Multi-replicas integrity checking scheme with supporting probability audit for cloud-based IoT

Yilin Yuan[1], Fan Yang[1], Xiao Wang[2,3,4], Yimin Tian[1,5] and Zichen Li[1]

[1] College of Information Engineering, Beijing Institute of Graphic Communication, Beijing, China
[2] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Guangzhou, Guangdong, China
[3] Department of Information Science and Technology, Tianjin University of Finance and Economics, Tianjin, China
[4] College of Intelligence and Computing, Tianjin University, Tianjin, China
[5] Science Education Department, Beijing Institute of Graphic Communication, Beijing, China

Submitted 14 July 2023
Accepted 11 December 2023
Published 16 January 2024

Corresponding authors
Yilin Yuan, yuanyilin@bigc.edu.cn
Xiao Wang, wangxiao8343@163.com

Academic editor
Shi Dong

Additional Information and Declarations can be found on page 21

DOI **10.7717/peerj-cs.1790**

ABSTRACT

Nowadays, more people are choosing to use cloud storage services to save space and reduce costs. To enhance the durability and persistence, users opt to store important data in the form of multiple copies on cloud servers. However, outsourcing data in the cloud means that it is not directly under the control of users, raising concerns about security and integrity. Recent research has found that most existing multicopy integrity verification schemes can correctly perform integrity verification even when multiple copies are stored on the same Cloud Service Provider (CSP), which clearly deviates from the initial intention of users wanting to store files on multiple CSPs. With these considerations in mind, this paper proposes a scheme for synchronizing the integrity verification of copies, specifically focusing on strongly privacy Internet of Things (IoT) electronic health record (EHR) data. First, the paper addresses the issues present in existing multicopy integrity verification schemes. The scheme incorporates the entity Cloud Service Manager (CSM) to assist in the model construction, and each replica file is accompanied with its corresponding homomorphic verification tag. To handle scenarios where replica files stored on multiple CSPs cannot provide audit proof on time due to objective reasons, the paper introduces a novel approach called probability audit. By incorporating a probability audit, the scheme ensures that replica files are indeed stored on different CSPs and guarantees the normal execution of the public auditing phase. The scheme utilizes identity-based encryption (IBE) for the detailed design, avoiding the additional overhead caused by dealing with complex certificate issues. The proposed scheme can withstand forgery attack, replace attack, and replay attack, demonstrating strong security. The performance analysis demonstrates the feasibility and effectiveness of the scheme.

Subjects Cryptography, Distributed and Parallel Computing, Security and Privacy, Internet of Things
Keywords Multi-replicas integrity verification, Public auditing, EHR data, Identity-based encryption, Probability audit

INTRODUCTION

With the advent of the era of big data, the types and quantities of data have shown explosive growth. At the same time, the methods and devices of data storage have also received more attention. For example, from large-capacity nonportable solid-state storage devices to small-capacity portable USB flash drives, to large-capacity portable mobile hard drives, people are always willing to store data on devices with high flexibility and capacity. Fortunately, cloud storage services can better meet the needs of users. Users who choose to use cloud storage services do not need to deploy any physical devices locally, nor do they need to be involved in the daily maintenance of outsourcing data, they can simply focus on enjoying the service. Therefore, cloud storage services have been chosen by more and more users in recent years. However, users who choose cloud storage services will, by default, transfer the control of the data to the Cloud Storage Provider (CSP) after uploading the outsourced file. Despite the popularity of cloud storage services, their security and reliability remain subject to skepticism. Therefore, ensuring the security and integrity of outsourced data when using cloud storage services is a research hotspot for scholars. Currently, more valuable schemes have been proposed that can effectively verify the integrity of remote data.

To enhance the availability and durability of the outsourced data, users choose to store important data on multiple CSPs with different geographic locations or different types. Therefore, after completing the data upload, verifying the integrity of duplicate files is an issue worth considering. On the one hand, due to the increased complexity of verifying multicopy files compared to a single file, the following issues need to be considered: (1) How should the duplicate files be generated to guarantee storage security? (2) How to design the homomorphic verification tag (HVT) to realize synchronous verification of duplicate file integrity? (3) How can we improve verification efficiency? (4) How do we implement recovery for damaged replicas? These are the primary issues that need to be addressed when designing a multicopy data integrity verification scheme. On the other hand, most existing multicopy file integrity verification schemes have almost not taken into account the distribution of replica storage locations. Specifically, while most schemes claim to simultaneously check replicas stored in different geographical locations, this is not actually the case, as the duplicate files in their schemes are actually stored on the same CSP (which will be detailed in the 'Related Work' section). Clearly, if the storage server fails, all duplicate files of the user will be damaged. Even if the cloud service provider offers compensation, the user's important data has already been compromised, which is bound to erode the user's confidence in them. Therefore, the user's duplicate files should be stored on multiple CSPs located in different geographical locations to minimize the risk of data loss. Similarly, how to conduct synchronous checks on these duplicate files is also a crucial issue.

As mentioned earlier, the era of big data has arrived and both storage and privacy security should be guaranteed for outsourced cloud data. The Internet of Things (IoT), as a rapidly evolving technology in recent years, connects devices and sensors through networks, utilizing cloud computing to process and transmit data to achieve interconnectedness of all things. Recently, the proliferation of wearable devices has made the integration of IoT and big data in healthcare even more closely intertwined. For example, most hospitals currently use electronic health records (EHR), which serve as a form of healthcare big data encompassing a patient's entire life process, including identity information, health status, and medical history, among other details. The EHR data comes from multiple channels, making it comprehensive and detailed. Due to its sheer volume, storing it in the cloud is a viable solution. However, an EHR contains various sensitive information, and outsourcing it directly to the cloud would inevitably lead to privacy breaches. Furthermore, not all EHR data can be shared among hospitals, so creating copies of EHR when patients visit different hospitals would be more convenient. Therefore, as one of the data types in cloud-based IoT, it is essential to safeguard the security and integrity of EHR replica files.

Related work

To verify remote data integrity in the cloud storage environment, existing schemes can be broadly categorized into two types: data possession verification and data retrieval verification. In 2007, *Ateniese et al. (2007)* proposed the Provable Data Possession (PDP) scheme. The PDP scheme employs random sampling and is essentially a probabilistic detection model. Notably, it not only enables blockless verification but also significantly reduces the I/O overhead during the remote checking process. In the same year, *Juels & Kaliski (2007)* proposed the Proof of Retrievability (PoR) scheme. PoR scheme adds a special data block named "Sentinels" for detection and introduces erasure coding technology, so it can complete remote data integrity checking and data retrieval simultaneously. Building on the PoR scheme, *Shacham & Waters (2008)* proposed an enhanced scheme. In *Shacham & Waters (2008)*, two methods for constructing homomorphic verification tag (HVT) are presented: when constructed based on pseudo-random functions (PRF), this scheme supports private verification and is shown to be secure in the standard model; when constructed using BLS signatures, this scheme supports public verification and is proven secure in a random oracle model. Building on these foundational schemes, subsequent research has made significant contributions to the field of data integrity verification (*Ateniese et al., 2008*; *Erway et al., 2009*; *Wang et al., 2011b*; *Wang et al., 2011a*; *Tian et al., 2015*; *Li, Yan & Zhang, 2020*; *He et al., 2021*; *Shu et al., 2021*; *Wang, Wang & He, 2021*; *Zhang et al., 2020*; *Shang et al., 2021*).

In the scheme of using public key infrastructure (PKI) to distribute keys, PKI is an indispensable entity. However, the presence of certificates places a substantial burden on verification processes. For example, during data integrity check, the users must verify both the data and the certificate, while the system is tasked with tasks such as certificate generation, forwarding, storage, checking, and updates. In actual use, certificate management will be laborious and inefficient. In 1984, *Shamir (1984)* proposed an identity-based key system, in which the user's unique identity, such as e-mail, phone number, *etc.*, can serve as a public key and the corresponding private key is generated by the private key generator (PKG). This eliminates the need for PKI, greatly reducing the reliance on certificates in Identity-Based Encryption (IBE). In 2001, *Boneh & Franklin (2001)* provided the first practical IBE scheme based on the weil pairing. Following this, *Wang et al. (2014)*

proposed the first public data integrity verification scheme constructed using IBE. *Zhang & Dong (2016)* proposed a public auditing scheme that combines bilinear mapping and IBE construction, requiring only constant-level computational cost. *Tian, Gao & Chen (2019)* applied the ideal lattice based on the polynomial structure to key generation and proposed the scheme that can achieve efficient key generation and low-cost storage. *Wang, He & Tang (2016)* introduce Proxy, a trusted entity, and discuss how to conduct public auditing when users face restrictions on accessing CSPs. *Shen et al. (2018)* discussed that in the context of a big data environment, by adding a trusted entity Sanitizer, the purpose of hiding sensitive user information is achieved. *Li et al. (2017)* provide a method to convert the feature vector generated by the biometric information of the users, such as the iris, fingerprint, *etc*., into a usable key and construct a public audit scheme that supports the input of fuzzy identities. *Yu et al. (2016)* proposed a new method for key construction using RSA.

To address the challenge of public auditing of multiple-replica, *Curtmola et al. (2008)* introduced the initial multiple replica provable data possession (MR-PDP) scheme. This scheme employs RSA to construct the HVT, demonstrating that the time required to verify multiple copies of files together is significantly less than the time required to verify them individually. However, the calculation and communication cost of this scheme is relatively large. *Shamir (1984)* designed the HVT by assisting with the vector dot product and proposed a flexible multiple replica provable data possession (FMR-PDP) scheme. Although the FMR-PDP scheme has great advantages in computing and communication overhead, it only considers private verification, which limits its practicality. *Barsoum & Hasan (2014)* proposed a Provable Multicopy Dynamic Data Possession (PMDDP) scheme to realize replica dynamics by mapping version tables. The PMDDP scheme nests the number of copies into the HVT constructed by RSA. Although the modification, insertion, and deletion of the specified data block in the copy file can be completed, if the verification fails, the current integrity verification will inevitably fail and one cannot locate the corrupted copy. Furthermore, *Hou, Yu & Hao (2018)* devised a scheme that utilizes algebraic signatures to construct HVT and facilitate replica dynamics. *Long, Li & Peng (2019)* proposed applying chaotic mapping to the construction of full-node AVL trees to achieve replica dynamics. *Wei et al. (2016)* proposed to use fully homomorphic encryption (FHE) to generate multicopy files. Furthermore, *Zhang et al. (2016)* and *Guo et al. (2020)* independently proposed a public auditing scheme using the Merkle tree to achieve replica dynamics. *Zhou et al. (2020)* formalized a dynamic multicopy authentication scheme constructed using certificateless cryptography. To complete the unified management of multiple CSPs, *Wang (2014)* introduced an entity Combiner, which can transfer information between multiple CSPs and TPAs during the audit process. Likewise, *Li, Yan & Zhang (2021)* introduced the Cloud Organizer entity to achieve similar functions. Additionally, to facilitate dynamic operations on replicas, *Zhang et al. (2021)* combined a Merkle tree with a B+ tree to construct an IBM tree. *Zhou et al. (2020)* achieved dynamic data manipulation using certificateless signatures coupled with table structures and Merkle hash trees. *Yi, Wei & Song (2017)* focused on the generation of replica files using fully homomorphic encryption, while *Peng et al. (2019)* contemplated the construction of compressed identity arrays as a homomorphic verification substitute for replicas.

Motivation and contribution

In the MR-PDP scheme, the user first encrypts the outsourced file, and then utilizes the encrypted file to generate multiple replicas and tags set, respectively. These duplicate files along with their respective tag sets are subsequently uploaded to the CSP by the user. This approach has been adopted by the references *Curtmola et al. (2008)*, *Li, Yang & Wu (2017)*, *Barsoum & Hasan (2014)*, *Hou, Yu & Hao (2018)*, *Long, Li & Peng (2019)* and *Wei et al. (2016)*. As shown in Fig. 1, the relationship between encrypted file, duplicate files, and tags set is illustrated.

As Fig. 1 demonstrated, the tags set is derived from the encrypted file and remains independent of the content and quantity of the replica files. Clearly, using this method can greatly reduce computational overhead, especially when dealing with a large number of replicas. Although cloud service providers claim to send replica files and the tag set $< T, F_i >$ to multiple CSPs, even if all the content is sent to the same CSP, it will not affect the normal execution of subsequent data integrity verification. However, if the CSP storing all replica files experiences an outage, the user's cloud replicas will be lost completely, and it will not even be possible to recover the damaged replicas with the help of other replica files, the consequences would be disastrous. Hence, when designing an integrity verification scheme involving multiple replicas, precautions must be taken to prevent the cloud service operator from storing all replicas on the same CSP to avoid irreparable losses to the users.

Based on the above considerations, in this article, we focus on the EHR data and aim to solve the multi-replica synchronization integrity verification problem. The contributions are summarized as follows: (1) We employ identity-based encryption (IBE) to generate the private key and then construct HVT, effectively bypassing the overhead of public key certificates. We combine symmetric encryption and masking technology to generate duplicate EHR files. This method can keep storage safe and enable bad block recovery in the event of replica corruption. (2) Considering that duplicate EHR files are stored in multiple CSPs with diverse geographical locations, therefore, in our proposal, we introduce a crucial entity known as the Cloud Server Manager (CSM) that can act as a 'bridge' between the Patient and various CSPs. The CSM allocates storage servers for multiple copies of the Patient EHR and records the allocation results in the storage distribution table (SDT). In the public auditing phase, the CSM transmits the integrity challenge initiated by the TPA to multiple CSPs and then aggregates the audit proofs returned by the multiple CSPs. However, due to irresistible factors such as channel delay, CSPs in different geographical locations may experience delays in returning audit proof in time. Thus, to ensure the practical implementation, our proposal supports probability audit and provides a specific description. (3) Since the CSPs are untrusted, under the given security model, our proposal can effectively resist forgery attack, replace attack, replay attack, and collusion attack. Lastly, the performance evaluation section validates the feasibility and effectiveness of our scheme.

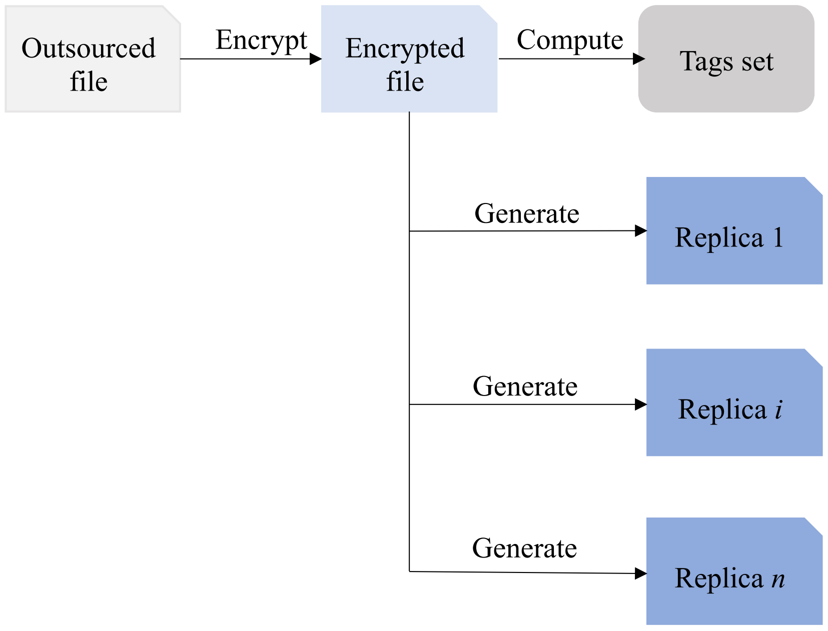

Figure 1 **The relationship between encrypted file, duplicate files and tags set.**
Full-size DOI: 10.7717/peerjcs.1790/fig-1

The remaining sections of this paper are arranged as follows. Preliminaries introduce the system model, design model, notations, and cryptographic knowledges. The next section presents the system components and the security model. The probability audit section describes the auxiliary data structures including the storage distribution table, fault tolerance value, and result record table. Following that, we provide a detailed description of the proposal. Subsequently, the paper presents the security analysis and performance evaluation sections. The final section concludes this paper.

PRELIMINARIES

System model

Our proposal consists of five entities, and the model is shown in Fig. 2. (1) Patient: considering the sensitivity and importance of EHRs data, the Patient produces multiple replicas and uploads them to multiple CSPs in diverse geographical locations and types. The Patient expects that the security and integrity of the replicas can be guaranteed. (2) Cloud Service Manager (CSM): our proposed scheme introduces an important and indispensable entity named CSM, which is equivalent to the 'intermediary' between TPA and multiple CSPs. It allocates storage servers for replica files of the Patient, transmits the integrity challenge launched by the TPA to the multiple CSPs, and aggregates the audit proofs returned by the CSPs. (3) Cloud Service Provider (CSP): the untrusted entity that provides the Patient with data storage services. In the public auditing phase, the CSPs will respond to the integrity challenge initiated by the TPA, calculate, and return the audit proof to the CSM. (4) Private Key Generator (PKG): the trusted entity that generates a reliable private key for the Patient according to its unique identifier. (5) Third-Party Auditor (TPA): the

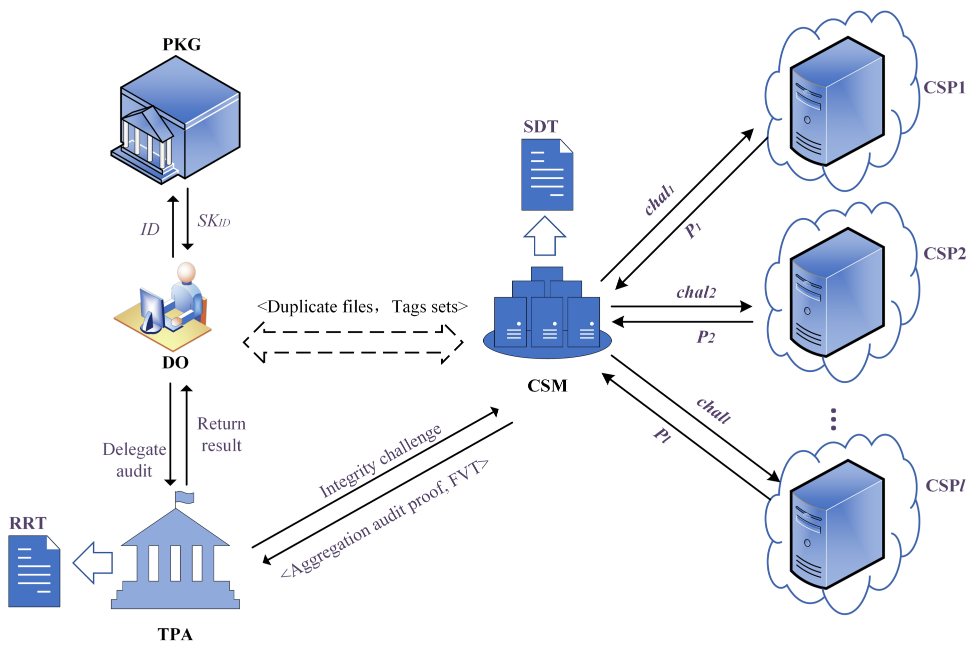

Figure 2 The system model diagram.

Full-size DOI: 10.7717/peerjcs.1790/fig-2

trusted entity performing remote data integrity checks for duplicate files of the Patient after obtaining the authorization.

Design model

Our proposal should achieve the following goals:

(1) **Correctness**: the correctness should include private key correctness and audit correctness.

(a) **Private key correctness**: the private key generated by the PKG will only be accepted after successfully passing the Patient's correctness verification.

(b) **Audit correctness**: the correctness of the aggregation audit proof returned by the CSM can be verified by the TPA. Note that if the FVT returned by the CSM is invalid, the TPA will abort the integrity checking and notify the Patient immediately.

(2) **Resist forgery/replace/replay attack**: our proposal can effectively resist forgery/replace/repay attack.

(3) **Support probability audit**: our proposal supports probabilistic detection based on guaranteeing the storage security of duplicate files.

Notations

We give the notations used in the description of our scheme in Table 1.

Cryptographic knowledge

(1) Bilinear maps

Let G_1, G_2 be multiplicative cyclic groups with the order p, g is a generator of G_1. A bilinear map $e : G_1 \times G_1 \rightarrow G_2$ satisfies the following properties:

Table 1 Notations and descriptions. Symbols used in full text and their descriptions.

Notation	Meaning
p	One large prime
G_1, G_2	Multiplicative cyclic groups
g	A generator of group G_1
e	A bilinear map: $e: G_1 \times G_1 \rightarrow G_2$
Z_P^*	A prime field with nonzero elements
H, H_1, H_2	Cryptographic hash function
pp	The system public parameter
ID	The group user's identity
SK_{ID}	The group user's private key
msk_{ID}, mpk_{ID}	The master secret key and master public key
F	The Patient's encrypted outsourced file
n	The number of replicas
s	The number of the CSP
σ_i	The i-th block's tag, $1 \leq i \leq n$

(a) Bilinearity: $\forall u, v \in G_1$ and $\forall a, b \in Z_P^*$, $e(u^a, v^b) = e(u, v)^{ab}$;
(b) Non-degeneracy: $e(g_1, g_2) \neq 1$;
(c) Computable: there is an efficient algorithm to calculate e.
(2) Security assumptions
Computational Diffie-Hellman assumption. For unknown $\forall a, b \in Z_P^*$, given g, g^a and g^b as input, output $g^{ab} \in G_1$.
Definition 1 (CDH assumption). The advantage of a *PPT* (probabilistic polynomial time) algorithm $\mathcal{A}$ in solving the CDH problem in G_1 defined below is negligible:

$$AdvCDH_{\mathcal{A}} = \Pr[\mathcal{A}(g, g^a, g^b) = g^{ab} : a, b \overset{R}{\leftarrow} Z_P^*].$$

SYSTEM COMPONENTS AND SECURITY MODEL

System components

Our proposed scheme consists of nine algorithms: Setup, KeyGen, ReplicaGen, TagGen, Challenge, ProofGen, ProofAgg, ProofVerify, Compensation. Each algorithm is described as follows.

Setup $(1^k) \rightarrow (pp, mpk, msk)$ is the "System Initialization" algorithm run by the PKG. It takes the security parameter k as input and outputs the system public parameter pp, the master public key mpk and the master secret key msk.

KeyGen $(pp, mpk, msk, ID) \rightarrow SK_{ID}$ is the "Private Key Generation" algorithm run by the PKG. It takes the system public parameter pp, the master public key mpk, the master secret key msk and Patient's identifier ID as input, and outputs the private Patient key SK_{ID}.

ReplicaGen $F' \rightarrow \mathbb{F}$ is the "Replica Files Generation" algorithm run by the Patient. It takes the outsourced EHR file as input and outputs the cloud duplicates.

TagGen $(\mathbb{F}, pp, mpk, SK_{ID}) \rightarrow \mathbb{T}$ is the "Tags Set Generation" algorithm run by the Patient. It takes the duplicate files $\mathbb{F}$, the system public parameter pp, the master public key mpk and the private Patient key SK_{ID} as input, and outputs the tags set for each replica. Then, the Patient sends duplicates $\mathbb{F}$ and all tags sets $\mathbb{T}$ to the CSM and delete the local storage. Following this, the CSM verifies the accuracy of all tag sets. Upon successful verification, it proceeds to allocate them and upload them to the storage server. Subsequently, the CSM records the allocation results in the Storage Distribution Table (SDT).

Challenge is the "Launch Integrity Challenge" algorithm run by the TPA. The TPA periodically generates the integrity challenge *chal* for multiple copies and sends them to the CSM. Upon received, the CSM searches the SDT and transmits integrity challenge set to the corresponding CSPs.

ProofGen is the "Audit Proof Generation" algorithm run by the CSPs. The CSPs receive the challenge message, compute, and respond to the audit proofs to the CSM.

ProofAgg is the "Audit Proof aggregation" algorithm run by the CSM. After receiving the responses, the CSM counts the number of audit proofs, sets the fault tolerance value (FTV) ξ, calculates the aggregation audit proof P_{agg}, and then sends P_{agg}, ξ to the TPA.

ProofVerify is the "Audit Proof Verification" algorithm run by the TPA. After receiving the response from the CSM, the TPA searches the Result Record Table (RRT) to judge the validity of the FVT. Once the FVT is illegal, or the check fails, the TPA aborts and notifies the Patient. Otherwise, the TPA checks the correctness of the aggregation audit proof P_{agg}.

Compensation is the "Claim Compensation" algorithm. The loss or leakage of sensitive information from the EHR cannot be tolerated, and the Patient claims compensation from the cloud service operator after receiving a negative notification.

Remark 1: The responsibilities of the CSM are outlined as follows: (a) validate the correctness of the tags set for each replica file. Only after successful verification, the CSM allocates the storage servers for all copy files and documents the outcomes in the SDT. (b) Upon receiving the integrity challenge launched by the TPA, consult the SDT and forward it to the CSPs. (c) Compute the aggregation audit proof according to the audit proofs and the FVT returned by the CSPs and then reply to the TPA.

Security model

In our proposal, untrustworthy CSPs may launch the following three types of attack models.

(1) **Forgery attack**: During the public auditing phase, if the data block in the replica file stored on the CSP has been damaged due to the CSP's misbehavior, and this corrupted data block is just challenged, then the CSP has to forge this data block and its corresponding tag to pass the TPA's integrity verification.

(2) **Replace attack**: During the public auditing phase, if the data block in the replica file stored on the CSP has been damaged due to the CSP's misbehavior, and this corrupted data block is just a challenge, then the CSP has to replace this data block and its corresponding tag with another intact one to pass the TPA's integrity verification.

(3) **Replay attack**: During the public auditing phase, if the data block in the replica file stored on the CSP has been damaged due to the CSP's misbehavior, and this corrupted

data block is just a challenge, then the CSP returns the audit proof that has been previously checked to pass the TPA's integrity verification.

PROBABILITY AUDIT

To realize probabilistic auditing, the proposed scheme should incorporate some auxiliary data structures, which are described in this section.

Storage distribution table

Since duplicate EHR files are stored on multiple CSPs, the CSM should maintain a storage distribution table (SDT) locally for easy storage management. One Patient corresponds to one SDT, which is used to record the storage servers of each replica. The SDT consists of three columns and its structure is illustrated in Table 2. Replica number (RN) indicates the serial number, where $n(1 \leq i \leq n)$ is the number of the copy files. File identifier (Fid) is the replica identifier. Storage location (SL) indexes the storage location, where $l(1 \leq l \leq s)$ is the number of storage servers.

Remark 2: Take $< RN_i, Fid_i, SL_l >$ as an example to explain the usage of the SDT. RN_i denotes the ith replica, Fid_i is the copy identifier, and SL_l records the storage location. The CSM will assign the CSP to the successfully verified replica and record the result in the SDT.

Fault tolerance value

During the public auditing phase, the TPA executes the challenge-response protocol and launches replica integrity verification. Upon receiving the message from the TPA, the CSM searches the SDT, dispatches the integrity challenge *chal* to the CSPs, initiates a countdown *cd*, and awaits the return of the audit proofs. However, due to the dispersed geographical locations of the CSPs and variations in channel transmission performance, the response times of different CSPs may differ significantly. Consider the following scenario. The CSM transmits the integrity challenge *chal* to *s* CSPs and starts a countdown *cd*. A geographically distant, yet responsive, CSP promptly computes and returns the audit proof upon receiving the challenge. However, due to factors like channel transmission delay, the CSM has not received the response from this CSP when the *cd* expires. In this situation, the CSM faces two challenges: (1) Since the CSM has only received s-1 responses, the aggregation of audit proofs cannot not complete. (2) The response delay is not intentionally caused by this positive CSP, so it is unfair to directly conclude that it is malicious. Thus, to ensure the feasibility in actual deployment, our proposal incorporates a fault-tolerant mechanism, that is, enabling probabilistic auditing. We denote the fault tolerance value (FTV) by $\xi(1 \leq \xi \leq n)$, which also represents the number of audit proofs returned each time during the public auditing. But note that we will not focus on how to determine the FVT, which should be selected according to the actual deployment environment.

Result record table

Since CSPs are untrustworthy, if the results of each public auditing are gathered through probabilistic audit, the security of the proposal will be weakened. Therefore, the TPA

Table 2 **The structure of SDT.** The data structure of storage distribution table (SDT).

RN	***ID***	***SL***
1	Fid_1	CSP_2
2	Fid_2	CSP_l
…	…	…
n	Fid_n	CSP_s

Table 3 **The structure of RRT.** The data structure of result record table (RRT).

chal	**CR**	**PC**	**FTV**
$chal_1$	1	0	None
$chal_2$	1	1	FVT_2
$chal_3$	0	1	FVT_3
$chal_4$	0	0	None
…	…	…	

should record the contents of each check in the result record table (RRT) stored locally, and its structure is shown in Table 3. *Chal* is an integrity challenge set generated by the TPA. Check result (CR) shows the audit result, and probability checking (PC) indicates whether it is a probabilistic verification; if so, the TPA needs write the FVT to the RRT.

To be exact, there are four situations in RRT: $\{\{chal, CR = 1, PC = 0, FVT = None\}$, $\{chal, CR = 1, PC = 1, FVT = \xi\}$, $\{chal, CR = 0, PC = 1, FVT = \xi\}$, $\{chal, CR = 0, PC = 0, FVT = None\}\}$, and we give a detailed discussion.

(1) $\{chal, CR = 1, PC = 0, FVT = None\}$:means that the aggregated audit proof returned by the CSM has passed the TPA's correctness verification, and this checking is not a probabilistic verification.

(2) $\{chal, CR = 1, PC = 1, FVT = \xi\}$: means that the aggregated audit proof returned by the CSM has passed the TPA's correctness verification, but this checking is a probabilistic verification. The FVT indicates the number of CSPs participating in this public auditing.

(3) $\{chal, CR = 0, PC = 1, FVT = \xi\}$ or $\{chal, CR = 0, PC = 0, FVT = None\}$: Since CR $= 0$, it means that the aggregated audit proof returned by the CSM has not passed the TPA' correctness verification. The TPA terminates the check and immediately informs the Patient, and then the Patient runs the *Compensation* algorithm to claim compensation from the cloud service operator. And the highlighted part in Table 3 shows illegal situations.

Remark 3: Emphasize that for audit security, the number of consecutive CSM return probabilistic audit needs to be limited, for example: only 3 consecutive returns are allowed. That is, when the situation $\{chal, CR = 1, PC = 1, FVT = \xi\}$ occurs in the RRT for the fourth time, the TPA will no longer proceed with the follow-up process and immediately inform the Patient.

Remark 4: The TPA will record the relevant information from each check in to the RRT in earnest, and the RRT can be reset at intervals during the actual deployment to save storage space.

In summary, in the public auditing phase, the TPA initiates integrity verification and dispatches the challenge message to the CSM. Then, the CSM transmits the challenge set *chal* to multiple CSPs according to the SDT and starts a countdown *cd*. Upon completion of the *cd*, the CSM calculates the aggregation audit proof P_{agg} based on the audit proofs and FVT replied by the CSPs. Following this, the TPA assesses whether to stop the correctness verification according to the FVT returned by the CSM. If affirmative, the TPA informs the Patient; if not, the TPA proceeds to verify the P_{agg}'s correctness and updates the RRT. Regardless of the verification outcome, the TPA records all information in the RRT.

The proposed scheme

A multi-replicas integrity checking scheme with supporting probability audit for cloud-based IoT are detailed introduced in this section.

Setup

The PKG chooses two multiplicative cyclic groups G_1 and G_2 with prime order p, and g is a generator of G_1. The Patient selects cryptographic hash functions: $H, H_1 : 0, 1^* \rightarrow G_1$ and the bilinear map $e : G_1 \times G_1 \rightarrow G_2$. The PKG selects elements $x \in Z_p^*$, and computes the master secret key $msk = g_1^x$ and master public key $mpk = g^x$. The PKG randomly picks values $u, \mu \in G_1$, publishes the system public parameter $pp = (G_1, G_2, p, e, g, g_1, H, H_1, u, \mu, mpk)$ and holds the master secret key $msk = g_1^x$ private.

KeyGen

Upon receiving the key generation request from the Patient, the PKG performs the following operations: The PKG picks $r_1 \in Z_p^*$ and computes $R_1 = g^{r_1}$. The PKG calculates $R_2 = g_1^x \cdot (u \cdot H(ID))^{r_1}$ according to Patient's identifier $ID \in Z_p^*$. And then the PKG returns $SK = (R_1, R_2)$ to the Patient. After receiving the private key SK, the Patient verifies the correctness according to formula (1):

$$e(R_2, g) \stackrel{?}{=} e(g_1, mpk) \cdot e(u \cdot H(ID), R_1). \tag{1}$$

If (1) equation holds, accept; otherwise, reject it and inform to retransmit.

ReplicaGen

To generate duplicate files, we utilize the symmetric encryption algorithm to obtain the encrypted file and then use PRP to obtain blinding factors corresponding to replica data blocks. That is, the Patient secretly chooses the encryption key K_1 and the PRP key K_2. Given that the outsourced file is F', the Patient encrypts it using a symmetric encryption algorithm (AES, DES, *etc.*) denoted as $F = E_{k1}(F')$. The Patient divides the encrypted file F into m blocks as $F = b'_{j_{1 \leq j \leq m}}$, where m represents the number of data blocks. For each data block $b'_j (1 \leq j \leq m)$, the Patient calculates the blinding factor as $\varpi_{ij} = \psi_{k2}(i||j)$, where i represents the number of the duplicates and ψ_{k2} is the PRP with key k_2, and computes the replica block as $b_{ij} = b'_j + \varpi_{ij}$. Then, the Patient obtains the replica files as $\mathbb{F} = F_{i_{1 \leq i \leq n}} = b_{ij_{1 \leq i \leq n, 1 \leq j \leq m}}$.

TagGen

Since multiple copies are stored on different CSPs in our proposal, the tags set should be generated for each one. The Patient sets the replica identifier $Fid_i \in Z_p^*$, computes $H_1(Fid_i||i)$ and selects a random value $r_2 \in Z_p^*$. For each block $b_{ij}(1 \leq i \leq n, 1 \leq j \leq m)$, the Patient computes tag as $\sigma_{ij} = R_2 \cdot (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2} = g_1^x \cdot (u \cdot H(ID))^{r_1} \cdot (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2}$. Here, the tags set of each replica is denoted as $T_i = \sigma_{ij_{1 \leq i \leq n, 1 \leq j \leq m}}$ and all tags sets are $\mathbb{T} = T_{i_{1 \leq i \leq n}}$. The Patient sends duplicate file and tags sets $\mathbb{F}, \mathbb{T}$ to the CSM and deletes the local storage. Subsequently, the CSM check all T's correctness, allocates CSP, records the allocation result in SDT, and uploads files.

Challenge

The TPA periodically performs remote data integrity checking, records audit results in the RRT, and informs the Patient when necessary. The TPA picks a set Q with c elements, where $Q \subseteq [1, m]$, and generates a set of random value $v_j \in Z_p^*$ for each $j \in Q$. Then the TPA sends $chal = (j, v_j)_{j \in Q}$ to the CSM. After receiving the message, the CSM searches the SDT, sends the integrity challenge $chal = (j, v_j)_{j \in Q}$ to the CSPs, and sets a countdown cd.

ProofGen

Upon receiving the integrity challenge, the CSPs compute the block proof $\lambda_i = \sum_{(j,v_j) \in Q} v_j b_{ij}$ andthe tag proof $\sigma_i = \prod_{(j,v_j) \in Q} \sigma_{ij}^{v_j}$, and then reply to the CSM with the audit proofs $P_i = \lambda_i, \sigma_{i_{1 \leq i \leq n}}$.

ProofAgg

When the response from the CSPs is obtained, the CSM counts the number of audit proofs, updates FTV ξ, and aggregates the block proofs $\lambda_{agg} = \sum_{i=1}^{\xi} \lambda_i (1 \leq i \leq \xi)$ andthe tag proofs $\sigma_{agg} = \prod_{i=1}^{\xi} \sigma_i (1 \leq i \leq \xi)$. Then the CSM returns the aggregation audit proof $P_{agg} = \lambda_{agg}, \sigma_{agg}$ andFVT ξ to the TPA.

ProofVerify

After receiving the response, the TPA looks for the FVT to assess if the FVT complies with the requirements. If the aggregation audit proof returned by the CSM is already the fourth probability verification, the TPA immediately informs the Patient and terminates the verification. Otherwise, the TPA continues and checks the correctness of the aggregation audit proof through Eq. (2):

$$\begin{aligned} e(\sigma_{agg}, g) &= e(g_1, mpk)^{\sum_{(j,v_j) \in Q} v_j} \cdot e(u \cdot H(ID), g^{r_1})^{\sum_{(j,v_j) \in Q} v_j} \\ &\cdot e\left(\prod_{i=1}^{\xi} (H_1(Fid_i||i))^{\sum_{(j,v_j) \in Q} v_j} \cdot \mu^{\lambda_{agg}}, g^{r_2}\right). \end{aligned} \tag{2}$$

If (2) holds, returns '1', which means that the integrity verification of duplicate files is successful and then the TPA records the AR and PC into the RRT. Otherwise, returns '0', the TPA informs the Patient that the duplicate file was damaged, and the Patient runs the *Compensation* algorithm to claim compensation from the cloud service operator.

Compensation

As mentioned above, the Patient runs this algorithm to claim against the cloud service operator when integrity checking fails.

Remark 5: Actually, when replica damage is detected, due to the blinding factor added to the *ReplicaGen* algorithm, a divide-and-conquer method can be used to recover bad blocks. The detailed process is no longer given here.

SECURITY ANALYSIS

Theorem 1 (Private key correctness): The private key generated by the PKG will only be accepted after successfully passing the Patient's correctness verification.

Proof: In *KeyGen* algorithm, after receiving the private key *SK*, the Patient verifies the correctness by checking the validity of formula (1):

$$\begin{aligned} e(R_2,g) &= e(g_1^x \cdot (u \cdot H(ID))^{r_1}, g) \\ &= e(g_1^x, g) \cdot e((u \cdot H(ID))^{r_1}, g) \\ &= e(g_1, g^x) \cdot e(u \cdot H(ID), g^{r_1}) \\ &= e(g_1, mpk) \cdot e(u \cdot H(ID), R_1). \end{aligned}$$

If Eq. (1) holds, the Patient accepts and uses it as the private key. Otherwise, reject it and inform to retransmit.

Theorem 2 (Audit correctness): Only when the CSPs correctly store the Patient's replicas file, during the public auditing phase, the aggregation audit proof P_{agg} generated by the CSM can pass the TPA's correctness verification.

Proof: In the *ProofVerify* algorithm, the TPA validates the correctness of the aggregation audit proof by checking the formula (2):

$$\begin{aligned} e(\sigma_{agg}, g) &= e\left(\prod_{i=1}^{\xi} \sigma_i, g\right) \quad = e\left(\prod_{i=1}^{\xi}\left(\prod_{(j,v_j)\in Q} \sigma_{ij}^{v_j}\right), g\right) \\ &= e\left(\prod_{i=1}^{\xi}\left(\prod_{(j,v_j)\in Q} (R_2 \cdot (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2})^{v_j}\right), g\right) \\ &= e\left(\prod_{i=1}^{\xi}\left(\prod_{(j,v_j)\in Q} R_2^{v_j} \cdot ((H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2})^{v_j}\right), g\right) \\ &= e\left(\prod_{i=1}^{\xi}\prod_{(j,v_j)\in Q} R_2^{v_j} \cdot \prod_{i=1}^{\xi}\prod_{(j,v_j)\in Q} (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2 v_j}, g\right) \\ &= e\left(\prod_{i=1}^{\xi}\prod_{(j,v_j)\in Q} R_2^{v_j}, g\right) \cdot e\left(\prod_{i=1}^{\xi}\prod_{(j,v_j)\in Q} (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2 v_j}, g\right) \\ &= e\left(\prod_{(j,v_j)\in Q} (g_1^x \cdot (u \cdot H(ID))^{r_1})^{v_j}, g\right) \cdot e\left(\prod_{i=1}^{\xi}\prod_{(j,v_j)\in Q} (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{v_j}, g^{r_2}\right) \end{aligned}$$

$$= e\left(\prod_{(j,v_j)\in Q} (g_1^x)^{v_j}, g\right) \cdot e\left(\prod_{(j,v_j)\in Q} (u \cdot H(ID))^{v_j}, g^{r_1}\right)$$

$$\cdot e\left(\prod_{i=1}^{\xi}\left(\prod_{(j,v_j)\in Q} (H_1(Fid_i||i))^{v_j} \cdot \prod_{(j,v_j)\in Q} \mu^{b_{ij}v_j}\right), g^{r_2}\right)$$

$$= e\left(g_1^{\sum_{(j,v_j)\in Q} v_j}, g^x\right) \cdot e\left((u \cdot H(ID))^{\sum_{(j,v_j)\in Q} v_j}, g^{r_1}\right)$$

$$\cdot e\left(\prod_{i=1}^{\xi}(H_1(Fid_i||i))^{\sum_{(j,v_j)\in Q} v_j} \cdot \mu^{\sum_{(j,v_j)\in Q} b_{ij}v_j}, g^{r_2}\right)$$

$$= e(g_1, mpk)^{\sum_{(j,v_j)\in Q} v_j} \cdot e(u \cdot H(ID), g^{r_1})^{\sum_{(j,v_j)\in Q} v_j}$$

$$\cdot e\left(\prod_{i=1}^{\xi}(H_1(Fid_i||i))^{\sum_{(j,v_j)\in Q} v_j} \cdot \prod_{i=1}^{\xi}\mu^{\lambda_i}, g^{r_2}\right)$$

$$= e(g_1, mpk)^{\sum_{(j,v_j)\in Q} v_j} \cdot e(u \cdot H(ID), g^{r_1})^{\sum_{(j,v_j)\in Q} v_j}$$

$$\cdot e\left(\prod_{i=1}^{\xi}(H_1(Fid_i||i))^{\sum_{(j,v_j)\in Q} v_j} \cdot \mu^{\sum_{i=1}^{\xi}\lambda_i}, g^{r_2}\right)$$

$$= e(g_1, mpk)^{\sum_{(j,v_j)\in Q} v_j} \cdot e(u \cdot H(ID), g^{r_1})^{\sum_{(j,v_j)\in Q} v_j}$$

$$\cdot e\left(\prod_{i=1}^{\xi}(H_1(Fid_i||i))^{\sum_{(j,v_j)\in Q} v_j} \cdot \mu^{\lambda_{agg}}, g^{r_2}\right).$$

If (2) holds, it indicates that the integrity verification of duplicate files is successful; otherwise, the TPA informs the Patient.

Theorem 3 (Resist forgery attack): Our proposed scheme can effectively resist forgery attack.

Proof

Suppose that the lth data block of the κth replica has been corrupted, and this data block is just challenged by the TPA during the public auditing phase. As a result, the CSP is compelled to fabricate both the data block and its associated tag in an attempt to deceive the TPA's verification. Denote the intact block and tag as $(b_{\kappa l}, \sigma_{\kappa l})_{1\le\kappa\le n, 1\le l\le m}$, and the forged block and tag as $(\delta_{\kappa l}, \sigma'_{\kappa l})$. Note that, in accordance with the mathematical structure of the tag in our proposed scheme, the CSP can only fabricate the corresponding tag after successfully forging the data block. Then we proceed to analyze the probability that the CSP successfully forges both the data block and its corresponding tag.

Analysis

(1) CSP forges data block

In the *ReplicaGen* phase, to obtain the encrypted file, our proposed scheme utilizes the symmetric encryption algorithm with the key $K_1 \in Z_p^*$. And to obtain blinding factors $\varpi_{\kappa l}$ corresponding to replica data blocks, our proposed scheme uses PRP with the key $K_2 \in Z_p^*$. So, it means that if the CSP can forge a valid data block $b_{\kappa l}$, it must be able to successfully guess K_1 and K_2 with non-negligible probability. But the probability of guessing that K_1 and K_2 are both $1/p$. Furthermore, it implies that the probability of guessing K_1 and K_2 at the same time is $1/p \times 1/p = 1/p^2$, which can be ignored since p is a large prime. Therefore, the probability that the CSP successfully forges a valid data block is $1/p^2$, which is negligible.

(2) CSP forges tag

After the CSP successfully guesses the block with the probability of $1/p^2$, it further attempts to forge the corresponding tag. The tag of valid data block $b_{\kappa l}$ is denoted as

$$\sigma_{\kappa l} = R_2 \cdot (H_1(Fid_i||i) \cdot \mu^{b_{\kappa l}})^{r_2} = g_1^x \cdot (u \cdot H(ID))^{r_1} \cdot (H_1(Fid_i||i) \cdot \mu^{b_{\kappa l}})^{r_2}. \tag{3}$$

The tag of forged block $\delta_{\kappa l}$ denoted as

$$\sigma'_{\kappa l} = R'_2 \cdot (H_2(Fid_i||i) \cdot \mu^{\delta_{\kappa l}})^{r_2} = g_1^{x'} \cdot (u \cdot H(ID))^{r_1} \cdot (H_2(Fid_i||i) \cdot \mu^{\delta_{\kappa l}})^{r_2}. \tag{4}$$

If the CSP can successfully forge the tag, then formula (3) is equal to formula (4), and we have the following:

$$1 = \frac{\sigma_{ij}}{\sigma'_{ij}} = \frac{g_1^x \cdot (u \cdot H(ID))^{r_1} \cdot (H_1(Fid_i||i) \cdot \mu^{b_{ij}})^{r_2}}{g_1^{x'} \cdot (u \cdot H(ID))^{r_1} \cdot (H_1(Fid_i||i) \cdot \mu^{\delta_{ij}})^{r_2}} = \frac{g_1^x \cdot \mu^{b_{ij}r_2}}{g_1^{x'} \cdot \mu^{\delta_{ij}r_2}}. \tag{5}$$

If Eq. (5) holds, it can infer that $g_1^x \cdot \mu^{b_{ij}r_2} = g_1^{x'} \cdot \mu^{\delta_{ij}r_2}$ holds and further, it means that $g_1^x = g_1^{x'}$ holds. Actually, in the *Setup* phase, we know that $x \in Z_p^*$ is randomly picked by the PKG, and g_1^x is the master secret key that its mathematical structure and content are both kept secret and known only to the PKG. Since the PKG is the trusted entity, it means that g_1^x is not forgeable. That is, even if the CSP can successfully forge a valid data block with the probability of $1/p^2$, it cannot forge the corresponding tag.

Thus, from the above series of analysis, it can be seen that our proposed scheme can effectively resist the forgery attack.

Theorem 4 (Resist replace attack): Our proposed scheme can effectively resist the forgery attack.

Proof

Suppose that the lth data block of the κth replica has been corrupted, and this data block is just challenged by the TPA during the public auditing phase. Therefore, the CSP replaces the challenged data block and tag with another in an attempt to pass the TPA's verification. The analysis is similar to Theorem 3, so is omitted here.

Theorem 5 (Resist replay attack): Our proposed scheme can effectively resist forgery attack.

Proof

Suppose that the lth data block of the κth replica has been corrupted, and this data block is just challenged by the TPA during the public auditing phase. Therefore, the CSP returns the audit proof that has been previously checked in an attempt to pass the TPA's verification. The analysis is similar to Theorem 3, so is omitted here.

Table 4 **Notations and meanings.** Notations and meanings used in quantitative analysis and comparison.

Notation	Meaning	Notation	Meaning
H	One hash operation	n	The number of replica file
Mul	One multiplication operation	m	The number of data block
P	One pair operation	δ	FVT
Exp	One exponentiation operation	c	The number of the challenged block
E	One encryption operation	$\|p\|$	The size of an element in Z_p^* or G_1
$\|q\|$	The size of an element in Z_q	–	–

RESULTS

Quantitative analysis and comparison

We first define the symbols used and their meanings as shown in Table 4. To be fair, we set the number of data sectors to 1. Here, we no longer take the addition and PRP operation into account, because they are time-saving in actual deployment. Note that δ represents the fault tolerance value and index the number of audit proofs collected during the audit process.

Computation overhead

The computational overhead comparison with the scheme (*Li, Yan & Zhang, 2021*) is shown in Table 5. The process begins with the Patient encrypting the outsourced file, dividing the data blocks, and applying a random mask for blind operation. This entails time-saving PRP and addition operations, which can be ignored. That is, the overhead of the *ReplicaGen* stage amounts to just one encryption operation. To generate the tag set, the Patient sets the replica identifier F_{id} and executes the hash operation, and then calculates the tag for each block. As a result, the total computational overhead of the *TagGen* stage is denoted by $mnH + 2mnMul + 2mnExp$. In the *ProofGen* phase, the CSPs compute and return the block proof λ_i and tag proof σ_i to CSM. Among them, the calculation of λ_i requires c multiplication operations, and the calculation of σ_i requires nc exponentiation operation and n (2c-1) multiplication operations, so the total overhead is $ncExp + n(2c-1)Mul$. During the *ProofAgg* phase, the CSM aggregated audit proof according to the number of block proofs and tag proofs returned by the CSPs. Since the time-consuming addition operation is not considered, the cost is $(\xi - 1)Mul$. To verify the correctness of the aggregation audit proof, in the *ProofVerify* phase, the TPA leverages FVT to check whether formula (2) holds and the overall calculation cost is $5P + (\xi + 1)H + (\xi + 3)Exp + (\xi + 1)Mul$.

Communication overhead

Table 6 presents a comparison of the communication costs incurred in three stages between our proposal and the scheme (*Li, Yan & Zhang, 2021*). Note that the data fragmentation technique is employed in Scheme (*Li, Yan & Zhang, 2021*), so s represents the number of the data sector. During the integrity challenge phase, the CSM sends $chal$ to n CSPs. Since each challenge occupies $c(|p| + |q|)$, the communication cost for this phase is $cn(|p| + |q|)$.

Table 5 Computational overhead comparison.

Schemes	Phase				
	ReplicaGen	TagGen	ProofGen	ProofAgg	ProofVerify
Scheme (Li, Yan & Zhang, 2021)	nE	$mnH + 3mnMul + 2mnExp$	$cExp + (cn + c - 1)Mul$	$nMul$	$3P + (nc + 1)H + 2(nc + 1)Mul + ncExp$
Our scheme	E	$mnH + 2mnMul + 2mnExp$	$ncExp + n(2c - 1)Mul$	$(\xi - 1)Mul$	$5P + (\xi + 1)H + (\xi + 3)Exp + (\xi + 1)Mul$

Table 6 Communication overhead comparison.

Schemes	Challenge	Response from CSPs	Response to TPA
Scheme (Li, Yan & Zhang, 2021)	$n(3\|p\| + \log 2c)$	$\xi(\|p\| + s\|p\|)$	$(3 + 2s)\|p\|$
Our scheme	$cn(\|p\| + \|q\|)$	$2\xi\|p\|$	$3\|p\|$

After receiving the verification tasks, the CSPs compute and return the adult proof. Due to the adoption of probabilistic audting, the number of adult proofs returned is denoted as ξ, and this phase incurs the communication cost of $2\xi|p|$ bits. The CSM sends the aggregation audit proof along with FVT to the TPA, resulting in a total communication cost of $3|p|$ bits. From Table 6, it can be observed that the communication costs incurred in all three stages of our scheme are lower than those of the scheme (Li, Yan & Zhang, 2021).

Experiments

We run a series of experiments on the 2.80 GHz Intel Core i7 processor and 16.0GB RAM. All the experiments use the Type A with the free Pairing-Based Cryptography (PBC) Library. In the implementation, we selected the file "a.param" as the parameter for the free Pairing- Based Cryptography (PBC) Library. In the experiment, we created a 4M sized data file, with each data block set to a size of 4KB.

In the experimental section, a series of comparisons were conducted between the scheme (Li, Yan & Zhang, 2021) and our proposal. It can be seen from Fig. 3 that the cost incurred in the replica generation stage is similar for both schemes, increasing linearly with the number of replicas between 3s and 4.5s. Figure 4 illustrates that the time required to generate the tag set does not increase with the number of replicas for both schemes, but our proposal requires significantly less time than the scheme (Li, Yan & Zhang, 2021). In the audit proof generation phase, as shown in Fig. 5, our scheme takes more time due to the expensive modular exponentiation operations compared to the scheme (Li, Yan & Zhang, 2021). In the audit proof verification stage, as shown in Fig. 6, the scheme (Li, Yan & Zhang, 2021) exhibits significantly higher time consumption than our proposal, which is consistent with the analysis results in Table 5.

DISCUSSION

This article proposes a multi-replica integrity verification scheme that supports probabilistic auditing, taking into account the context of the Internet of Things (IoT) and shared

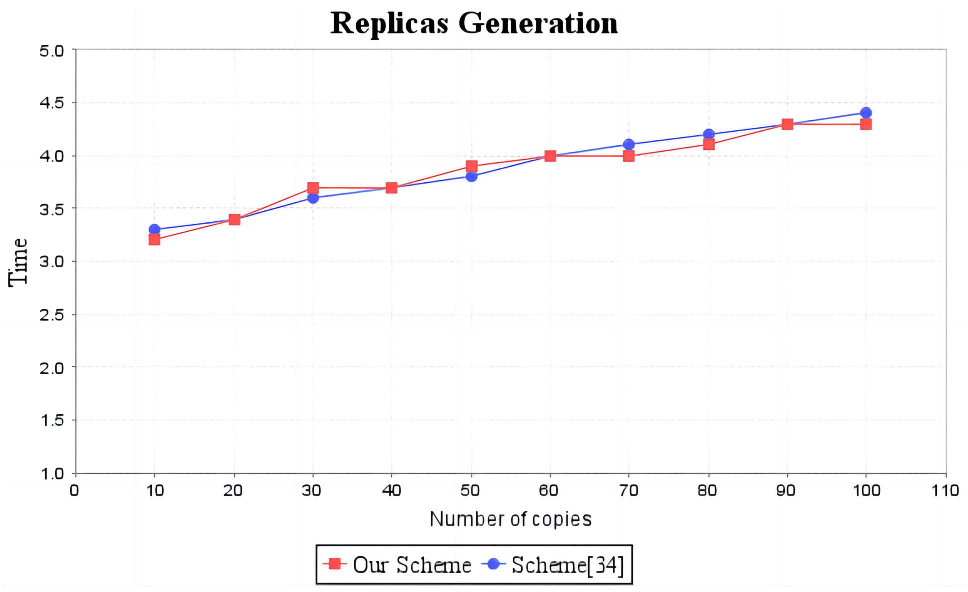

Figure 3 **Computation cost of replicas generation.** The red line shows the change trend of replica generation time of our scheme as the number of replicas increases. The blue line shows the change trend of the copy generation time of scheme (*Li, Yan & Zhang, 2021*) as the number of copies increases.
Full-size DOI: 10.7717/peerjcs.1790/fig-3

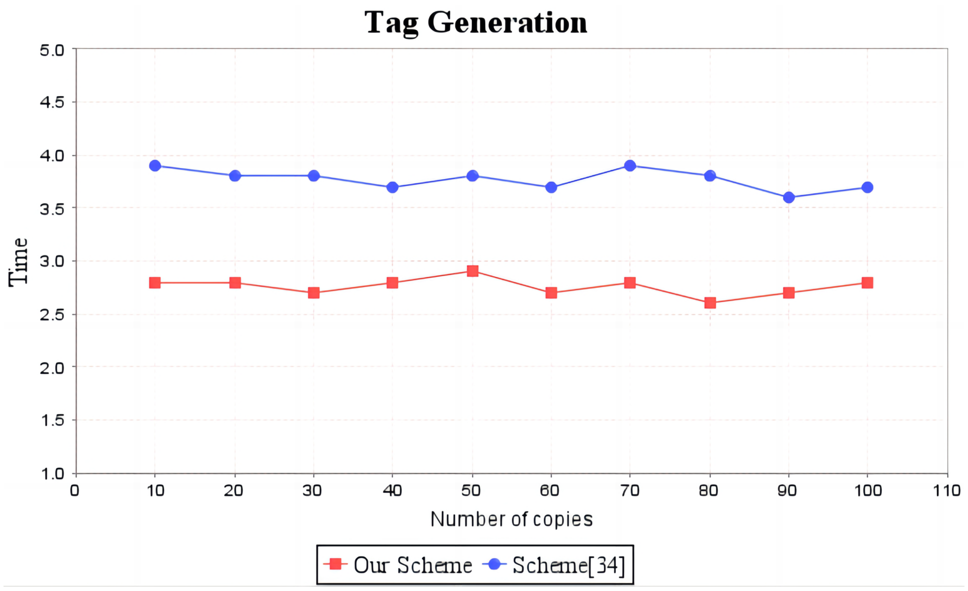

Figure 4 **Computation cost of tag generation.** The red line shows the trend of tag generation time of our scheme as the number of replicas increases. The blue line shows the change trend of the tag generation time of scheme (*Li, Yan & Zhang, 2021*) as the number of replicas increases.
Full-size DOI: 10.7717/peerjcs.1790/fig-4

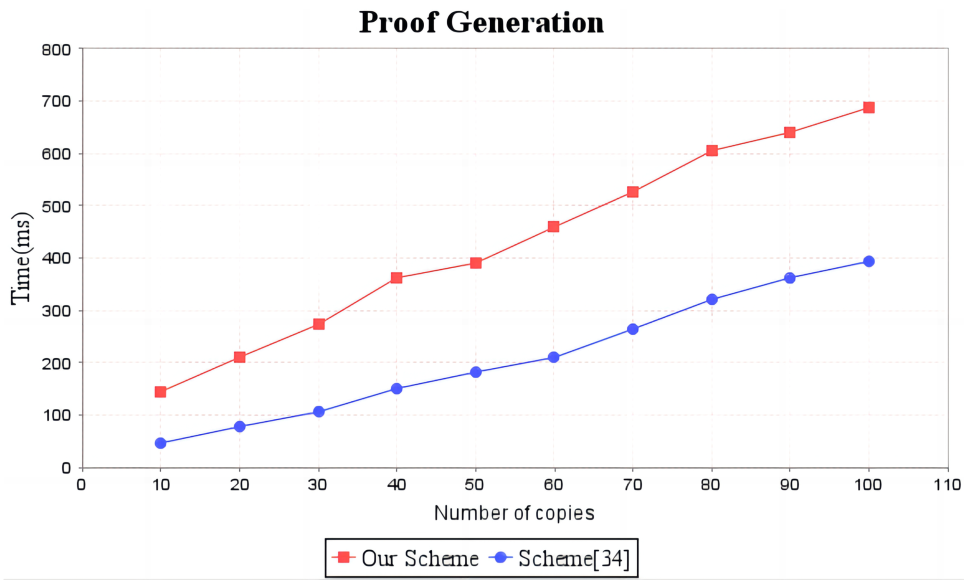

Figure 5 **Computation cost of proof generation.** The red line shows the trend of audit proof generation time in our scheme as the number of replicas increases. The blue line shows the change trend of the audit proof generation time of scheme (*Li, Yan & Zhang, 2021*) with the increase of the number of replicas.
Full-size DOI: 10.7717/peerjcs.1790/fig-5

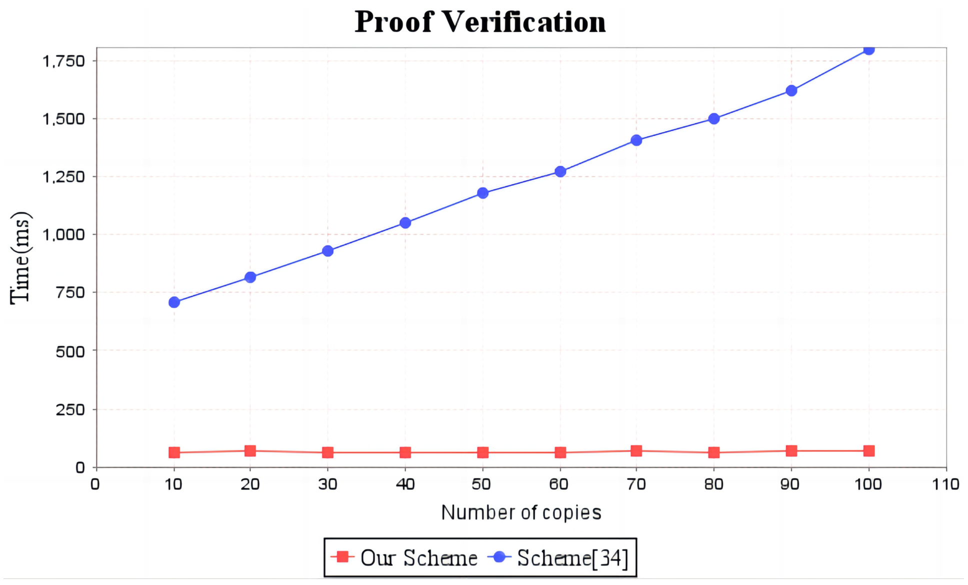

Figure 6 **Computation cost of proof verification.** The red line shows the trend of audit proof verification time of our scheme as the number of replicas increases. The blue line shows the change trend of the audit proof verification time of scheme (*Li, Yan & Zhang, 2021*) with the increase of the number of copies.
Full-size DOI: 10.7717/peerjcs.1790/fig-6

healthcare. The article begins by analyzing critical issues in existing multi-replica integrity verification schemes. The proposed scheme aims to address the problem of synchronization verification of EHR replica files on CSPs located in different geographical locations. We introduce a novel approach called probabilistic auditing, and based on IBE, we generate private keys and construct an HVT, effectively avoiding the overhead of using public key certificates. Under the CDH assumption, the proposed scheme has been proven to be secure and can effectively withstand forgery, replace, and replay attacks. Theoretical analysis and experimental results demonstrate the efficiency and practicality of our scheme. However, when verifying the integrity of replicas on different CSPs, there will inevitably be a trade-off between accuracy and computational or communication costs. In future work, we will focus on addressing this issue and identify effective measures to strike a balance between cost and efficiency.

ADDITIONAL INFORMATION AND DECLARATIONS

Funding

This work was supported by the Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies under Grant 2022B1212010005. The funders had a role in the decision to publish and preparation of the manuscript. The funders had no role in study design and data collection and analysis.

Grant Disclosures

The following grant information was disclosed by the authors:
Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies: 2022B1212010005.

Competing Interests

The authors declare there are no competing interests.

Author Contributions

- Yilin Yuan conceived and designed the experiments, analyzed the data, prepared figures and/or tables, and approved the final draft.
- Fan Yang performed the experiments, performed the computation work, authored or reviewed drafts of the article, and approved the final draft.
- Xiao Wang analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Yimin Tian analyzed the data, authored or reviewed drafts of the article, and approved the final draft.
- Zichen Li analyzed the data, authored or reviewed drafts of the article, and approved the final draft.

Data Availability

The following information was supplied regarding data availability:

The code is available at Zenodo: Yang. (2023). yf0219/Multi-replica-experiment-of-Cloud-storage: First release (code). Zenodo. https://doi.org/10.5281/zenodo.10117378.

REFERENCES

Ateniese G, Burns R, Curtmola R, Herring J, Kissner L, Peterson Z, Song D. 2007. Provable Data Possession at Untrusted Stores. In: *Proceedings of the 14th ACM conference on computer and communications security*. 598–609.

Ateniese G, Dipietro R, Mancini LV, Tsudik G. 2008. Scalable and efficient provable data possession. In: *Proceedings of the 4th international conference on security and privacy in communication networks*. 1–10.

Barsoum A, Hasan M. 2014. Provable multicopy dynamic data possession in cloud computing systems. *IEEE Transactions on Information Forensics and Security* **10(3)**:485–497 DOI 10.1109/TIFS.2014.2384391.

Boneh D, Franklin M. 2001. Identity-based encryption from the weil pairing. In: *Annual international cryptology conference*. Berlin, Heidelberg: Springer, 213–229.

Curtmola R, Khan O, Burns R, Ateniese G. 2008. MR-PDP: multiple-replica provable data possession. In: *2008 the 28th international conference on distributed computing systems*. Piscataway: IEEE, 411–420.

Erway C, Kupcu A, Papamanthou C, Tamassia R. 2009. Dynamic provable data possession. In: *Proceedings of the 16th ACM conference on computer and communications security*. 213–222.

Guo W, Qin S, Gao F, Zhang H, Li W, Jin Z, Wen Q. 2020. Dynamic proof of data possession and replication with tree sharing and batch verification in the cloud. *IEEE Transactions on Services Computing* **15(4)**:1813–1824 DOI 10.1109/TSC.2020.3022812.

He K, Chen J, Yuan Q, Ji S, He D, Du R. 2021. Dynamic group-oriented provable data possession in the cloud. *IEEE Transactions on Dependable and Secure Computing* **3(18)**:1395–1408 DOI 10.1109/TDSC.2019.2925800.

Hou H, Yu J, Hao R. 2018. Provable multiple-replica dynamic data possession for big data storage in cloud computing. *International Journal of Security and Networks* **20(3)**:575–584.

Juels A, Kaliski J. 2007. PORs: proofs of retrievability for large files. In: *Proceedings of the 14th ACM conference on computer and communications security*. 584–597.

Li J, Yan H, Zhang Y. 2020. Identity-based privacy preserving remote data integrity checking for cloud storage. *IEEE Systems Journal* **15(1)**:577–585 DOI 10.1109/JSYST.2020.2978146.

Li J, Yan H, Zhang Y. 2021. Efficient identity-based provable multi-copy data possession in multi-cloud storage. *IEEE Transactions on Cloud Computing* **1(10)**:345–365 DOI 10.1109/TCC.2019.2929045.

Li L, Yang Y, Wu Z. 2017. FMR-PDP: flexible multiple-replica provable data possession in cloud storage. In: *2017 IEEE symposium on computers and communications (ISCC)*. Piscataway: IEEE, 1115–1121.

Li Y, Yu Y, Min G, Susilo W, Ni J, Choo K. 2017. Fuzzy identity-based data integrity auditing for reliable cloud storage systems. *IEEE Transactions on Dependable and Secure Computing* **16**(**1**):72–83.

Long M, Li Y, Peng F. 2019. Dynamic provable data possession of multiple copies in cloud storage based on full-node of AVL tree. *International Journal of Digital Crime and Forensics* **11**(**1**):126–137 DOI 10.4018/IJDCF.2019010110.

Peng S, Zhou F, Li J, Wang Q, Xu Z. 2019. Efficient, dynamic and identity-based remote data integrity checking for multiple replicas. *Journal of Network and Computer Applications* **134**:72–88 DOI 10.1016/j.jnca.2019.02.014.

Shacham H, Waters B. 2008. Compact proofs of retrievability. In: *International conference on the theory and application of cryptology and information security*. Berlin, Heidelberg: Springer, 90–107.

Shamir A. 1984. Identity-based cryptosystems and signature schemes. In: *Advances in cryptology (CRYPTO'84). Santa Barbara, (8)*. 47–53.

Shang T, Zhang F, Chen X, Liu J, Lu X. 2021. Identity-based dynamic data auditing for big data storage. *IEEE Transactions on Big Data* **6**(**7**):913–921 DOI 10.1109/TBDATA.2019.2941882.

Shen W, Qin J, Yu J, Hao R, Hu J. 2018. Enabling identity-based integrity auditing and data sharing with sensitive information hiding for secure cloud storage. *IEEE Transactions on Information Forensics and Security* **14**(**2**):331–346 DOI 10.1109/TIFS.2018.2850312.

Shu J, Zou X, Jia X, Zhang W, Xie R. 2021. Blockchain-based decentralized public auditing for cloud storage. *IEEE Transactions on Cloud Computing* **10**(**4**):2366–2380 DOI 10.1109/TCC.2021.3051622.

Tian H, Chen Y, Chang C, Jiang H, Huang Y, Chen Y, Liu J. 2015. Dynamic-hash-table based public auditing for secure cloud storage. *IEEE Transactions on Services Computing* **10**(**5**):701–714 DOI 10.1109/TSC.2015.2512589.

Tian M, Gao C, Chen J. 2019. Identity-based cloud storage integrity checking from lattices. *Journal of Communication* **40**(**4**):128–139 DOI 10.11959/j.issn.1000-436x.2019073.

Wang H. 2014. Identity-based distributed provable data possession in multicloud storage. *IEEE Transactions on Services Computing* **8**(**2**):328–340 DOI 10.1109/TSC.2014.1.

Wang C, Chow S, Wang Q, Ren K, Lou W. 2011a. Privacy-preserving public auditing for secure cloud storage. *IEEE Transactions on Computers* **62**(**2**):362–375.

Wang H, He D, Tang S. 2016. Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud. *IEEE Transactions on Information Forensics and Security* **11**(**6**):1165–1176 DOI 10.1109/TIFS.2016.2520886.

Wang H, Wang Q, He D. 2021. Blockchain-based private provable data possession. *IEEE Transactions on Dependable and Secure Computing* **5**(**18**):2379–2389 DOI 10.1109/TDSC.2019.2949809.

Wang Q, Wang C, Ren K, Lou W, Li J. 2011b. Enabling public auditability and data dynamics for storage security in cloud computing. *IEEE Transactions on Parallel and Distributed Systems* **22**(**5**):847–859 DOI 10.1109/TPDS.2010.183.

Wang H, Wu Q, Qin B, Domingo-Ferrer J. 2014. Identity-based remote data possession checking in public clouds. *IET Information Security* **8(2)**:114–121 DOI 10.1049/iet-ifs.2012.0271.

Wei J, Liu J, Zhang R, Niu X. 2016. Efficient dynamic replicated data possession checking in distributed cloud storage systems. *International Journal of Distributed Sensor Networks* **12(1)**:1894713–1894723 DOI 10.1155/2016/1894713.

Yi M, Wei J, Song L. 2017. Efficient integrity verification of replicated data in cloud computing system. *Computers & Security* **65**:202–212 DOI 10.1016/j.cose.2016.11.003.

Yu Y, Xue L, Au M, Susilo W, Ni J, Zhang Y, Vasilakos AV, Shen J. 2016. Cloud data integrity checking with an identity-based auditing mechanism from RSA. *Future Generation Computer Systems* **62**:85–91 DOI 10.1016/j.future.2016.02.003.

Zhang J, Dong Q. 2016. Efficient ID-based public auditing for the outsourced data in cloud storage. *Information Sciences* **343**:1–14 DOI 10.1016/j.ins.2015.12.043.

Zhang Y, Lin G, Gu H, Zhuang F, Wei G. 2021. Multi-copy dynamic cloud data auditing model based on IMB tree. *Enterprise Information Systems* **15(2)**:248–269 DOI 10.1080/17517575.2020.1812004.

Zhang Y, Ni J, Tao X, Wang Y, Yu Y. 2016. Provable multiple replication data possession with full dynamics for secure cloud storage. *Concurrency and Computation: Practice and Experience* **28(4)**:1161–1173 DOI 10.1002/cpe.3573.

Zhang J, Wang B, Wang X, Wang H, Xiao S. 2020. New group user based privacy preserving cloud auditing protocol. *Future Generation Computer Systems* **106**:585–594 DOI 10.1016/j.future.2020.01.029.

Zhou L, Fu A, Yang G, Wang H, Zhang Y. 2020. Efficient certificateless multi-copy integrity auditing scheme supporting data dynamics. *IEEE Transactions on Dependable and Secure Computing* **19(2)**:1118–1132 DOI 10.1109/TDSC.2020.3013927.