $\tilde{\sigma}^{\widehat{\gamma}} = \left(\prod_{(i_\beta,\nu_\beta)\in C_1} H_1(name||i_\beta)^{v_i}\right)^{pk_c} \cdot \widetilde{M}^{\gamma}$

$$\begin{aligned}\tilde{\sigma}^{\widehat{\gamma}} &= \left(\prod_{(i_\beta,\nu_\beta)\in C_1} \left(\sigma_{i_\beta}^{\nu_\beta}\right)\right)^{\widehat{\gamma}} \\ &= \left(\prod_{(i_{\dot{\beta}},\nu_{\dot{\beta}})\in C_1} \left((H_1(name||i_{\dot{\beta}}))^{\gamma'} \cdot m_{i_{\dot{\beta}}}^{sk_{c1}}\right)^{\nu_{\dot{\beta}}}\right)^{\widehat{\gamma}} \\ &= \left(\prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} H_1(name||i_{\dot{\beta}})^{v_{\dot{\beta}}}\right)^{pk_{c1}} \cdot \widetilde{M}^{\gamma}\end{aligned}$$

□

Theorem 2. *The protocol ensures extractability under the RSA assumption in the random oracle model.*

Proof. First, we assume the existence of an adversary $\mathcal{A}$ capable of breaking the extractability of the protocol. We then construct a verifier B that interacts with adversary $\mathcal{A}$, leveraging the assumption that $\mathcal{A}$ can indeed compromise extractability.

- Preliminary stage: Specify the large prime $y \leftarrow \mathbb{Z}_N^*$ and α to generate the RSA instance (N, α, y).
- Setup stage: B calculates $a = y^2 \ mod \ N$ and sends (N, α) to the adversary $\mathcal{A}$. Meanwhile, B itself calculates $\gamma = H_4(\alpha'||name), \alpha' = \alpha^{-1} mod q_0' q_1'$ and with $\gamma' = \gamma^{-1} mod \ q_0' q_1'$ as the key.
- Query phase: Adversary $\mathcal{A}$ adaptively queries B in this phase with the only restriction that $\mathcal{A}$ can never query individual blocks with the same index i. B honestly responds as follows:
  1. When B receives a query for σ_i, if there is no record for σ_i in B's record table, it generates $\theta_i \xleftarrow{R} \mathbb{QR}_N$ as σ_i and returns it to $\mathcal{A}$. It also records in the record table $(m_i, i, \theta_i, \sigma_i)$.
  2. When B receives a query for σ_i, it returns σ_i directly if there is already a record for σ_i in B's record table.
  3. When B receives a query for $H_1(name||i)$, if there is no record for $H_1(name||i)$ in B's record table, generate $H_1(name||i) = \left(\frac{\theta_i}{m_i{}^{\alpha}}\right)^{\gamma}$ as $H_1(name||i)$ and return it to $\mathcal{A}$. It also records $(name, i, H_1(name||i))$ in the record table.
  4. When B receives a query for $H_1(name||i)$, return it to $\mathcal{A}$ if there is already a record for $H_1(name||i)$ in B's record table.
- Challenge Phase: B generates a challenge set for the data block and data copy to be challenged: $C_{\mathcal{A}} = (C_1, C_{\mathcal{A}2}) = (\{(i_{\dot{\beta}}, v_{\dot{\beta}}\}, \{j_\varepsilon\}), i = \pi(rs_1, \dot{\beta}), v_{\dot{\beta}} = \phi(rs_2, \dot{\beta}), j_\varepsilon = \pi(rs_1, \varepsilon), \dot{\beta} \in [1, c], \varepsilon \in [1, cp] \cap \mathcal{A}_{Rep}$. And give $\mathcal{A}$ the challenge set $C_{\mathcal{A}}$.
- Forge phase: Based on the challenge given by B, $\mathcal{A}$ generates its own proof $P_{\mathcal{A}}^* = (\widetilde{M}_{\mathcal{A}}^*, \widetilde{\sigma}_{\mathcal{A}}^*, \widetilde{R}_{\mathcal{A}}^*, U_{\mathcal{A}}^*)$. At the same time, we have other honest provers for generating proofs of their own challenged data blocks and data replicas: $\{P_{or}\}_{other} = \{(\widetilde{M}_{or}, \widetilde{\sigma}_{or}, \widetilde{R}_{or}, \widetilde{U}_{or})\}$. Also, record a special honest prover that is challenged in the same way as adversary $\mathcal{A}$. We record the proof generated by this special honest prover as $P_{uni} = (\widetilde{M}_{uni}, \widetilde{\sigma}_{uni}, \widetilde{R}_{uni}, \widetilde{U}_{uni})$. We determine, by assumption, that $P_{\mathcal{A}}^* + \{P_{or}\}_{other}$ is able to pass the verification after undergoing aggregation. At the same time, based on the correctness of the protocol, we are able to obtain that the final proof generated as a result of the aggregation by $\{P_{or}\}_{other} + P_{uni}$ must also be able to pass validation given that we have the assumption that the adversary $\mathcal{A}$ is able to

similarly pass validation. Then, comparing the adversary $\mathcal{A}$ and the special honesty prover *uni*, both of the following should hold:

$$\tilde{\sigma}_{uni}\hat{\gamma} = \left(\prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} H_1(name||i_{\dot{\beta}})^{v_{\dot{\beta}}}\right)^{\alpha} \cdot \tilde{M}_{uni}^{\gamma}$$

$$\tilde{\sigma_{\mathcal{A}}^{*}}\hat{\gamma} = \left(\prod_{(i_{\dot{\beta}},v_{\dot{\beta}})\in C_1} H_1(name||i_{\dot{\beta}})^{v_{\dot{\beta}}}\right)^{\alpha} \cdot \widetilde{M}_{\mathcal{A}}^{*\gamma}$$

So there we have it:

$$\left(\frac{\tilde{\sigma}_{uni}}{\tilde{\sigma}_{\mathcal{A}}^{*}}\right)^{\widehat{\gamma}} = \frac{\widetilde{M}_{uni}^{\gamma}}{\widetilde{M}_{\mathcal{A}}^{*\gamma}}$$

simplifying gives the following:

$$\left(\frac{\tilde{\sigma}_{uni}}{\tilde{\sigma}_{\mathcal{A}}^{*}}\right)^{\alpha} = \frac{\widetilde{M}_{uni}}{\widetilde{M}_{\mathcal{A}}^{*}}$$

Let $\mathrm{x} = \frac{\tilde{\sigma}_{uni}}{\tilde{\sigma}_{\mathcal{A}}^{*}}$ and $\mathrm{y} = a^{\Phi(\mathrm{m},\mathrm{v})} = \frac{\widetilde{M}_{uni}}{\widetilde{M}_{\mathcal{A}}^{*}}$; then, we have $\mathrm{x}^{\alpha} = \mathrm{y}^{2*\Phi(m,v)}$. Since α is a large prime, $gcd(\alpha, 2\cdot\Phi(m,v)) = 1$. Therefore, we obtain b_0, b_1 according to the extended Euclidean principle such that $(b_0\cdot\alpha + b_1\cdot(2\cdot\Phi(m,v))) = gcd(\alpha, 2\cdot\Phi(m,v)) = 1$. and generates the solution for the RSA instance:

$$\mathrm{y}^{\frac{1}{\alpha}} = \mathrm{y}^{b_0} x^{b_1}$$

Second, we assume that adversary $\mathcal{A}$ has ϵ-admissible, i.e., adversary $\mathcal{A}$ can pass the ϵ part of the challenge. We set the ratio of recovered files to ρ and compute $\omega = 1/2^{\lambda} + (\rho n)^{|C_1|}/(n-|C_1|+1)^{|C_1|}$. We have that $\epsilon - \omega$ is non-negligible, so $\mathcal{A}$ is able to perform a number of interactions before recovering the encoded file of the ρ ratio.

Eventually, it is possible to complete the recovery of the entire file through a number of coded files recovered according to the ρ ratio. □

Theorem 3. *If the replica generation algorithm based on time-locked puzzles is time-consuming for the prover, the proposed protocol guarantees storage allocation, which is an authentic-replica feature.*

Proof. Through the *Replicate* algorithm, we are able to find that for each replica, the time of generating the replica depends on the time when the generation of $\mu_{S_l,i,j}$ is finished, and its generation time can be determined by ϑ_i, which is simplified according to (9). $\mu_{S_l,i,j}$ is based on the variable δ. Therefore, we define the time of replica generation as $T_{cp}(\delta)$. At the same time, we assume that each adversary $\mathcal{A}_i$ did not keep all the replicas, but the only ratio of the replicas kept is ϱ. Therefore, for the adversary $\mathcal{A}_i$, the only data actually stored by the adversary $\mathcal{A}_i$ is the original encoded data and the ϱ radio of the replicas that should have been stored for each copy, that is, $|\mathbb{F}|(1+\rho\cdot|S_{\mathcal{A}}|)$, and upon receiving a challenge to a block of data from $|C_1|$, adversary $\mathcal{A}_i$ usually needs to compute the missing portion, i.e., the $(1-\varrho)$ radio of each replica. We therefore assume that the time used by adversary $\mathcal{A}_i$ to compute the missing copies $(1-\varrho)|C_1|$ in its storage is $T_{\mathcal{A}_{min}}$. To achieve this, we only need to set the forced time for copy generation to exceed the time at which the adversary is able to compute the missing copy, i.e., $T_{cp}(\delta) > T_{\mathcal{A}_{min}}$. This is because when the adversary $\mathcal{A}_i$ tries to recover the missing block, the time for the adversary $\mathcal{A}_i$ to generate the copy block is greater than $T_{\mathcal{A}_{min}}$ according to the limitations of the algorithm. In such a case, it is guaranteed that the adversary $\mathcal{A}_i$ is not able to obtain a copy between the receipt of the challenge and the generation of the replica proof, and an honest verifier can clearly notice that the adversary $\mathcal{A}_i$ does not store a full storage replica. At the same

time, in distributed environments, it is common to set $T_{cp}(\delta)$ to have a significant difference with $T_{\mathcal{A}_{min}}$, so that the verifier can obviously feel the difference between the honest prover and the adversary $\mathcal{A}_i$, which is able to resist outsourcing attacks and generation attacks.

We are given the premise that the time for adversary $\mathcal{A}_i$ to recover the $(1-\varrho)$ ratio replica at the time of receiving the challenge is at least $T_{\mathcal{A}_{min}}$, which is equivalent to adversary $\mathcal{A}_i$ being able to recover the $(1-\varrho)$ ratio replica in at most $T_{\mathcal{A}_{min}}$ time. If the ratio of copies to be recovered is less than $(1-\varrho)$, then it is possible for adversary $\mathcal{A}_i$ to complete the challenge without being suspected by the verifier. To achieve this, we need to be able to generalize and obtain the probability that the adversary will defeat the scheme in the worst case.

Given all challenge data blocks $|C_1|$, the adversary $\mathcal{A}_i$ still keeps only replicas of the ϱ ratio, and the probability that one of the challenge blocks received by the adversary $\mathcal{A}_i$ is in the replica retained by itself is Pr. Pr is the probability that out of all the challenge-selected $|C_1|$ blocks, at least one of the selected challenge blocks is in a copy block that is actually kept by the adversary $\mathcal{A}_i$:

$$Pr = \frac{C_{\varrho n}^{|C_1|} + C_{\varrho n}^{|C_1|-1} + \cdots + C_{\varrho n}^{Q|C_1|+1}}{C_n^{|C_1|}}$$

We have the number of challenged blocks $|C_1| << n$. For ease of computation, we give the reasonable assumption that $|C_1| < \varrho n/2$. Thus, for $a > b$, we have $C_{\varrho n}^{a} > C_{\varrho n}^{b}$.

$$\begin{aligned}
Pr &= \frac{C_{\varrho n}^{|C_1|} + C_{\varrho n}^{|C_1|-1} + \cdots + C_{\varrho n}^{Q|C_1|+1}}{C_n^{|C_1|}} \\
&\leq (1-\varrho|C_1|)\frac{C_{\varrho n}^{|C_1|}}{C_n^{|C_1|}} \\
&= (1-\varrho|C_1|)\frac{\varrho n(\varrho n-1)\cdots(\varrho n-|C_1|+1)}{n(n-1)\cdots(n-|C_1|+1)} \\
&= \varrho\frac{n}{n}\varrho\frac{n-\frac{1}{\varrho}}{n-1}\cdots\varrho\frac{n-\frac{|C_1|-1}{\varrho}}{n-|C_1|+1}(1-\varrho)|C_1| \\
&\leq \varrho^{|C_1|}|C_1|(1-\varrho)
\end{aligned}$$

Each S_l is storing all the source data blocks, so the probability that any one S_l alone can pass the tapping scheme is at most $\varrho^{|C_1|}|C_1|(1-\varrho)$. Note that each S_l is independent of each other, so we are able to conclude that the probability that all S_l are breached is at most $\left(\varrho^{|C_1|}|C_1|(1-\varrho)\right)^k$. □

Theorem 4. *The proposed protocol is* $(\zeta_0, \zeta_1, \cdots, \zeta_{\mathcal{N}}, \psi)$ *detectable. Specifically, each* S_l *has* $\left(\zeta_0, \{\zeta_{j_\varepsilon}\}, \psi_{S_l}\right)$ *detectable, where* $\psi_{S_l} = 1 - \prod_{j_\varepsilon \in \left([1,cp]\cap S_{l_{Rep}}\right)}(1-\zeta_{j_\varepsilon})^{|C_1|}$, *and* $(\zeta_0, \zeta_1, \cdots, \zeta_{\mathcal{N}})$ *is the corruption rate of the copy.*

Proof. Consider a data file with a corruption rate of ζ_0 and $\mathcal{N}$ replicas, each with a corruption rate of $\zeta_i, i \in [1, \mathcal{N}]$, with a corruption detectability rate of not less than $1 - \prod_{j_\varepsilon \in [0,cp]}\left(1-\zeta_{j_\varepsilon}\right)^{|C_1|}$.

A questioned data block or replica block is considered complete only if it is selected with probability $1-\zeta_{j_\varepsilon}$ from the complete portion, where $j_\varepsilon \in \left([1,cp]\cap S_{l_{Rep}}\right)$, and then, $(1-\zeta_{j_\varepsilon})^{|C_1|}$ represents the probability that all challenged blocks are deemed well preserved, applying to both the data file and its replicas. In this scenario, corruption of both the data and the replicas remains undetectable. In addition, a challenge replica is specified in each S_l such that the probability of non-detectability is $\prod_{j_\varepsilon \in ([1,cp]\cap S_{l_{Rep}}}(1-\zeta_{j_\varepsilon})^{|C_1|}$. Therefore, the probability of being undetectable is $\prod_{j_\varepsilon \in [0,cp]}\left(1-\zeta_{j_\varepsilon}\right)^{|C_1|}$ in all the challenge copies

throughout the protocol and for this reason, our protocol is detectable with a probability at least $1 - \prod_{j_\varepsilon \in [0,cp]} (1 - \zeta_{j_\varepsilon})^{|C_1|}$. □

Theorem 5. *If the $q - DABDHE$ problem is difficult, then Replicate$(\mathbb{F}, \mathcal{N}, \xi, sk_s, pp, \mathcal{T}, S_l)$ in which S_l is given to C is then safe.*

Proof. Let us assume that there is an adversary $\mathcal{A}$ capable of attacking the program. Given a problem instance $g_0, g_0^{aq+2}, g, g^a, g^{(a^2)}, \ldots, g^{(a^q)}, \mathbb{Z}$ on a bilinear pairing group $\mathbb{PG}$ and an honest entity B:

- Setup: For B, choose three polynomials of order q, $F_1(x)$, $F_2(x)$, $F_3(x)$ at random from $\mathbb{Z}_p[x]$. Then, compute

$$g_1 = g^a, h_1 = g^{F_1(a)}, h_2 = g^{F_2(a)}, h_3 = g^{F_3(a)}$$

  Then, there is a private–public key pair $(\alpha = a, g_1)$, $(\dot{\beta}_1 = F_1(a), h_1)$, $(\dot{\beta}_2 = F_2(a), h_2)$, $(\dot{\beta}_3 = F_3(a), h_3)$
- Query 1. Adversary $\mathcal{A}$ asks for ID's private key, B computes d_{ID} and gives d_{ID} to adversary $\mathcal{A}$. Adversary $\mathcal{A}$ asks for the decryption result of CT, and B computes the decryption result and gives it to adversary $\mathcal{A}$. d_{ID} = $(F_1(ID), g^{\frac{F_1(a)-F_1(ID)}{x-ID}}, F_2(ID), g^{\frac{F_2(a)-F_2(ID)}{x-ID}}, F_3(ID), g^{\frac{F_3(a)-F_3(ID)}{x-ID}})$.
- Challenge: The adversary $\mathcal{A}$ outputs two equal-length messages $m_0, m_1 \in G_T$, and the challenge identity $ID^* \in \{0,1\}^n$. Let $d_{ID} = (d_1^*, d_2^*, d_3^*, d_4^*, d_5^*, d_6^*)$ be a private key corresponding to ID^*. B randomly selects $c \in \{0,1\}$ and computes the ciphertext $CT^* = (\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*, \mathbb{C}_4^*)$, where $\mathbb{C}_1^* = g_0^{aq+2} - (ID^*)q + 2, \mathbb{C}_2^* = Z \cdot e\left(g_0, \prod_{i=0}^{q} g^{f_i a_i}\right)$, $\mathbb{C}_3^* = e(\mathbb{C}_1^*, d_6^*) \cdot (\mathbb{C}_2^*)^{d_5^*} \cdot m_c, \mathbb{C}_4^* = e(\mathbb{C}_1^*, d_2^*(d_4^*)^{w^*}) \cdot (\mathbb{C}_2^*)^{d_1^* + d_3^* w^*}$, where $w^* = H(\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*)$, and f_i represents the coefficients of x^i in the polynomial $\frac{x^{q+2} - (ID^*)^{q+2}}{x - ID^*}$.
- Query 2. Adversary $\mathcal{A}$ asks $ID \neq ID^*$ for the private key with respect to the decryption result of the ciphertext $CT \neq CT^*$, and B computes $d_{ID} = (F_1(ID), g^{\frac{F_1(a)-F_1(ID)}{x-ID}}, F_2(ID), g^{\frac{F_2(a)-F_2(ID)}{x-ID}}, F_3(ID), g^{\frac{F_3(a)-F_3(ID)}{x-ID}})$ and the corresponding decryption result.
  We let z be a random and non-zero integer, $Z = e(g_0, g)^{a^{q+1}} \cdot e(g, g)^z$. Challenge ciphertexts $\mathbb{C}_1^* = g^{s(\alpha - ID^*)}$, $\mathbb{C}_2^* = e(g, g)^{s+z}$, $\mathbb{C}_3^* = e(g, g)^{z d_3^*} \cdot e(h_3, g)^s \cdot m_c$.

By randomness, adversary $\mathcal{A}$ can only obtain d_5^* from a decryption query to (ID^*, CT). B can make $CT = (\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3, \mathbb{C}_4)$, $\mathbb{C}_1 = (g_1 g^{ID^*})^{s'}$, $\mathbb{C}_2 = e(g, g)^{s''}, w = H(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3)$. If $s\prime = s\prime\prime$, then B can output $\frac{\mathbb{C}_3}{e(\mathbb{C}_1, d_6^*) \cdot \mathbb{C}_2^{d_5^*}} = \frac{\mathbb{C}_3}{e(h_3, g)^{s'}}$, and the adversary cannot obtain d_5^* from it. If $s' \neq s''$, then the ciphertext is wrong and B can reject it outright.

Since $a, F_1(x), F_2(x), F_3(x), log_g(g_0)$ are all random, the encryption scheme is randomized. Then,

- If $(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3) = (\mathbb{C}_1^*, \mathbb{C}_2^*, \mathbb{C}_3^*)$ and $H(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3) = w = w^*$, in order for the incorrect ciphertext to be verified, it would need to be $\mathbb{C}_4 = \mathbb{C}_4^*$. However, the ciphertext is a challenge ciphertext and is not interrogated.
- Else, $H(\mathbb{C}_1, \mathbb{C}_2, \mathbb{C}_3) = w \neq w^*$. In order to make the ciphertext go through, the adversary needs to compute to obtain $\mathbb{C}_4$. But in $\mathbb{C}_4 = e(\mathbb{C}_1, d_2^*(d_4^*)^w) \cdot \mathbb{C}_2^{d_1^* + d_3^* w}$, since $d_1^* + d_3^* w$ and $d_1^* + d_3^* w^*$ are random and independent, the adversary has no advantage to generate a verifiable $\mathbb{C}_4$. The probability of the first $\mathbb{C}_4$ adaptive selection is $\frac{1}{p}$ and the second is $\frac{1}{p-1}$. Thus, q_d decrypted queries with a probability of at most $\frac{q_d}{p - q_d}$. Plus, the probability of guessing at least c correctly is $1/2$. Thus, the probability that the adversary succeeds is at most $1/2 + \frac{q_d}{p - q_d}$.

□

5. Performance of Our Scheme

To evaluate performance, we compare the property analysis and experimental results of the proposed protocol with those of the schemes by Armknecht et al. [22], Guo et al. [39], and Shen et al. [23]. All three protocols support proof of data replication.

5.1. Property Analysis

The performance analysis is shown in Table 2. Armknecht et al. [22] and Guo et al. [39] require a large amount of precomputation of private copy parameters to generate replicas on the client side due to the use of secret and public LFSRs, which leads to the fact that these two protocols only support private authentication. Secondly, the user-friendly solution [23] to the above two problems proposes a new replica structure that is publicly verifiable. However, the new replica structure in Shen et al. [23] does not satisfy the requirements of a distributed environment, cannot satisfy public verifiability in distributed environment, does not have authentic-replica characteristics, and does not have the problem of resisting disaster environments. Therefore, we propose a new public authentic-replica sampling mechanism in distributed storage environments, which ensures a low amount of preparation work for replica generation. It also ensures that under the same number of replicas, each server can generate replicas faster, and the authentication expenditure is independent of the number of replicas. At the same time, it can resist outsourcing attacks and generation attacks, and it can achieve public verifiability in distributed storage environments.

Table 2. Comparison of properties.

Protocols	Armknecht et al. [22]	Guo et al. [39]	Shen et al. [39]	Our
Storage allocation	×	×	✓	✓
Efficient preparation for replication	×	×	✓	✓
Public verifiability	×	×	✓	✓
Verification time	high	high	low	low
Disaster recovery capability	low	low	low	high
Resist generation attacks, outsourcing attacks	×	×	×	✓
Support multi-cloud	×	×	×	✓

5.2. Communication Overhead

The communication overhead for different schemes is shown in Table 3, where λ^* denotes the length of the public LSFR. In this comparison, the other schemes are subjected to multiple experiments according to the k-cloud environment, and the results show that we achieve the transmission of Ref-Tables and random numbers through a small amount of additional overhead, which must be borne to ensure the security of the scheme.

Table 3. Comparison of communication.

Scheme	Armknecht et al. [22]	Guo et al. [39]	Shen et al. [23]	Ours
Copy parameters	$\mathcal{N}\lambda^*(\lvert\mathbb{Z}_{q_0'}\rvert + \lvert\mathbb{Z}_{q_1'}\rvert + 2\lambda^*\lvert\mathbb{Z}\rvert)$	$\mathcal{N}\lambda^*(\lvert\mathbb{Z}_{q_0'}\rvert + \lvert\mathbb{Z}_{q_1'}\rvert + 2\lambda^*\lvert\mathbb{Z}\rvert)$	$\lvert\mathbb{Z}_p\rvert + \lvert\mathbb{Z}\rvert$	$\lvert\mathbb{Z}_p\rvert + \lvert\mathbb{Z}\rvert$
Data uploading	$2n[k]\lvert\mathbb{Z}_N^*\rvert$	$2n[k]\lvert\mathbb{Z}_N^*\rvert$	$2[k](n\lvert\mathbb{Z}_N^*\rvert + \lvert\mathbb{Z}_{q_0'q_1'}\rvert)$	$((2n+1)k\lvert\mathbb{Z}_N^*\rvert + 2k\lvert\mathbb{Z}_{q_0'q_1'}\rvert$
Proof responding	$2\lvert\mathbb{Z}_N^*\rvert$	$2\lvert\mathbb{Z}_N^*\rvert$	$5\lvert\mathbb{Z}_N^*\rvert$	$6\lvert\mathbb{Z}_N^*\rvert$

5.3. Experimental Performance

In the experiments, the test file size was 100 MB, the RSA mode size was set to 3072 bits, the experiments were performed on Ubuntu 20.04.4 LTS, the computer was configured with an Intel i5 CPU running VMware Fusion 13.0.0, and the programs were executed with Python 3.8.

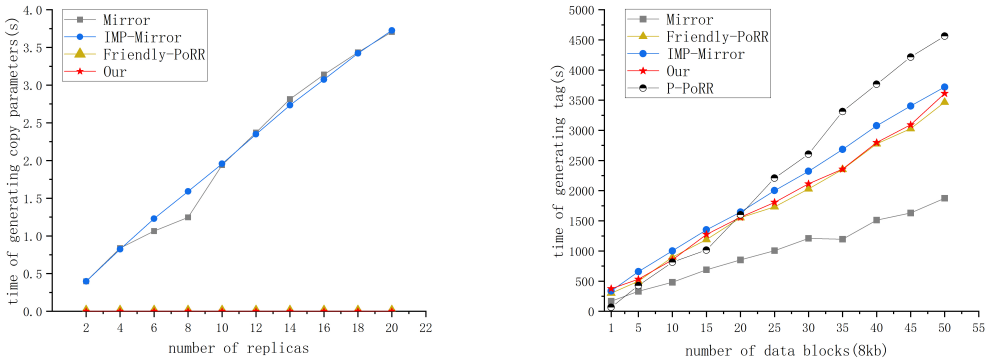
A numerical comparison of the computational costs is shown in Table 4, where P denotes power taking on $\mathbb{Z}$, M is multiplication on groups, R denotes PRF, V is the inverse operation, λ is the length of the secret $LFSR$, δ is the tunable parameter when generating the replicas, and $|C_1|$ and $|C_2|$ are the number of all challenge blocks and replicas, respectively. Several operations with negligible computational overhead, such as hashing and modulo addition, are ignored in the comparison. To ensure that the single replica generation time is consistent in the experiments, we set the adjustable parameter to 8 in both the Shen et al. [23] scheme and our scheme.

Table 4. Comparsion of computation.

Scheme	Armknecht et al. [22]	Guo et al. [39]	Our
Copy parameter generation	$2\mathcal{N}(\lambda(\lambda^* - \lambda)\mathrm{M} + \lambda^* P)$	$2\mathcal{N}(\lambda(\lambda^* - \lambda)\mathrm{M} + \lambda^* P)$	0
Tag generation	nP	$n(2P + M + R)$	$2nP + nM + V$
Replica generation	$2\mathcal{N}\lambda^*((n - \lambda^*)(P + M) + M)$	$2t\lambda^*((n - \lambda^*)(P + M) + M)$	$\mathcal{N}((n+1)P$ $+n(1+\delta)M + \lvert C_1\rvert)$
Proof generation	$2\lvert C_1\rvert\lvert P + (2 + \lvert C_2\rvert)\lvert C_1\rvert\mathrm{M}$	$2\lvert C_1\rvert\lvert P + (2 + \lvert C_2\rvert)\lvert C_1\rvert M$	$3\lvert C_1\rvert P + (3 + \lvert C_2\rvert)\lvert C_1\rvert M$
Proof Verification	$\lvert C_1\rvert(\lvert C_2\rvert + 1)P +$ $(\lvert C_1\rvert + \lvert C_2\rvert + \lvert C_1\rvert\lvert C_2\rvert + \lambda n)M$	$\lvert C_1\rvert(\lvert C_2\rvert + 1)P$ $+(\lvert C_1\rvert + \lvert C_2\rvert + \lvert C_1\rvert\lvert C_2\rvert + \lambda n)M$ $+\lvert C_1\rvert(M + R)$	$(\lvert C_1\rvert + 3\lvert C_2\rvert)P$ $+(\lvert C_2\rvert + (\delta + 1)\lvert C_1\rvert)M$

We demonstrate that the authentic-replica sampling scheme is effective in five ways: the overhead time of copy parameters, the time of generating data tags, the time of generating the same number of replicas, the time of generating proofs and verifying the proofs of the challenge blocks, and the total time of the different replicas and challenged blocks. We compare this paper scheme with those of Guo et al. [39], Armknecht et al. [22], Gritti et al. [45] and Shen et al. [23].

We demonstrate the time required to generate copy parameters for different numbers of replicas. To maintain consistency with the subsequent experiments, we limit the number of replicas to 20 or fewer. As shown in Figure 2a, the copy parameters in our scheme are simply the random seeds sent by the client to each cloud storage node and the Ref-Table. Thus, the transmission time is extremely short and can be almost neglected. This indicates that the overhead of generating copy parameters in our scheme is independent of the number of replicas, presenting a very low constant overhead.

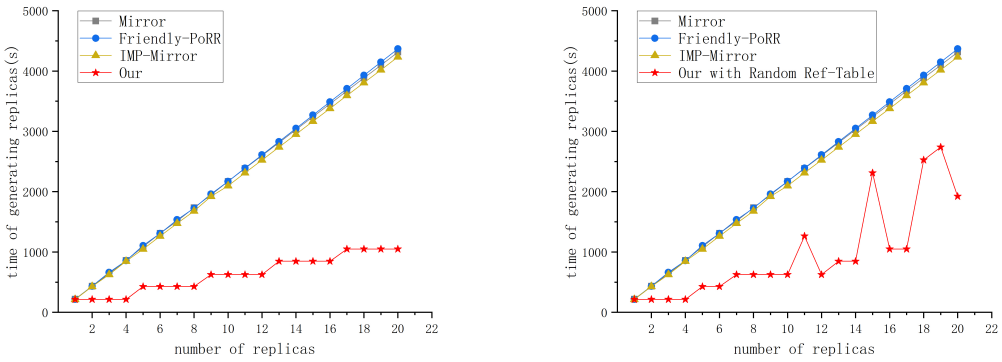


(**a**) (**b**)

Figure 2. The time cost of copy parameters generation (**a**) between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23] and time cost of tags generation (**b**) between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23], P-PoRR [45].

We demonstrate the time consumed to generate the corresponding tags for different data blocks. As shown in Figure 2b, the tag generation time in our scheme is not the fastest. This is because the tags we set need to ensure the final tags can be aggregated. Additionally, to ensure higher security, especially to resist outsourcing attacks and generation attacks, we made a compromise in the tag generation time. Constructing more secure tags requires additional computation time, which leads to an increase in time overhead. However, this overhead is justified, as the extra time is invested to enhance the security and robustness of the overall scheme, better protecting data integrity.

We demonstrate the time overhead required for generating replicas for different sizes of replica data when the data block size is fixed at 8 KB Figure 3a. We know that all the necessary parameters for generating replicas, such as random numbers and the Ref-Table, have already been obtained before replica generation. Therefore, in our scheme, after distributing the delegated replica data across different clouds, the replica generation time is significantly accelerated. In contrast, in the schemes proposed by Armknecht et al. [22], Guo et al. [39], and Shen et al. [23] for single-cloud storage, the replica generation time increases linearly with the number of replicas. Additionally, we randomly constructed the Ref-Table to calculate the time for replica generation in a multi-cloud environment; see Figure 3b. Although the time overhead is not ideal, it demonstrates that our scheme can handle different delegation numbers for different clouds based on the Ref-Table.

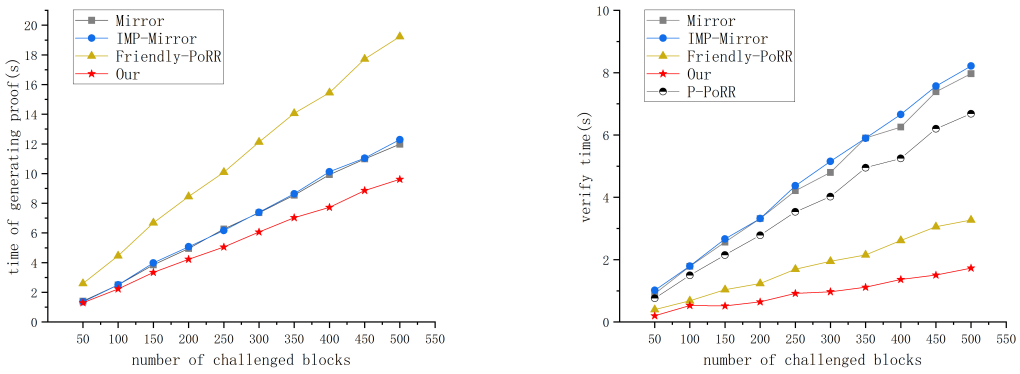


(**a**) (**b**)

Figure 3. The time cost of copy parameters generation with uniformly distributed Ref-Table (**a**) and random Ref-Table (**b**) between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23].

We demonstrate the time overhead required for generating proofs for different numbers of challenged blocks. In comparison with the schemes of Armknecht et al. [22], Guo et al. [39], and Shen et al. [23] in a single-cloud environment, as shown in Figure 4a, by eliminating communication delays and using the same delegated replicas, our approach generates proofs more efficiently. In contrast to a single-cloud environment, when each cloud is challenged, it only needs to generate proofs for the challenged blocks corresponding to the replicas it stores. Since the number of replicas is distributed across different clouds, the challenges are also spread out, accelerating the proof generation process.

In Figure 4b, compared to the schemes of Armknecht et al. [22], Guo et al. [39], Gritti et al. [45] and Shen et al. [23], our solution aggregates all servers belonging to the same cloud in the authentication phase with a unique identifier for that cloud. At the same time, the extension to all clouds will enable batch authentication in multi-cloud environments, which will lead to a reduction in authentication time.



(**a**) (**b**)

Figure 4. Time cost of proofs generation (**a**) between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23] and time cost of verify time (**b**) with different numbers of challenged blocks between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23], P-PoRR [45].

We present the total time overhead required for different numbers of replicas and different challenged blocks under the condition of maintaining 20 data blocks, each sized 8 KB, across various schemes. As shown in Figure 5, the efficiency advantage of our scheme is clearly demonstrated. Regardless of the scenario, our scheme consistently achieves the optimal overall time efficiency.

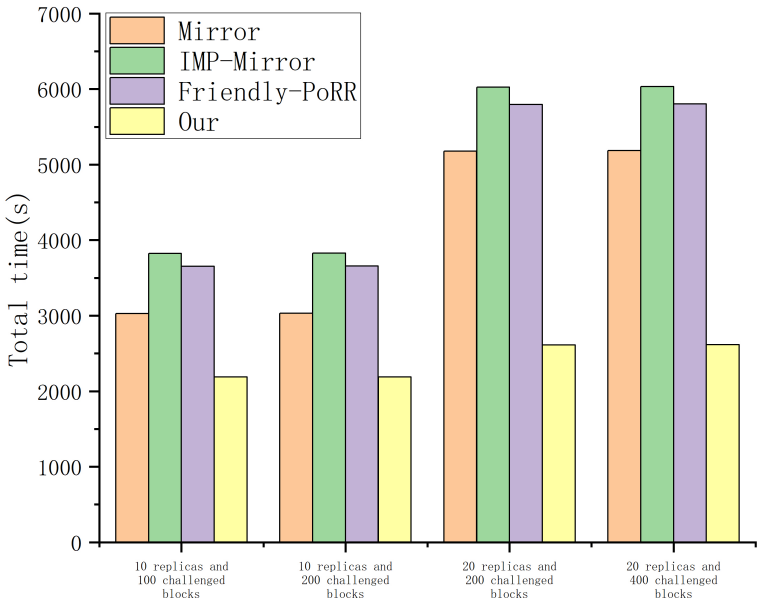


Figure 5. The total time with between our scheme and Mirror [22], IMP-Mirror [39], Friendly-PoRR [23].

The proposed scheme demonstrates excellent performance under various parameter settings, making it adaptable to environments with large files, multiple replicas, and multi-cloud storage. Furthermore, its resistance to outsourcing attacks and generation attacks, combined with a constant-level user computation overhead, makes it suitable for most

multi-cloud environments. Examples include its application in cloud storage systems for electronic medical records, electronic evidence, and e-commerce systems.

6. Conclusions

In this paper, we propose a public authentic-replica sampling mechanism in a distributed storage environment, give a formal definition of authentic-replica, and at the same time give a security model of the authentic-replica sampling mechanism. Based on the time-locking puzzle, we design an authentic-replica auditing mechanism using an identity encryption mechanism and succinct proof technique, which enables the scheme to satisfy the authentic-replica characteristics and resist outsourcing attacks and generation attacks. In addition, the scheme achieves the cost of generating and uploading copy parameters by the client to be at a constant level even if the number of stored replicas grows through the combination of random numbers and Ref-Table, which ensures the server will store data with a smaller bandwidth, and at the same time, guarantees the publicly verifiable characteristic through the publicly recoverable copy parameters. The experimental results show that the scheme is secure, acceptably efficient, and reasonable. The experiments show that our scheme makes the time required to generate copy parameters negligible while significantly optimizing the time required for replica generation. As the amount of replica data increases, the time spent does not grow linearly. Additionally, due to the multi-party aggregation design, the verification time is also optimal. Compared to the latest schemes, the verification time is reduced by approximately 30%. We will expand the functionality of data migration as the future direction of this paper, aiming to dynamically migrate user data in distributed storage while maintaining low costs. Ensuring that clients are convinced the new physical space can meet their needs will be a key issue we need to focus on and resolve moving forward.

Author Contributions: Conceptualization, J.Y.; methodology, J.Y.; validation, J.Y. and Y.B.; resources, J.X.; writing—original draft preparation, J.Y.; writing—review and editing, J.Y. and S.H.; visualization, J.Y.; supervision, Z.H.; funding acquisition, W.W. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Natural Science Foundation of China grant number 62072249. This work was also supported by the National Natural Science Foundation of China (62032025, 62172258), and the Shenzhen Science and Technology Program (JCYJ20210324134810028).

Data Availability Statement: Data are contained within the article.

Conflicts of Interest: Author Wei Wan was employed by the company State Grid ZaoZhuang Power Supply Company. The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

References

1. Xue, K.; Li, S.; Hong, J.; Xue, Y.; Yu, N.; Hong, P. Two-cloud secure database for numeric-related SQL range queries with privacy preserving. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 1596–1608. [CrossRef]
2. Majumdar, A.; Biswas, A.; Majumder, A.; Sood, S.K.; Baishnab, K.L. A novel DNA-inspired encryption strategy for concealing cloud storage. *Front. Comput. Sci.* **2021**, *15*, 1–18. [CrossRef]
3. Noor, T.H.; Sheng, Q.Z.; Zeadally, S.; Yu, J. Trust management of services in cloud environments: Obstacles and solutions. *ACM Comput. Surv. (CSUR)* **2013**, *46*, 1–30. [CrossRef]
4. Mansouri, Y.; Toosi, A.N.; Buyya, R. Data storage management in cloud environments: Taxonomy, survey, and future directions. *ACM Comput. Surv. (CSUR)* **2017**, *50*, 1–51. [CrossRef]
5. Khan, A.A.; Zakarya, M. Energy, performance and cost efficient cloud datacentres: A survey. *Comput. Sci. Rev.* **2021**, *40*, 100390. [CrossRef]
6. Li, Y.; Yu, Y.; Min, G.; Susilo, W.; Ni, J.; Choo, K.K.R. Fuzzy identity-based data integrity auditing for reliable cloud storage systems. *IEEE Trans. Dependable Secur. Comput.* **2017**, *16*, 72–83. [CrossRef]
7. Wei, J.; Chen, X.; Wang, J.; Huang, X.; Susilo, W. Securing fine-grained data sharing and erasure in outsourced storage systems. *IEEE Trans. Parallel Distrib. Syst.* **2022**, *34*, 552–566. [CrossRef]
8. Zhang, Y.; Yu, J.; Hao, R.; Wang, C.; Ren, K. Enabling efficient user revocation in identity-based cloud storage auditing for shared big data. *IEEE Trans. Dependable Secur. Comput.* **2018**, *17*, 608–619. [CrossRef]

9. Xu, J.; Wang, C.; Jia, X. A survey of blockchain consensus protocols. *ACM Comput. Surv.* **2023**, *55*, 1–35. [CrossRef]
10. Xiao, Y.; Zhang, N.; Lou, W.; Hou, Y.T. A survey of distributed consensus protocols for blockchain networks. *IEEE Commun. Surv. Tutor.* **2020**, *22*, 1432–1465. [CrossRef]
11. Majumdar, S.; Chawla, G.S.; Alimohammadifar, A.; Madi, T.; Jarraya, Y.; Pourzandi, M.; Wang, L.; Debbabi, M. ProSAS: Proactive security auditing system for clouds. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 2517–2534. [CrossRef]
12. He, D.; Kumar, N.; Zeadally, S.; Wang, H. Certificateless provable data possession scheme for cloud-based smart grid data management systems. *IEEE Trans. Ind. Inform.* **2017**, *14*, 1232–1241. [CrossRef]
13. Miao, Y.; Huang, Q.; Xiao, M.; Susilo, W. Blockchain assisted multi-copy provable data possession with faults localization in multi-cloud storage. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 3663–3676. [CrossRef]
14. Gudeme, J.R.; Pasupuleti, S.K.; Kandukuri, R. Certificateless multi-replica public integrity auditing scheme for dynamic shared data in cloud storage. *Comput. Secur.* **2021**, *103*, 102176. [CrossRef]
15. Zhao, Y.; Qu, Y.; Xiang, Y.; Uddin, M.P.; Peng, D.; Gao, L. A comprehensive survey on edge data integrity verification: Fundamentals and future trends. *ACM Comput. Surv.* **2024**, *57*, 1–34. [CrossRef]
16. Yu, H.; Yang, Z.; Waqas, M.; Tu, S.; Han, Z.; Halim, Z.; Sinnott, R.O.; Parampalli, U. Efficient dynamic multi-replica auditing for the cloud with geographic location. *Future Gener. Comput. Syst.* **2021**, *125*, 285–298. [CrossRef]
17. Garg, N.; Bawa, S.; Kumar, N. An efficient data integrity auditing protocol for cloud computing. *Future Gener. Comput. Syst.* **2020**, *109*, 306–316. [CrossRef]
18. Zhou, L.; Fu, A.; Mu, Y.; Wang, H.; Yu, S.; Sun, Y. Multicopy provable data possession scheme supporting data dynamics for cloud-based electronic medical record system. *Inf. Sci.* **2021**, *545*, 254–276. [CrossRef]
19. Benisi, N.Z.; Aminian, M.; Javadi, B. Blockchain-based decentralized storage networks: A survey. *J. Netw. Comput. Appl.* **2020**, *162*, 102656. [CrossRef]
20. Susilo, W.; Li, Y.; Guo, F.; Lai, J.; Wu, G. Public cloud data auditing revisited: Removing the tradeoff between proof size and storage cost. In Proceedings of the European Symposium on Research in Computer Security, Copenhagen, Denmark, 26–30 September 2022; pp. 65–85.
21. Sellami, Y.; Imine, Y.; Gallais, A. A verifiable data integrity scheme for distributed data sharing in fog computing architecture. *Future Gener. Comput. Syst.* **2024**, *150*, 64–77. [CrossRef]
22. Armknecht, F.; Barman, L.; Bohli, J.M.; Karame, G.O. Mirror: Enabling proofs of data replication and retrievability in the cloud. In Proceedings of the 25th USENIX Security Symposium (USENIX Security 16), Austin, TX, USA, 10–12 August 2016; pp. 1051–1068.
23. Shen, J.; Chen, X.; Huang, X.; Xiang, Y. Public Proofs of Data Replication and Retrievability with User-friendly Replication. *IEEE Trans. Dependable Secur. Comput.* **2023**, *31*, 2057–2067. [CrossRef]
24. Ren, Y.; Leng, Y.; Cheng, Y.; Wang, J. Secure data storage based on blockchain and coding in edge computing. *Math. Biosci. Eng* **2019**, *16*, 1874–1892. [CrossRef] [PubMed]
25. Sookhak, M.; Gani, A.; Talebian, H.; Akhunzada, A.; Khan, S.U.; Buyya, R.; Zomaya, A.Y. Remote data auditing in cloud computing environments: A survey, taxonomy, and open issues. *ACM Comput. Surv. (CSUR)* **2015**, *47*, 1–34. [CrossRef]
26. Daniel, E.; Tschorsch, F. IPFS and friends: A qualitative comparison of next generation peer-to-peer data networks. *IEEE Commun. Surv. Tutor.* **2022**, *24*, 31–52. [CrossRef]
27. Yu, H.; Chen, Y.; Yang, Z.; Chen, Y.; Yu, S. EDCOMA: Enabling Efficient Double Compressed Auditing for Blockchain-Based Decentralized Storage. *IEEE Trans. Serv. Comput.* **2024**, *17*, 2273–2286. [CrossRef]
28. Zhou, M.; Yang, Z.; Yu, H.; Yu, S. VDFChain: Secure and verifiable decentralized federated learning via committee-based blockchain. *J. Netw. Comput. Appl.* **2024**, *223*, 103814. [CrossRef]
29. Wang, X.; Yu, H.; Chen, Y.; Sinnott, R.O.; Yang, Z. PrVFL: Pruning-Aware Verifiable Federated Learning for Heterogeneous Edge Computing. *IEEE Trans. Mob. Comput.* **2024**, 1–18.. [CrossRef]
30. Ren, Y.; Lv, Z.; Xiong, N.N.; Wang, J. HCNCT: A cross-chain interaction scheme for the blockchain-based metaverse. *ACM Trans. Multimed. Comput. Commun. Appl.* **2024**, *20*, 1–23. [CrossRef]
31. Du, Y.; Duan, H.; Zhou, A.; Wang, C.; Au, M.H.; Wang, Q. Enabling secure and efficient decentralized storage auditing with blockchain. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3038–3054. [CrossRef]
32. Li, Y.; Yu, Y.; Chen, R.; Du, X.; Guizani, M. IntegrityChain: Provable data possession for decentralized storage. *IEEE J. Sel. Areas Commun.* **2020**, *38*, 1205–1217. [CrossRef]
33. Yang, Y.; Chen, Y.; Chen, F.; Chen, J. An efficient identity-based provable data possession protocol with compressed cloud storage. *IEEE Trans. Inf. Forensics Secur.* **2022**, *17*, 1359–1371. [CrossRef]
34. Tian, G.; Hu, Y.; Wei, J.; Liu, Z.; Huang, X.; Chen, X.; Susilo, W. Blockchain-based secure deduplication and shared auditing in decentralized storage. *IEEE Trans. Dependable Secur. Comput.* **2021**, *19*, 3941–3954. [CrossRef]
35. Ren, Y.; Leng, Y.; Qi, J.; Sharma, P.K.; Wang, J.; Almakhadmeh, Z.; Tolba, A. Multiple cloud storage mechanism based on blockchain in smart homes. *Future Gener. Comput. Syst.* **2021**, *115*, 304–313. [CrossRef]
36. Tang, J.; Cui, Y.; Li, Q.; Ren, K.; Liu, J.; Buyya, R. Ensuring security and privacy preservation for cloud data services. *ACM Comput. Surv. (CSUR)* **2016**, *49*, 1–39. [CrossRef]
37. Sun, L.; Wang, Y.; Ren, Y.; Xia, F. Path signature-based xai-enabled network time series classification. *Sci. China Inf. Sci.* **2024**, *67*, 170305. [CrossRef]

38. Barsoum, A.F.; Hasan, M.A. Provable multicopy dynamic data possession in cloud computing systems. *IEEE Trans. Inf. Forensics Secur.* **2014**, *10*, 485–497. [CrossRef]
39. Guo, W.; Qin, S.; Lu, J.; Gao, F.; Jin, Z.; Wen, Q. Improved proofs of retrievability and replication for data availability in cloud storage. *Comput. J.* **2020**, *63*, 1216–1230. [CrossRef]
40. Zhang, C.; Li, X.; Au, M.H. epost: Practical and client-friendly proof of storage-time. *IEEE Trans. Inf. Forensics Secur.* **2023**, *18*, 1052–1063. [CrossRef]
41. Boneh, D.; Bonneau, J.; Bünz, B.; Fisch, B. Verifiable delay functions. In Proceedings of the Annual International Cryptology Conference, Santa Barbara, CA, USA, 9–23 August 2018; pp. 757–788.
42. Liu, Y.; Wang, Q.; Yiu, S.M. Towards practical homomorphic time-lock puzzles: Applicability and verifiability. In Proceedings of the European Symposium on Research in Computer Security, Copenhagen, Denmark, 26–30 September 2022; pp. 424–443.
43. Katz, J.; Loss, J.; Xu, J. On the security of time-lock puzzles and timed commitments. In Proceedings of the Theory of Cryptography: 18th International Conference, TCC 2020, Durham, NC, USA, 16–19 November 2020; Proceedings, Part III 18; Springer: Berlin/Heidelberg, Germany, 2020; pp. 390–413.
44. Boneh, D.; Bünz, B.; Fisch, B. Batching techniques for accumulators with applications to IOPs and stateless blockchains. In Proceedings of the Advances in Cryptology–CRYPTO 2019: 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, 18–22 August 2019; Proceedings, Part I 39; Springer Berlin/Heidelberg, Germany, 2019; pp. 561–586.
45. Gritti, C. Publicly verifiable proofs of data replication and retrievability for cloud storage. In Proceedings of the 2020 International Computer Symposium (ICS), Tainan, Taiwan, 17–19 December 2020; pp. 431–436.

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.