Enabling Secure and Efficient Decentralized Storage Auditing With Blockchain

Yuefeng Du, *Student Member, IEEE*, Huayi Duan, *Student Member, IEEE*, Anxin Zhou, *Student Member, IEEE*, Cong Wang, *Fellow, IEEE*, Man Ho Au, *Member, IEEE*, and Qian Wang, *Senior Member, IEEE*

Abstract—As a promising alternative solution to cloud storage, decentralized storage networks (DSN) are widely anticipated to develop continuously and reshape the storage market share in the foreseeable future. In particular, one of the most important research problems is how to enforce the quality of service (QoS) in the context of storage solutions. Despite plenty of auditing-related works in the context of cloud storage, none of them can be directly applied to the decentralized storage paradigm. The challenges of designing a feasible storage auditing framework emanate from two aspects: 1) security problems unique to the decentralized settings and 2) performance overhead due to on-chain operations. In this article, we first put forward a basic storage auditing framework that satisfies the security and efficiency requirements, and outperforms the existing approaches. We also identify a critical and overlooked security problem that would compromise the integrity of storage auditing solutions in the blockchain paradigm. With our refined storage auditing design based on customized zero knowledge protocols, we propose a convenient mitigation solution in our revised security model. The evaluation results confirm that our solution would only incur a 10–15 percent increase in the overall auditing costs for common usage scenarios, compared to the basic design.

Index Terms—Decentralized storage, public auditability, blockchain-enabled auditing, smart contract, lightweight verification, zero knowledge, storage outsourceability

1 Introduction

Decentralized storage network (DSN), as a useful supplement to cloud storage, witnesses massive deployment in the industry, a trend well underway in the recent years driven by the evolving blockchain technology. Initially in the 2000s, peer-to-peer storage designs such as Bittorrent [1] first promoted the concept of distributed storage. The Bittorrent-alike system consists of storage peers, who can join and leave the system voluntarily. Without mandatory compliance, such a volunteer-based storage ecosystem is only commensurate with the storage of public data. In order to mitigate the loss of leaving peers at any moment, the system needs to press as many peers as possible to store a replica of data. In the end, only those public data of high popularity would be distributed with many replicas in the volunteer-based storage ecosystem. The booming development of blockchain technology, however, is foreseeable to reshape the ecosystem of distributed storage. With an arbitrator (blockchain consensus) and a proper incentive system to fulfill the service-level agreements, the blockchain technology promises the rise of a new paradigm of decentralized storage. Aside from public data, for the first time users can safely store their private data in decentralized storage nodes. On the other hand, ordinary users with idle devices instead of centralized corporations can provide storage services and share the monetary rewards.

Despite the promises, the surge of blockchain-enabled decentralized storage solutions is also naturally accompanied with many challenges. As commonly acknowledged by all DS proposals today, the openness of this decentralized paradigm mandates effective auditing mechanisms to assure owners of their remotely stored data integrity. Though the problem is thoroughly discussed by plenty of prior work [2], [3], [4], [5] in the context of centralized cloud storage, the proposed data auditing designs are not entirely applicable to the decentralized situation.

In our preliminary conference version of this paper [6], we discover that the major hindrances in the decentralized paradigm would be the security and efficiency considerations caused by the immutability and public accessibility of blockchain. More specifically, in terms of auditing security, we have addressed the potential exploitation of the on-chain auditing trails for unsolicited data recovery. However, as we will extensively explain in this paper, it is also indispensable to handle another unforeseen threat risen from the prevalent outsourcing behaviours in the decentralized paradigm. This security problem empowers storage providers to compromise the integrity of the storage auditing scheme. Accordingly, we propose a zero knowledge storage auditing scheme with an on-chain check mechanism to address the storage freeriding

- *Yuefeng Du, Huayi Duan, Anxin Zhou, and Cong Wang are with the Department of Computer Science, City University of Hong Kong, Hong Kong SAR, China, and also with the City University of Hong Kong Shenzhen Research Institute, Shenzhen 518057, China. E-mail: {yf.du, hy.duan, ax.z}@my.cityu.edu.hk, congwang@cityu.edu.hk.*
- *Man Ho Au is with the University of Hong Kong, Hong Kong SAR, China. E-mail: allenau@cs.hku.hk.*
- *Qian Wang is with the Wuhan University, Wuhan 430070, China. E-mail: qianwang@whu.edu.cn.*

Manuscript received 15 Nov. 2020; revised 7 May 2021; accepted 11 May 2021. Date of publication 19 May 2021; date of current version 2 Sept. 2022. (Corresponding author: Cong Wang.)
Digital Object Identifier no. 10.1109/TDSC.2021.3081826

security problem in the settings of rational actors. In addition to the identified new security problem, we also endeavor to improve the overall auditing efficiency on the basis of prior results derived from the conference paper. To this end, we propose a more secure and affordable storage auditing scheme with enhanced security guarantees and improved overall efficiency.

1.1 Our Contributions

The improvements of the results compared to the conference version of the paper and the main contributions can be summarized as follows:

- We prioritize the main design goals of decentralized storage auditing in terms of efficiency, scalability, and security trade-offs by investigating blockchain-assisted auditing frameworks from the constantly evolving practices.
- We identify a new problem unique in the context of blockchain-assisted auditing framework. The problem would compromise the integrity of the storage auditing scheme in the rational settings.
- We provide mitigation strategies based on our revised zero knowledge auditing scheme to prevent storage providers from abusing this problem.
- We optimize our proof-of-concept prototype and dramatically boost the auditing efficiency (both on-chain and off-chain) on the basis of our prior results.
- We provide new insights on how to extend our storage auditing scheme to the dynamic settings and further improve usability from the perspective of participants.

We emphasize that a small portions of the work presented in this paper have previously appeared in [6]. But we have revised the paper tremendously and improved many technical details as compared to [6]. Below we briefly summarize the primary improvements: First, we provide more details with regard to storage auditing primitives in Section 2, including the latest results in this emerging field. Second, in Section 5.2, we propose a storage freeriding strategy that is able to compromise the integrity of the storage auditing design in our prior work and in Section 5.3, we propose proper mitigation solutions. Third, we provide a complete security framework for our identified security problem in Section 6.2. Fourth, in Section 7, we study the feasibility of our proposed attack and demonstrate the effectiveness of our mitigation solution based on experiments on our new auditing design. Lastly, we also substantially polish the overall writing and add numerous new contents in remaining parts of this paper.

2 Related Work

2.1 Storage Integrity Schemes

Numerous research has been done upon storage auditing in the past decade. The most naive storage guarantee method leverages the standard hash function [7] or message authentication codes (MAC) [8] to check the data integrity at an untrusted location. To prove the possession of data, the storage provider is supposed to compute the MAC or algebraic signatures [9] on the block level for each round of spot checking [10]. However, these techniques incur a significant amount of communication overhead linear to the number of blocks. They also only serve for the application scenario of private verification with pre-generated randomness.

Later work [11] uses more efficient data structures such as Merkle tree to reduce communication and verification overhead mentioned above. Though auditing schemes based on Merkle tree can support public verification, the major limitation that hinders the wide deployment of such schemes is that they can support limited challenges and would eventually run out of fresh blocks to be challenged. Nonetheless, it still brings up fruitful applications in the existing DSN auditing constructions [12], [13].

2.2 Proof of Data Possession/Retrievability

Another line of work dedicated to efficient auditing schemes is built upon public-key cryptography. Concretely, the auditing schemes are constructed on the basis of homomorphic linear authenticator (HLA) [8], [14], which aggregates the signatures of blocks and thus produces succinct proofs. Compared to the easy-to-implement auditing schemes mentioned above, HLA-based auditing schemes support unbounded usage due to unlimited challenge freshness. These schemes are developed under the two mainstream models Provable Data Possession (PDP) [8], [10], [15], [16], [17] and Proof of Retrievability (POR) [14], [18], [19], [20], [21]. They provide high probabilistic assurance of a remote file's integrity through verifying random chunks of the file. POR additionally ensures that the file can be retrieved in its entirety, typically by applying erasure coding. As a salient feature, most of these schemes produce compact proofs with a size irrespective of the number of challenged chunks in an audit. Later on, the developments in the field of public verification [3], [14] and dynamism [3], [22], [23] have led to a renewed interest in the literature of storage auditing and its further application in the cloud storage scenario.

In addition to conventional Proof of Storage, Curtmola *et al.* [24] propose a PDP scheme that allows a client to audit file replicas stored across multiple servers in a way more efficient than auditing them individually. Hail [25] achieve similar guarantees with public auditability.

One of the first work to consider storage auditing in the decentralized settings is Audita [26]. Similar to our proposed storage auditing service, Audita adopts the strategy of frequent storage auditing. However, the focus of Audita's design is around the compatibility of the underlying storage auditing scheme and the incentive mechanism. To enable PoR for decentralized storage, a recent work [27] first proposes a construction built upon accumulators. The intention of accumulator-based constructions is to utilize the expressiveness of accumulators to achieve position binding and aggregatable authenticators simultaneously instead of treating them independently. With this flexible primitive, it would be more convenient to achieve storage auditing with dynamic data that is frequently updated. In spite of the gratifying progress, its poor concrete efficiency, especially in terms of its intrinsic computational penalties brought by accumulators, makes it more of theoretical interest in its early development. Therefore, most mature storage auditing schemes are still constructed using homomorphic linear authenticators, together with authenticated data block indexing.

Our work separates from the above works, for we are the among the first to address core *security issues* and *concrete efficiency*, which are largely overlooked by prior works. As an early attempt to facilitate storage auditing in the decentralized settings, we only consider storage auditing for static data, such as long-term data archiving in this paper, which is also of great interest in practice.

2.3 Proof of Storage Time and Proof of Replication

Storage auditing that leverages the primitive of Proof of Storage only guarantees data possession at the time the storage provider generates the proof. To strengthen assurance of data storage and quality of storage service, continuous storage auditing is a must. In light of this, recent study [28] proposes the concept of Proof of Storage Time (PoST), which aims to certify storage guarantees within a certain time period. Despite its reduced setup and verification cost, the proposed PoST construction inevitably incurs a substantial amount of overhead upon proof generation procedures. In addition, it is inherently not suitable for the efficient support of dynamic data, which is an open challenge today.

Orthogonal to certifying storage, Proof of Storage Time protocols can also be used to build more efficient and sustainable consensus algorithms for the next generation of blockchain platforms. Most notably, Filecoin [13] proposes its own PoST construction to incentivized miners, with integration of zksnark to create succinct and easy-to-verify proofs. Despite certain progress, the main performance bottleneck is the notoriously cumbersome preprocessing and proof generation [29]. While such overhead might be tolerable for leader election in consensus, it's not applicable to our auditing scenarios, where we believe the auditing overhead to storage providers should not outweigh the normal storage serving purposes. Moreover, how to handle data dynamics remains unclear for this line of constructions.

As another orthogonal line of work, there has been a surge of interest [30], [31], [32], [33] in constructing more strict storage schemes that guarantees multiple replications of data in face of colluded storage providers, especially for public (i.e., not encrypted) data. Intuitively, in order to guarantee data replication, data owners are required to encode data in an incompressible form before outsourcing, such that easy replication will become unique for checking. We stress that our proposed research does not consider the verification of multiple data replications, but focus on pushing the frontier of decentralized on-chain auditing with single data copy. Our research findings will likely serve as the underlying component to also advance the understanding for more applicable constructions of proof of replication.

2.4 State-of-the-Art DSNs

The blockchain technology equips the industry-deployed DSNs with a mature incentive mechanism as well as an arbitration system. Almost every DSN has a inherent storage auditing system, with which wrongdoers are penalized with fines while honest parties are rewarded with storage fees. As a representative of early DSN projects, Siacoin [12] first use Merkle tree based auditing schemes. Though Sia takes a more decentralized approach in its White paper design, the current implementation in practice still relies on centralized managements when it comes to storage auditing services.

TABLE 1
Summary of Different Desirable Goals of Decentralized Storage Auditing Frameworks

	Merkle tree		SNARK	Homomorphic tag	
	Storj	Sia	Filecoin	[6]	This work
Public Audit	✓ ⇒ ×	✓	✓	✓	✓
Unltd. Audit	×	×	✓	✓	✓
Data Rec. Risk	High	High	Low	Medium	Low
Proof Size	Medium	Medium	Small	Small	Small
Quick Ver.	✓	✓	✓	✓	✓
Setup Req.	PA	PA	PB	PA	PA
Prover Eff.	✓	✓	×	✓	✓
Scalability	×	×	✓	×	×

[a] *PA stands for private preprocessing protocols while PB stands for distributed preprocessing protocols.*

Concurrent DSN project Storj [34] also adopts a similar auditing scheme in its early stage. To support unlimited number of challenges, Storj later shifts to a private auditing framework, where dedicated third-party auditors called satellites perform the data audit. Though the new auditing framework brings in user convenience and auditing efficiency, it would inevitably lead to very essential trust issues in the decentralized world, as the third-party auditors can be bribed and compromised. We note Storj is one of the very DSNs that adopts a private storage auditing framework based on a reputation system, which to much extent is still an open question in the decentralized paradigm.

Compared with private auditing that may lead to complicated resolution procedures and potential collusion problems, it is more reasonable to publish audit trails to the public accessible blockchain and every participant verify the audit trails. To alleviate the inherent constraint of limited auditing rounds brought by Merkle tree based auditing schemes, Filecoin [13] adopts a generic succinct proof framework to produce unlimited proofs to be audited. In contrast to traditional succinct proof frameworks based on probabilistic checkable proofs [35], the recent development of so-called SNARK frameworks [36], [37], [38], [39] makes it much more practical to be used in decentralized applications. One of the primary advantage brought by SNARK frameworks is the concrete on-chain verification efficiency. The proof size as well as verification time is minimized to a usable level with little verification cost. The major overhead incurred in the SNARK frameworks, accounts for the trusted setup phase and the proof generation phase. The trusted setup phase is meant to produce structured reference strings consisting of fresh proving keys and verification keys. With a universal updatable structured reference strings, the prover is able to produce a combined storage proof for data contributed by different data owners. In spite of recent efforts on constructing updatable structured reference strings [40], [41], the prover still has to perform heavy workload in order to produce a single proof.

In Table 1, we provide a full comparison of different candidate schemes for a secure, efficient, and scalable auditing framework. Due to the intrinsic problem that challenge freshness would run out eventually, Merkle tree based auditing schemes are not suitable to serve for long-term storage solutions. As a direct consequence of limited challenge

randomness, the data recovery problem would be more severe with Merkle tree based auditing schemes, as each proof includes the original data block for opening the committed Merkle root.

In contrast to Merkle tree based auditing schemes, SNARK-based auditing schemes bear low risk of data recovery, as long as the setup protocol is completed by the honest majority (proving/verification key pair would be different in each round of storage auditing). The essential challenge that hinders the further deployment of SNARK-based auditing protocol, which is acknowledged by the Filecoin project, is the inherent performance overhead. For projects like Filecoin, this property satisfies their actual needs of building a sustainable consensus algorithm in the decentralized settings. But in the usage scenario of constructing a secure and efficient storage auditing framework, this characteristic makes it hard to compete with other customized solutions. Indeed, a more practical solution is to leverage the homomorphic tags based auditing schemes. Note that the storage provider has to provide proofs for each of data owner by default, as each data owner generates its own public key for data verification. In addition, we also conduct comparison with regard to the security problem that we refer as data recovery risk here and we will later demonstrate. To decrease this risk when using homomorphic tags based auditing schemes, we propose an improved auditing scheme that would mitigate the risk caused by the data recovery problem.

3 Overview

Most of the existing works provide their specialized and customized blockchain system and consequently blockchain-enabled storage auditing enabled in these works has to suit various demands. This leaves it extremely difficult to analyze the underlying auditing designs across different blockchain systems. In this work, we study the compatible storage auditing framework that can be used on the generic blockchain model such as Bitcoin [42] and Ethereum [43].

*Smart Contract as an Auditor.*We leverage the smart contract to generalize the power of executing highly expressive programs without any individual off-chain party. Intuitively, smart contract is a self-enforcing agreement stored on the blockchain. When pre-defined rules are met, the agreement will be automatically enforced. Smart contract is widely used to establish trust in the decentralized context and certainly can be used for decentralized auditing. More formally, the on-chain computation achieves quasi-Turing completeness, whose execution cost is bounded by the predetermined gas cost. We stress this generic model can certainly be adapted to customized blockchain systems.

3.1 Storage Procedures

Before we proceed to the auditing procedures, we first provide background information of the concrete storage procedures. In a DSN, there are two major parties aside from the blockchain platform, namely the data owner who pays for the outsource of encrypted data and the storage provider who get paid for providing storage. To give readers an overarching picture of the storage procedures in the decentralized paradigm, we summarize four concrete phases.

1) *Negotiation.* Through a negotiation phase, the data owner and the storage provider achieves the consensus of executing a storage contract, which follows a generic smart contract template and specifies requirements such as storage duration, storage fees, and penalty fines.
2) *Encryption.* By default, the data owner use symmetric key encryption to encrypt the original data before further processing. Unlike cloud storage solutions that mostly introduce data encryption on the server side, encryption is performed on the client side in the decentralized storage paradigm for stronger security guarantees.
3) *Chunkenization & Coding.* In order to ensure high availability, data chunkenization and erasure coding need to be applied to data after encryption. Compared to data replication, erasure coding based availability solutions save tremendous amount of bandwidth cost, which is a major bottleneck that constrains the scale of DSNs.
4) *Distribution.* After data processing is completed on the side of data owner, each chunk of data is routed to the storage provider in a distributed manner. A commonly used algorithm that can efficiently achieve peer-to-peer routing is distributed hash table [44].

The second phase *Encryption* is vital to ensure that the storage provider can never access the underlying data encrypted by the data owner. The third phase *Chunkenization & Coding*, on the other hand, is to increase the chance that the data owner can access his stored data at any time in case of unpredicted events such as unexpected network turbulence. Due to the existence of the third phase *Chunkenization & Coding*, the original data is expanded to coded data with a much larger size and the auditing is performed upon the coded data. Hence, the auditing cost is linear to the number of storage providers where a data owner stores data. In practice, this number is usually no smaller than three. For simplicity and without loss of generality, in this paper, we consider a simplified auditing scenario where a smart contract responsible for storage auditing binds a single data owner and a single storage provider.

3.2 Auditing Procedures

Parallel to the storage procedures mentioned above, the data owner can also pre-process the coded data before the phase of *Distribution* for the preparation of the periodical auditing. We will provide more details of this data pre-processing stage in Section 5.1 which describes the auditing details.

After the data owner outsources his data to the storage provider, the data is audited according to the negotiated results specified by the agreed-on smart contract. In each cycle of data auditing, the blockchain releases the randomness generated in a distributed manner for the challenging purpose. The storage provider is required to compute the auditing proof for the challenge and submit the proof to the blockchain in time. The smart contract, which has the ability to verify the auditing proof, automatically completes the proof verification procedures. After the expiration of the storage contract as specified in the smart contract, it would also derive a number from each of the auditing verification outcomes that the storage provider is supposed to pay as

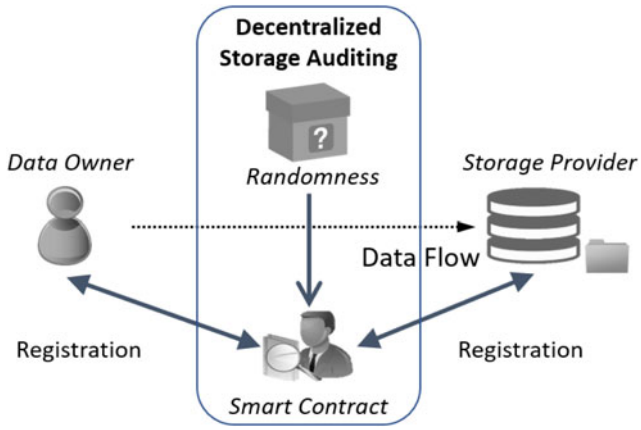


Fig. 1. Overview of blockchain-assisted DSN auditing.

penalty or receive as storage fees. As displayed in Fig. 1, the smart contract bridges the auditing procedures with the underlying arbitration system and the incentive system.

3.3 Adversarial Model

For the decentralized environment, it is more reasonable to consider rational participants rather than completely malicious participants. In the settings of DSN, our design purpose is to make sure that honest behavior specified in the contract is rewarded and dishonest behavior is punished through cryptocurrencies like Ethereum. Malicious adversaries who would not make rational economic decisions are out of the scope of this paper.

More concretely, we strive to achieve a ϵ-Nash equilibrium for the data owner and the storage provider in the DSN. In its essence, both parties are incentivized to earn more monetary rewards in the sense of Ethereum tokens. For instance, the data owner could sabotage the data preprocessing phase for the auditing such that the storage provider cannot generate a valid proof and thus is punished in every round of auditing. On the other hand, the storage provider could be bribed to discard partial data and is also inherently incentivized to save storage for more rewards.

Aside from the equilibrium between the data owner and the storage provider, we also carefully consider the potential collaboration between a storage provider and other parties. For instance, an individual storage provider could further outsource the data to the cloud. We emphasize that such behavior is allowed as long as the data owner can eventually recover its data. In the long run, it is often the case that the storage provider gains no leverage by further outsourcing data to a third party, as the third party can directly participate as the storage provider. Nonetheless, an exception to the above analysis is that a storage provider could *freeride* the transparent blockchain for partial storage outsourcing with *zero marginal cost*. We will elaborate on this inherent problem of the decentralized auditing framework in Section 5.2 and focus squarely on forming up a robust and coercive apparatus immune to the attacks based on the storage freeriding strategy.

3.4 Desirable Goals

In accordance with the analysis of the industry-deployed DSN practices, we prioritize the desirable goals of an effective auditing framework for DSN.

- *Strong data integrity*. Strong guarantees that the storage provider faithfully stores the outsourced data provided by the data owner is of upmost importance. Compared to the typical settings in the cloud storage, we need to additionally assure that even with the transparent blockchain publicly accessible, the storage provider cannot leverage such public information to generate auditing proofs.
- *On-chain efficiency*. Inherent cost due to on-chain storage and computation is inevitable in a fair and robust decentralized storage environment. The on-chain verification procedures account for the majority of the inherent cost and is desired to be minimized. Aside from on-chain verification procedures, we also need to come up with an economic way of generating trustworthy randomness in a distributed manner.
- *Off-chain scalability*. Most auditing frameworks sacrifice the performance during the proof generation phase in pursuit of an efficient verification phase. However, in DSN, the storage provider often stores data from a considerable amount of data owners and has to audit for multiple times (data from different data owners) in one round of auditing cycle. Hence, it is also desirable to propose an auditing framework, where the off-chain proof generation procedures are able to scale to serve a large amount of data owners simultaneously.

4 Preliminaries

4.1 Proof of Storage Definitions

A Proof of Storage scheme [45] consists of a generation algorithm *Gen* for data pre-processing and an interactive protocol between the prover P and the verifier V for auditing. We denote the data to be outsourced as $F \in \{0,1\}^*$, which embodies the parity codes for the increase of availability. The generation algorithm *Gen* which takes in the security parameter λ outputs public key information pk and corresponding private key information sk, which is kept from the public and the prover. With the private key information sk, the generation algorithm *Gen* also outputs pre-processed data $\tilde{F}$ from F.

During the challenge-response interaction phase, the verifier issues random challenge c, with which the prover generates a proof ver from $\tilde{F}$. The verifier then examines ver with public key pk and challenge c, and outputs a verdict $\in \{0,1\}$ indicating an accept or a reject of the proof.

Essentially, any proof of storage scheme can be regarded as a proof of knowledge protocol. More formally, the completeness of the protocol shows that for any interaction $\{P(\tilde{F}) \leftrightarrow V(ver)\}$, verifier always outputs the accept verdict. On the other hand, as for the soundness of the protocol, we can consider an adversary $\mathcal{A}$. From a high level, the soundness of the protocol requires that the verifier can run an extractor ϵ on $\mathcal{A}$ to cover partial data of F. More specifically, there exists an extractor ϵ and a negligible function that should cover δ-fraction of F, for any adversary $\mathcal{A}$ and any proof provided by $\mathcal{A}$.

4.2 Polynomial Commitment

Since the proposal of first proof of storage scheme, many following works have built proof of storage schemes based

on homomorphic linear authenticators (HLA). However, one of the notable impediments that hinders the use of most proposed designs in practice is the insufferable amount of data pre-processing time for users. Moreover, in the settings of DSN, most HLA-based schemes are also impractical to be adopted, because they need to incur either a considerable amount of storage overhead ($\tilde{F}$ for verification) or alternatively the communication overhead.

Recently, the polynomial commitment [46] has been widely applied to construct new cryptographic primitives [47], [48]. For the application of proof of storage, it is displayed [5], [18] that the polynomial commitment would provide benefits for multi-accesses of block data. Essentially, polynomial commitment can bind multiple data blocks to a single commitment, while it only requires to check one random point during the proof generation stage.

More concretely, the polynomial commitment proposed by Kate *et al.* (KZG) uses a degree n polynomial $f(X)$ to commit a vector. The security of KZG is based on the q-BSDH assumption [49], [50]. Compared to other commitment schemes, the major advantage of KZG is that a succinct evaluation proof of $f(a)$ for any a is easy to verify due to polynomial remainder theorem, and meanwhile $f(a)$ encapsulates a vector of message . The only cost accompanied with this advantage is the setup cost of generating a set of public parameters g^{τ^i}, where $i \in [0, n]$, g is a group generator, and τ is a trapdoor. Aside from the computational binding property based on q-BSDH assumption, we can also achieve computational hiding under the discrete log assumption [46]. From a technical point of view, KZG commitment consists of three functions.

- *Committing.* Following the derived public parameters above, let $f(X)$ be a polynomial of degree n comprised of coefficients $c_0, c_1, \ldots, c_n \in \mathbb{Z}_p$. After the committing procedures, a single group element $cm = \prod_{i=0}^{n}(g^{\tau^i})^{c_i} = g^{f(\tau)}$.
- *Witness creation.* To create a witness for the commitment, the prover can compute a evaluation proof such that $f(a) = y$. Given the polynomial remainder theorem, i.e., the sufficient and necessary condition for $f(a) = y$ is that $\exists h(X)$ such that $f(X) - y = h(X)(X - a)$, the prover can simply use the polynomial commitment to $h(X)$: $\pi = g^{h(\tau)}$.
- *Evaluation verification.* To verify the evaluation proof π, one simply checks whether the following bilinear pairing relation holds: $e(cm \cdot g^{-y}, g) = e(\pi, g^{\tau - a})$.

In our auditing scheme, we will leverage the *committing* phase for data preprocessing, the *witness creation* phase for proof generation, and the *evaluation verification* phase for proof verification.

4.3 Notions

We view the outsourced data $\tilde{F}$ as a set of data blocks $m_{i,j} \in \mathbb{Z}_p$ where $i \in [0, s], j \in [0, d]$ and p is a sufficiently large prime number with proper security strength specified by the security parameter λ. As illustrated in Fig. 2, from a high level, every s data blocks form a data *chunk* that produces a single polynomial commitment for auditing-assisted auxiliary data during the pre-processing phase. This reduces the required time for data pre-processing as well as the size of auxiliary data.

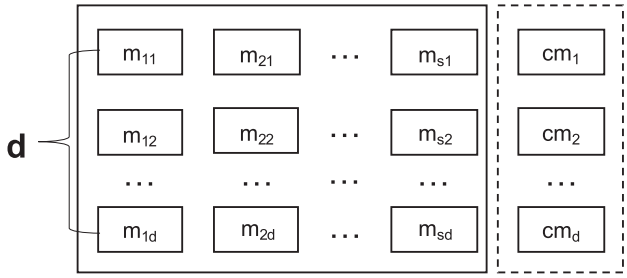


Fig. 2. Polynomial commitment as auxiliary metadata.

More notions are displayed in Table 2 with a brief explanation attached for ease of presentation. Aside from the meta information of the data to be stored, we additionally describe the notions prevalently used in the auditing framework. In our main design, we present more details for these summarized notions related to the auditing scheme.

5 Our Auditing Framework

5.1 Basic Auditing Scheme

Pre-Precessing Beforehand. Applying the primitive of polynomial commitment, the data owner needs to generate data tags in the form of polynomial commitments, as displayed in Fig. 2. In addition to the tag generation procedures, the storage provider also needs to verify the integrity of the generated data tags before the storage contract takes effect. We now describe the detailed steps of the three phases, namely key generation, tag generation and tag verification.

- *Key generation.* After the preparation of outsourced data, the data owner first runs the following function to obtain the secret key sk and the public key pk for public verification

  $\mathsf{GenKey}(\lambda) \rightarrow (sk, pk)$:

  $sk := (x, \alpha) \leftarrow_{\$} \mathbb{Z}_p, pk := (v = g^x, \delta = g^{x\alpha}, \{g^{\alpha^j}\}_{j=0}^{s-1})$.

  Note that the size of pk grows linearly with the number of data blocks within a data chunk. For simplicity, we assume a generic group model is used and the group generator $g \in \mathbb{G}$.
- *Tag generation.* Using the secret key sk, the data owner is then required to run the following function to generate tags for each chunk of data based on the primitive of polynomial commitment

  $\mathsf{GenTag}(sk, m_{i,j}, name) \rightarrow \sigma_j, i \in [0, s], j \in [0, d]$:

  $\sigma_j := (H(name \| j) \cdot g^{f_{\overrightarrow{m_j}}(\alpha)})^x$.

  Extending the notation used in Section 4.2, we denote the polynomial with coefficients comprised of s data blocks $\{m_{i,j}\}_{i=0}^{s-1}$ as $f_{\overrightarrow{m_j}}$, where $j \in [0, d]$. As part of procedures to generate a tag σ_j, the data owner simply commits this polynomial by raising it to the group generator g. In addition, the data owner

TABLE 2
Notations Used in This Paper

Symbol	Description
F	The original data to be outsourced.
$\tilde{F}$	Data after pre-processing with auxiliary metadata attached.
s	Number of data blocks of a data chunk.
d	Number of data chunks in $\tilde{F}$.
$\mathbb{Z}_p$	A finite field with the order of a sufficiently large prime p.
$\mathbb{G}$	A generic group for bilinear pairing $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$.
$f_{\overrightarrow{A}}$	A polynomial with coefficients in $\mathbb{Z}_p$, forming the array $\overrightarrow{A}$.
g	A group generator of the group $\mathbb{G}$.
x, α	Two random group elements in the group $\mathbb{G}$, used as sk.
v, δ	$v = g^x$, $\delta = g^{x\alpha}$, together with $\{g^{\alpha^j}\}_{j=0}^{s-1}$, forming pk.
$name$	A randomly sample group element of $\mathbb{G}$.
σ	Authenticator of binded data chunk, group element of $\mathbb{G}$.
y	Polynomial evaluation result, element of $\mathbb{Z}_p$.
ψ	Witness created from polynomial remainder, element of $\mathbb{Z}_p$.
r	Point to be evaluated for the polynomial, element of $\mathbb{Z}_p$.
k	Number of challenged data chunks in each auditing round.

also needs to randomly sample a file identifier $name \in \mathbb{Z}_p$, binds it with the index j, and apply a random oracle $H(\cdot) : \{0,1\}^* \rightarrow \mathbb{G}$. Lastly, the data owner uses x in the private key sk to sign the above results such that it can be publicly verifiable using pk.

- *Tag verification.* After the generation of the tags σ_j, the data owner sends pk, σ_j, and the outsourced data $m_{i,j}$ to the storage provider. In order to check whether the data owner faithfully generates the tags using sk, the storage provider has to run the following function using the paired pk

$$\mathsf{VeriTag}(pk, m_{i,j}, name, \sigma_j) \rightarrow \mathsf{accept} \text{ or } \mathsf{reject} : e(\sigma_j, g) \stackrel{?}{=} e(H(nam\|j) \cdot g^{f_{\overrightarrow{m_j}}(\alpha)}, v).$$

Similar to the procedures of the GenTag function, the storage provider also needs to transform the data into the expression of polynomial coefficients and obtains $f_{\overrightarrow{m_j}}$ for each data chunk. Once the verification procedure is completed with a positive response, the data owner can confirm that pk is valid for the consequent auditing procedures and agree upon the storage contract. Otherwise, the storage provider needs to reject the storage contract unless the data owner transfers the new tags that can pass the VeriTag function within a pre-defined period.

Periodical Auditing Enforced by Contract. Upon the confirmation that pk is valid, pk is then posted to the contract for public verification. During the auditing period, a number of data chunks and their corresponding auxiliary data are randomly sampled to perform homomorphic operations for the consequent verification of equality check.

Following Fig. 3, we briefly explain the auditing procedures. We first assume fresh randomness is already available in the decentralized paradigm. In each auditing round, only two random seeds specified by the security parameter λ as well as another random element r sampled from $\mathbb{Z}_p$ are used for the challenging purpose. Through two PRF functions pre-specified, the storage provider can run the Chal function to derive a set $\mathcal{C}$, with which the data blocks and tags can be challenged together.

To perform the proving procedures via the Prove function, the storage provider first computes the aggregated tag σ for all challenged k indexes. Similarly, the storage provider can derive a new polynomial $f_{\vec{U_k}}(x)$, which is the linear combination of the original data representing polynomial $f_{\overrightarrow{m_j}}(x), j \in \mathcal{I}$. With the new polynomial, the storage provider generates a polynomial commitment y. Next, the storage provider needs to create a witness ψ to the polynomial commitment by first obtaining a quotient polynomial with the degree $s-2$ and raising the quotient polynomial to the power of the generator g. The final proof π to be posted to the blockchain consists of two group elements σ, ψ and a number element $y \in \mathbb{Z}_p$.

For the public verification, the smart contract automatically runs the Verify function. To be more specific, the smart

Illustration of interaction between the target storage provider and the smart contract for each auditing round

Smart Contract		**Storage Provider**
Challenge:		
$C_1 \leftarrow_{\$} \{0,1\}^\lambda, C_2 \leftarrow_{\$} \{0,1\}^\lambda, r \leftarrow_{\$} \{0,1\}^\lambda$	$\xrightarrow{C_1, C_2, \mathrm{r}}$	$\mathrm{PRF}_1(seed_1) \leftarrow \mathcal{I} \subset \{1, \cdots, d\}$, where $\lvert\mathcal{I}\rvert = k \leq d$
		$\mathrm{PRF}_2(seed_2) \leftarrow \{\nu_j\}_{j \in \mathcal{I}} \in \mathbb{Z}_p^k$
Verify:		
		$\sigma := \prod_{(j,\nu_j) \in Q} \sigma_j^{\nu_j}, f_{\vec{U_k}}(x) \leftarrow \sum_{(j,\nu_j) \in C} \nu_j \cdot f_{\overrightarrow{m_j}}(x)$
		$y := f_{\vec{U_k}}(r) \in \mathbb{Z}_p, f_q(x) \leftarrow (f_{\vec{U_k}}(x) - f_{\vec{U_k}}(r))/(x - r)$
$\phi := \prod_{(j,\nu_j) \in \mathcal{C}} H(name\|j)^{\nu_j}$	$\xleftarrow{\sigma, y, \psi}$	$\psi := g^{f_q(\alpha)} \in \mathbb{Z}_p$ with $\{g^{\alpha^j}\}_{j=0}^{s-2}$ in pk
Check the verification equation in Eqn. (1)		

Fig. 3. Flowchart of the auditing interaction during our enhanced auditing protocol.

contract needs to re-compute ϕ from the scratch. The final verification step combines the polynomial evaluation verification and the BLS signature verification, as displayed below:

$$e(\phi, v) \cdot e(\psi, \delta \cdot v^{-r}) \stackrel{?}{=} e(\sigma g) \cdot e(g^{-y}, v). \quad (1)$$

The above verification equation requires the smart contract to perform four times of bilinear pairing before deriving a final verdict.

Randomness Generation for Challenging. To generate unbiased distributed randomness, we take the standard approach of using a network of independently-run nodes, which satisfies the design requirement of decentralization. Alternatively, it is also feasible to adopt other approaches under the framework of randomness beacon.

5.2 Storage Saving Through Data Recovery

The storage auditing scheme in the decentralized paradigm essentially serves for the ecosystem built upon the service-level agreements and the underneath principle of fair exchange. Hence, participants in the decentralized auditing framework are full motivated to maximize their personal interests. Below we identify an overlooked data recovery problem, which any rational participant is expected to fully exploit. The problem is closely related to the storage outsourcing patterns that exist prevalently in the decentralized paradigm. What makes the data recovery problem stands out from the conventional storage outsourcing problem is that the blockchain platform introduced in our decentralized auditing framework can be used as a storage medium with zero marginal cost.

Applicability. Our identified problem is applicable to most similar decentralized storage ecosystem built on customized blockchain. Intuitively, on-chain accountability often indicates traceability. For Proof of Retrievability protocols modeled using data extractability, one could always save storage by recovering the original data from publicly accessible storage proofs. It is noteworthy, though, that Filecoin-alike ecosystems use a different proving framework where the soundness is modeled differently. Hence, this problem does not work on SNARK-based constructions.

Storage Outsourcing. Aside from the utilization of idle storage capacity locally, the storage provider can additionally outsource data storage and proof generation procedures to the integrated cloud service. Such storage outsourcing scenarios are thoroughly discussed in the design of Permacoin [51]. We emphasize that most data outsourcing options are intrinsically accompanied with the extra bandwidth cost and the existence of the data outsourcing strategy would also help lower down the overall storage cost in DSN. However, with the blockchain serving as a public accessible logging system that stores all audit trails, the storage providers are provided with an essentially free storage platform. Basically, all storage providers are incentivized to fully exploit the audit trails stored on the blockchain such that the audit trails can be transformed into original data blocks at the cost of computation.

Two Strategies for Data Recovery. We now describe the concrete strategies of freeriding audit trails stored on the blockchain from the perspective of storage providers. Later we will show the feasibility to use these two strategies from the perspective of a storage provider.

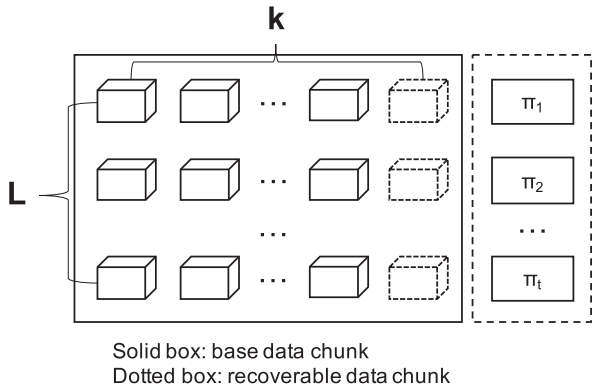


Fig. 4. Partial data recovery from auditing trails.

Strategy I. As a basic strategy, we first explain how to leverage the previous auditing trails stored on the blockchain, assuming at least one fresh challenged data chunk is issued in each auditing round. Fig. 4 shows two types of different data chunks, namely the base data chunks that the storage provider has to store faithfully and the recoverable data chunks that the storage providers can use previous audit trails to recover from. More specifically, in the process of leveraging previous audit trails $\{\pi_i\}$ to perform data recovery computation for the recoverable data chunks, we can use the polynomial interpolation to represent the same polynomial. Basically, we can view $(r, f_{\vec{U_k}}(r))$ used for a auditing proof as a data parity block. With a data block within the target data chunk (a polynomial coefficient group element) of the original data missing, the storage provider is still able to recover the entire target data chunk via polynomial interpolation. Compared to the case of faithfully storing the data block in the form of large group element, the storage provider only needs to take records of the small-sized metadata, e.g., the location of the recoverable data block, which would be used for the re-computation procedures of the recoverable data chunk.

Strategy II. The data recovery strategy described above is largely based on the assumption that the storage provider can locate at least one fresh challenged data chunk and targets the data chunk as the recoverable data chunk. In practice, as the auditing round proceeds, the probability that the challenged data chunks are all repeatedly challenged ones in the previous rounds increases constantly. For instance, if the challenged set of the second round is exactly the same as the first round, there would be no available new targets as recoverable data chunks. In face of this situation, the storage provider can take an alternative strategy of re-using recoverable data chunks. By re-using more pairs of data points $\{(r_i, f_{\vec{U_k}}(r_i))\}, (i \in [0, w-1])$ of the polynomial stored as π_i on the blockchain, where w is the total number of challenges targeting one data chunk, the storage provider can save the storage space without actually storing more data blocks. As an extreme example, after a sufficiently long time of storage auditing and with the target data chunk challenged s times, the storage provider can utilize the on-chain audit trails to re-compute the entire target recoverable data chunk. Nonetheless, we emphasize the probability of repeated challenges over one target data chunk shrinks remarkably, as the data size increases.

Number of Recoverable Data Chunks. We have identified the data recovery problem that theoretically could result in the

pass of verification procedures, even though the storage provider does not faithfully store the data as intended. The storage space can be saved for the storage provider using the above two strategies with a different amount of newly incurred storage overhead. To understand the actual storage saving capabilities, we need to carefully assess the amounts of storage overhead caused by indexing metadata with the two strategies in a quantitative manner. Before this quantitative analysis, we need to first assess the turning point of the strategy change.

We now elaborate on the computation of the total number of recoverable data chunks that the storage provider can take advantage of, which is a prominent indicator of the strategy shifting. The basic method of the estimation of the number of recoverable data chunks is to obtain a tight upper bound. As the auditing rounds proceed, the probability that the storage provider is incapable of finding a new target increases gradually. In the end, there would be a turning point when every chunk is challenged at least once. For convenience, we denote the number of auditing rounds at this point as L_{real}. In order to obtain a rough contour of the expected value of L_{real}, we first calculate an estimated number $L_{predict}$. Concretely, we apply the tail sum formula for expectation and derive $L_{predict} = \mathbb{E}\, X = \sum_{i=0}^{\infty} \mathbb{P}(X > i)$. Notice that the individual probability $\mathbb{P}(X > i)$, i.e., the first i rounds that do not cover anything, can be computed by aggregating the sub-probability that i batches will all miss a particular subset T. Specifically, we need to apply the *inclusion-exclusion principle* for the compensation of the over-estimated aggregated probability. With the observation that the sub-probability only depends on the size of subset T, we further denote the size of subset as t. Since $\binom{d}{t}$ sets contribute to sub-probability of the t-element subsets, we have the following formulation:

$$\mathbb{P}(X > i) = \sum_{t=1}^{d} \binom{d}{t} \cdot (-1)^{t+1} \binom{d-t}{k}^{i} / \binom{d}{k}^{i}.$$

Based on the derived $\mathbb{P}(X > i)$, we can further compute:

$$L_{predict} = \sum_{i=0}^{\infty} \sum_{t=1}^{d} \binom{d}{t} \cdot (-1)^{t+1} \binom{d-t}{k}^{i} / \binom{d}{k}^{i}.$$

We observe that the above infinite series converges absolutely and we can obtain the simplified expression as follows:

$$L_{predict} = \binom{d}{k} \sum_{t=1}^{d} \frac{(-1)^{t+1} \binom{d}{t}}{\binom{d}{k} - \binom{d-t}{k}}.$$

Because the number of challenged data chunks k is mostly fixed and the function regarding $L_{predict}$ is monotonically increasing, the expected number of auditing round for the turning point increases with the stored data size. Aside from the above closed-form expression for $L_{predict}$, we will also demonstrate the simulation results of L_{real} through the experiments.

Storage Saving Capabilities. In theory, both strategies described above are supposed to save the same amount of storage capacities, namely a single data block per auditing round. However, it is also indispensable to introduce extra storage overhead for the use of location indexing during the implementation of the strategies. Briefly, the metadata required for location indexing is comprised of a *key:value* pair, where the *key* identifies the data chunk number and the *value* stores the location information of the on-chain parity data block used for data recovery. As a direct consequence, the necessary storage space used for each *key:value* pair in the second strategy grows gradually with the value of w.

Applicability. As we assume we work on the generic blockchain paradigm, our discovered storage freeriding problem naturally prevail in other customized blockchain systems. The underlying reason is that most storage auditing designs are based on analysis of single-round proof of retrievability constructions. While it is natural to extend to multi-round scenarios in the conventional settings, the cost-free blockchain paradigm amplifies the bypassing possibility of the auditing framework through the outsourcing strategy. For SNARK-equipped constructions like Filecoin, the outsourcing strategy is also feasible, unless the costly setup procedures are faithfully completed.

5.3 Mitigation Strategy

The intrinsic essence of our identified problem above is that the blockchain serves as a publicly accessible platform (distributed nodes) with zero storage cost. To prevent a significant amount of the audit trails (mostly related to polynomial evaluation) from being abused by the incentive-driven storage providers, the underlying solution is to balance the public verifiability property and the public accessibility property of the blockchain.

As a once-for-all approach, we can utilize redactable blockchain platforms [52], where authorized parties are allowed to modify the stored audit trails. The desirable property of redactable blockchain platforms is that the size of the blockchain remains within a stable interval due to the redacted blocks after a pre-determined amount of time. Hence, the storage providers are deprived of the capability of freeriding the audit trails stored on the blockchain. In practice, though, commonly used blockchain platforms do not support the redaction of blocks. To this end, we need to use other methods to eradicate the data recovery concern caused by storage freeriding with zero marginal cost.

ZKP as a Solution. Blockchain-enabled applications have driven the development of zero knowledge proof (ZKP) protocols to be deployed in practice for the mitigation of on-chain information leakage. As discussed earlier, compared to generic ZK-SNARK frameworks would produce an unbearable amount of both computational and storage overhead for the storage auditing application. Hence, lightweight and customized zero knowledge schemes that are friendly to algebraic operations are more suitable for the auditing framework compatible with DSN.

Concretely, the random masking technique [4] can be employed to prevent other offline adversaries from inferring the original data with the masked audit trails stored on the blockchain. In other words, instead of using (σ, y, ψ) as a storage proof, the masked storage proof is supposed to avert data extraction guaranteed by definition. Specifically, a hiding parameter $z \leftarrow_{\$} \mathbb{Z}_p$ is required to be generated on the side of the storage provider. In addition, with the random oracle $H'(\cdot) : \mathbb{G}_T \rightarrow \mathbb{Z}_p$, the storage provider can input the

$\mathbb{G}_T$ element $R = e(g, v)^z$ and derives the output $\zeta = H'(R)$ as a new hiding parameter. Compared to the group element y that is directly exposed, the new group element y' in the ZKP version of the auditing scheme as displayed below:

$$y' = \zeta \cdot f_{\vec{U_k}}(r) + z.$$

The storage proof generated from $\mathcal{S}$ would be (R, σ, y', ψ), whose size is twice more than the original storage proof due to the enormous size of the $\mathbb{G}_T$ element. Correspondingly, the verification algorithm performed by the smart contract alters as follows:

$$e(\boldsymbol{\phi}^{\zeta}, v) \cdot e(\boldsymbol{\psi}^{\zeta}, \delta \cdot v^{-r}) \stackrel{?}{=} \cdot e(\sigma^{\zeta}, g) \cdot e(g^{-y'}, v). \quad (2)$$

Enhanced Storage Auditing With ZKP. Though the intuitive solution of applying customized ZKP protocols to the auditing framework is effective to address the on-chain data extraction problem, it is not sufficient to resolve the storage freeriding problem. The reason is that the ZKP auditing scheme still relies on the masking input provided by the storage provider. Unless the storage provider follows the proof generation procedures faithfully in the sense that the mask input is randomly generated for each of the auditing rounds, the storage provider can abuse the audit trails stored on the blockchain. Given this honest-prover ZKP auditing scheme, the more rational strategy for the storage provider is to reuse the random mask inputs every time and sticks to the data recovery strategy from linear equations we demonstrated. But in practice, we can still validate whether the mask inputs are unique in each auditing round through the smart contract logic, because the $\mathbb{G}_T$ element R is recorded on the blockchain.

In Fig. 5, we display the concrete on-chain procedures of the validation of the mask inputs. The validation of uniqueness of mask inputs is performed by the function Query. In order to constrain the on-chain computational cost caused by the function Query per audit, we set an upper limit num for the maximal pairs of past mask inputs to be compared by the smart contract. After the counter cnt exceeds this upper limit, it is compulsory for the data owner to repeat the data preprocessing procedures so that the generated data tags and the consequent audit trails to be generated are "renewed". With the indispensable renewing procedures, it is of vital importance to balance the overhead of periodical data preprocessing procedures and the maximal cost of on-chain validation. To this end, we will present more insightful discussions with the support of evaluation results in Section 7.3.

6 Security Analysis

Our entire design of the decentralized auditing framework is built upon the correctness of the auditing protocol. In this section, we briefly evaluate the correctness of our proposed auditing design with regard to the completeness and soundness of the interactive proof system. It uses a conventional security framework and we use the same approach to assess the security. Aside from the correctness of the auditing design, we also analyze our proposed countermeasures in the economic paradigm of rational actors.

Smart contract verification with mask input validation :

```
mapping (address => address) list;
address[] public addresslist;
Function Size() public returns address:
  return addresslist.length;
Function Query() public returns bool:
  for (i = 0; i < K.Size(); i++) // K is the constructed list
    if (K.addresslist(i) == R):
      return True;
  return False;
On receive ("prove", prf) from S :
  assert st = AUDIT;
  broadcast "proof posted"; call scheduling ("verify");
On trigger scheduling ("verify") :
  if cnt ≥ num : ⊥;   // max rounds of auditing allowed
  if Query(R) = TRUE : ⊥;   // through a mapping list
  if Verify(σ, y', ψ, R) = TRUE :
    broadcast "pass";
    unlock and transact fee to S;
  else:
    broadcast "fail";
    unlock and transact fee to D;
  cnt = cnt + 1;
  call scheduling("challenge");
  // issue the next auditing round of challenge
```

Fig. 5. Enhanced on-chain verification procedures against data recovery with zero marginal cost.

6.1 Integrity of Storage Auditing

*Completeness.*The first part of the security analysis is to prove that the verifier will always approve it, if the proof is correctly computed. By simplifying the Equation (2), we obtains the following:

$$\begin{aligned} RHS &= R \cdot e((\prod_j t_j)^{x\zeta} \cdot g^{x\zeta \cdot P_k(\alpha)}, g) \cdot e(g^{-(\zeta \cdot P_k(r)+z)}, g^x) \\ &= R \cdot e(g, g)^{x(\zeta \cdot (P_k(\alpha) - P_k(r)) - z)} \cdot e(\prod_j t_j, g)^{x\zeta} \\ &= e(\prod_j t_j, g)^{x\zeta} \cdot e(g, g)^{x\zeta\psi(\alpha - r)} \\ &= LHS, \text{where } t_j = H(name \| j)^{\nu_j} ((j, \nu_j) \in \mathcal{C}). \end{aligned}$$

Soundness and Extractability. The second, and more important security guarantee is to ensure the storage provider is indeed storing the data specified by the storage contract. As a customized auditing scheme, we inherit the formal security model used by [14]. The security proof is divided into two parts: 1) To prove that the verification algorithm will always reject it, if the proof is not correctly computed using the linear combination of data blocks $m_{i,j}$. 2) To prove that there exists an extraction algorithm, which can efficiently reconstruct a fraction of the data with the generated storage proofs. The first part of proof is based on CDH and q-BSDH assumption [49] and uses standard cryptographic techniques, while the second part of proof uses combinatorial techniques. Since the security proofs are standard, we refer the readers to the concrete analysis in [5], [14], for more

details of security proofs. Due to the space limit, we only present the brief proof sketch for the following theorem.

Theorem 1. *A valid storage proof can only be generated when a prover faithfully stores fraction of the data, given the discrete log and q-BSDH assumptions. After a sufficient number of auditing rounds, the storage proofs generated along ensures that even if ξ-fraction of the data is tampered, it can be detected with probability ϵ, where ξ and $1-\epsilon$ are admissible probabilities.*

Proof Sketch. This theorem is proved with similar approaches displayed in the work [14], [18], [53]. The proof consists of three parts, namely the unforgeability, extractability, and retrievability. The first part about unforgeability is to prove that no probabilistic polynomial time adversary can compute a valid but not correct proof except for negligible probability. Following the standard simulated PoR games, this is eventually reduced to the statement that the computationally bounded adversary cannot find the secret keys unless it solves discrete log and q-BSDH assumptions. The second part about extractability is to prove the μ within the proof is correctly computed in the way that μ represents a linear combination of data blocks m_{ij}.

We emphasize the following implication that draws from the above analysis: The existence of an efficient extraction algorithm used in the second part of proof does not eliminate the viable and profitable strategy of data outsourcing. This part exactly uses the combinatorial techniques introduced in the early PoR work [14].

Retrievability. Lastly, the third part regarding retrievability is to prove given a fraction of encoded data, it is always possible to recover the original data with an overwhelming probability. This part of proof can easily use the conventional coding techniques. Also note this part of proof is closely related to our identified security problems, which abuse this data recovery techniques.

Zero Knowledge. Our auditing protocol also can prevent other off-chain parties from learning useful information ofthe auditing proofs stored on the blockchain. Following the definition used in our previous work [6], the auditing protocol can be seen as a witness-indistinguishable Sigma protocol for the relation

$$\mathcal{R} = \left\{ (pk, \{t_i\}, chal, \sigma), F \mid \sigma^{pk} \equiv g_1^F \cdot \prod_i t_i \bmod N \right\},$$

where $\tilde{F}$ is the witness in the above relation serving as private inputs. Easily we can derive the following theorem that states any off-chain adversaries cannot learn information other than the statement itself.

Theorem 2. *For any probabilistic polynomial time adversary $\mathcal{A}_{PPT}$, the probability $\mathcal{A}_{PPT}$ can distinguish witness (w_1, w_2) after seeing all public inputs is negligible. Moreover, under the standard Discrete Logarithm assumption, the probability $\mathcal{A}_{PPT}$ can distinguish F from random data after seeing all public inputs are negligible.*

Concrete Parameterization. To achieve the standard 128-bit security strength, it is reasonable to select a 256-bit elliptic curve group for our public-key based auditing framework. In addition, it is also essential to specify the number of challenged data chunks k for the purpose of instantiating the concrete data recovery probability. Given that numerous studies have analyzed the possible range of k [8], we set k to 300, which equivalently guarantees the the data owner storage assurance of 95 percent, even if only 1 percent of entire data is corrupted. In the paradigm of decentralized archive storage, multiple auditing rounds makes it sufficient for feasible and effective storage auditing.

Pre-Auditing Fairness. In our threat model, the data owner and the storage provider are described as rational players. For the main auditing procedures, the fairness is enforced by the publicly verifiable smart contract. However, during the pre-auditing phase, the storage provider is always capable of launch a denial-of-service attack by refusing to provide storage service after initial negotiation, as long as the smart contract is not effective. We stress this a reputation checking system would be effective to alleviate this attack.

6.2 Arbitrage in the Paradigm of Rational Actors

The property of zero knowledge only proves that any other off-chain adversaries cannot learn useful information of the original data. Hoewever, it does not regulate the prover, i.e., storage providers. In this section, our adversarial model mainly focuses on the paradigm of rational actors, as have been indicated in Section 3.3. The outsourcing strategy, especially in the scenario of computation outsourcing, is widely recognized and well analyzed [51], [54] in the decentralized ecosystem. To this end, we adopt the definition of *weak non-outsourceability* illustrated in [54] and apply it to our usage scenario of storage outsourcing instead of computation outsourcing. By *weak non-outsourceability*, we mean that any rational storage providers will prefer *local storage* to drive down the overall resource costs.

Assumption. To effectively measure the resource costs, we assume the storage capacity is more valuable than other resources, for its profitability in the cases that a storage provider to accept extra storage contracts with data small volumes. Intuitively, in the blockchain paradigm, the outsourced storage cost and other resource costs (e.g., bandwidth cost and computational cost) through the public blockchain platform are negligible. Hence, to achieve weak *non-outsourceability* in the rational settings, the local storage cost in the outsourced storage scenario should be larger enough to match with that of the local storage only scenario.

Our Countermeasure. From the perspective of storage providers (provers), the local storage overhead in the outsourced storage scenario is controllable by further outsourcing and proper deduplication. Concretely, the mask inputs provided by provers to achieve witness indistinguishability, which is also outsourceable (the mask input R is directly recorded on the blockchain) and deduplicable if the mask inputs are deliberately picked by provers. To this end, we adopt a heuristic approach to add additional checking procedures to ensure the mask inputs are unique and non-deduplicable. The effectiveness of this approach is demonstrated with evaluation results in Section 7.3.

Remarks. In its essence, we have provided a heuristic countermeasure against storage providers' potential outsourcing tendency. By enforcing an additional check with the smart contract, our work provides a lightweight and

practical approach to discouraging profitable storage outsourcing, which is especially useful for long-term frequent auditing.

7 Evaluation

To understand the performance overhead of our decentralized auditing framework, we conduct a series of experiments to evaluate our prototype based on the existing decentralized storage infrastructure. The prioritized goal of the experiments is to assess the on-chain overhead of our decentralized auditing scheme, which occupies a dominant share of the total auditing cost. In the meantime, we also study the feasibility of our new attack and the on-chain cost of using our mitigation solution. Besides on-chain cost, we also demonstrate the operation-wise off-chain cost in details and assess the overarching auditing cost in general.

7.1 Implementation and Evaluation Overview

We implement a prototype of our proposed decentralized storage auditing framework. To show compatibility support, we use the testbed environment based on Tahoe-LAFS [56], which provides all core components (e.g., distributed routing mechanism) other than the storage auditing framework for a complete decentralized storage solution. Additionally, we use Ethereum as our blockchain-based arbitration platform for its influence among decentralized applications. But we emphasize our auditing design is not limited to this particular testbed and can be ported to other major platforms without restraint.

We have managed to integrate our decentralized auditing framework with the underlying storage platform with ease. To successfully port the on-chain auditing procedures, however, we need to surmount numerous obstacles, mostly due to compatibility issues of implementation. The primary hindrance we encounter on the Ethereum platform is the lack of support for advanced cryptographic operations, though the Ethereum architecture is proven to be quasi-Turing complete in theory. Specifically, most standard cryptographic libraries that support bilinear pairing operations cannot be directly ported to the Ethereum platform without sacrificing performance. Consequently, the straightforward testing method using Ethereum Testnet is also not applicable in our experiments.

To this end, we pre-compile the core components for elliptic curve computation and generate gas-efficient opcodes accordingly. With the design goal of maximizing on-chain efficiency, our implementation is comprised of over 1,500 lines of native C/C++ codes. Our implementation depend on the emerging mcl[1] cryptographic library and in particular, we choose the BN254 curve for the balance of security and efficiency. In order to simulate the Testnet environment with our own opcodes, we deploy a private blockchain network using the Ethereum template. The private Testnet consists of three independent miners, one dedicated storage provider and one node possessed by the data owner. The miners for proof verification and the storage provider for proof generation both use workstations as in reality (Intel Xeon E-2174G CPU x4 @ 3.80 GHz) in Linux (Ubuntu Server 18.04 LTS). While on the other hand, a Desktop PC (Intel Core i7 8700k CPU x6 @ 3.70 GHz) is equipped on the side of the data owner for the data preprocessing procedures.

Before we demonstrate the evaluation results with details, we first conduct a comparison with the state-of-the-art works, where similar concepts of decentralized storage auditing are adopted. The first notable work [27] is the construction based on the emerging primitive of accumulators. As this construction is not feasible for large-size data yet, we set the file size to 1 Mbit. We also use the design of ZKCSP [55] as our baseline benchmark for the data size of 1 Mbit, which is based on generic zksnark frameworks [38] and only suitable for scriptless Bitcoin. Aside from the above academic works, we also compare with the constantly evolving Filecoin project (as of Feb 2021), whose latest Lotus implementation is worth comparing for medium-sized data. The results are displayed in Table 3.

Note that the verification cost is estimated with the gas cost unified using the standard Ethereum baseline. We can see that from a holistic view, our main storage auditing solution is quite lightweight and efficient, compared with the above mentioned works. Below we demonstrate the detailed cost of one-time setup cost, the most prominent on-chain procedures,and the user conceivable off-chain procedures.

7.2 Overhead Upon Pre-Contract Stage

The initial step of the entire auditing framework is to generate public keys that the smart contract leverages for verification procedures during the periodical auditing stage. In order to successfully outsource data to the target storage provider, it also takes a non-negligible amount of time to complete the data preprocessing procedures. Below we demonstrate the concrete efficiency in the stage of public key generation and data preprocessing. First we emphasize that there exists endogenous conflicts between the public key size and the data preprocessing performance.

As displayed in Fig. 6, the increase of the public key size to be posted on the blockchain leads to the larger data chunk size to be employed in the auditing procedures, which indicates the ability to "batch" operations. However, the linear growth of the public key size also means multiplied on-chain storage cost. In particular, when the data chunk size is set to 256 kB, the on-chain storage cost also maintains at the same level (over 250 kB). In current deployment status of Ethereum, it would cost at least 200 blocks with full gas cost limit to complete the one-shot public key on-chain submission procedures.

With such overwhelming on-chain overhead, it is more desirable to specify a smaller data chunk size. However, as the size of the data chunk declines, the data preprocessing time grows substantially. Fig. 7 demonstrates the data preprocessing time measured in seconds. Particularly, for 1 GB data to be stored in the DSN, it takes over 40 seconds to complete the data preprocessing procedures for the medium-sized data chunk (32 kB). Hence, the data regarding the on-chain public key size and the data preprocessing time reveals the irreconcilable differences and urges a more careful selection upon the use of data chunk size in practice. In the following descriptions, we uniformly employ a medium-sized data chunk size

1. https://github.com/herumi/mcl

TABLE 3
Comparison With Existing Decentralized Storage Auditing Solutions

	File Info.	Pre-process		Proof Generation			Verification	
	Size	Time	Public Param. size	Time	Memory	Size	Time	Cost est.
VDS [27]	1 Mbit	~ 10 min	~ 2 KB	424 s	~ 300 MB	256 bytes	309 ms	$4.58
ZKCSP [55]	1 Mbit	14 s	40 MB	3 s	100 MB	374 bytes	37 ms	$0.62
Filecoin* [13], [29]	1 GB	~ 870 s	~ 2 GB	10.5 s	1 GB	384 bytes	24 ms	$0.44
Our main solution	1 GB	118 s	~ 5 KB	46 ms	3 MB	**288 bytes**	**7 ms**	$0.21

∗ *Our evaluation results are based on the April 2021 implementation of Lotus. It also uses all four CPUs of our testbed.*

32 kB, i.e., each data chunk is comprised of over 1,000 data blocks, unless specified otherwise.

7.3 Performance of On-Chain Activities

Feasibility of Storage Freeriding. In our main design, we spend enormous efforts on identifying the storage freeriding problem, a loophole that essentially depends on the past audit trails stored on the blockchain. More concretely, the exploitation of the past blockchain storage involves two separate strategies that could be possibly employed by the economically incentivized adversaries. Therefore, we simulate the auditing procedures for different sizes of data storage and attempt to observe the strategy shifting in these various scenarios. In Fig. 8, we demonstrate the simulated results, with the predicted upper boundary attached as well. As we can see in the graph, for a relatively small data volume, e.g., 32 MB, it takes merely approximately 10 auditing rounds for the adversary to shift strategies and maximize their profits. While for larger volumes of data, the required number of auditing rounds grows significantly to several hundreds, making the basic strategy (strategy I) the primary one and the strategy II the complementary one throughout the storage contract.

With the above understanding of the involvement of two strategies, we now proceed to measure the extra storage overhead due to the indexing metadata during the usage of both strategies. As mentioned in Section 5.2, each parity data block stored on the blockchain needs to be located through *key:value* indexing metadata. In the experiments, we allocate 2 bytes for the regular indexing of positions of on-chain audit trails once the data size is constrained such that the position of each data chunk can be accurately described with this indexing information.

To further save storage space on this basis, we observe it is possible to employ the position delta with regard to the target data chunk when applying the strategy I. However, the usage of position favors small volumes of data, since the strategy I would count as the primary strategy for data of small sizes. Under this circumstance, Fig. 9 displays the newly incurred storage overhead with details in the adversarial usage of the past audit trails stored on the blockchain. The quick observation is that for any adversary who intends to accomplish storage freeriding, the overall extra storage overhead incurred due to the use of different strategies embodies little differences for various data sizes. Hence, storage providers who tends to maximize their profits would almost always employ the storage freeriding strategy as much as possible and it urges for our mitigation solution based on the ZKP-adapted storage auditing scheme.

Concrete On-Chain Cost. With the numerical results regarding on-chain freeriding exploitation, we now proceed to analyze primary components of the on-chain cost undertaken by ordinary users. For the baseline auditing protocol that involves pure proof evaluation verification operations, the number of bilinear pairing operations play a dominant role in the time consumption for the smart contract. While for the ZKP-adapted solution that requires extra boundary check, the operations listed in Fig. 5 also account for a decisive share. To compare such differences, we prepare Fig. 10 for the detailed demonstration of on-chain gas cost, considering the involvement of ZKP adaption. Because the bilinear operation is not natively supported by Ethereum, we estimate the expected

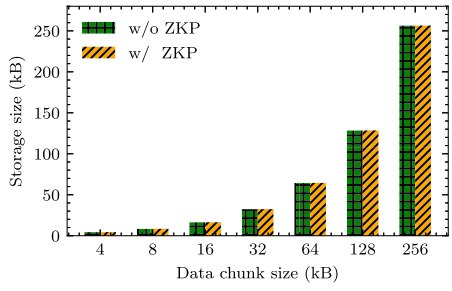


Fig. 6. Public key size for one-shot on-chain submission per user.

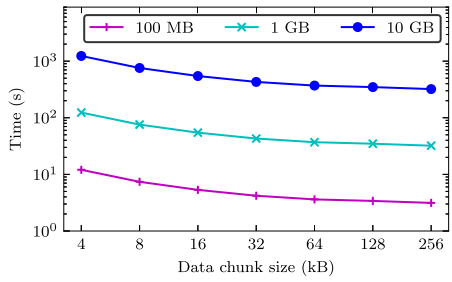


Fig. 7. Data preprocessing time before a storage contract takes effect.

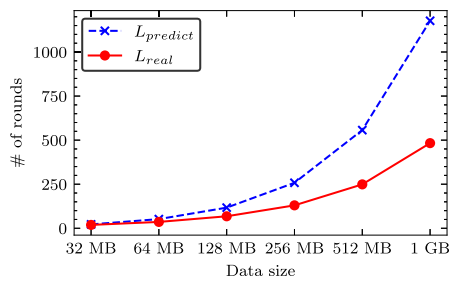


Fig. 8. Shifting point of the storage freeriding strategies.

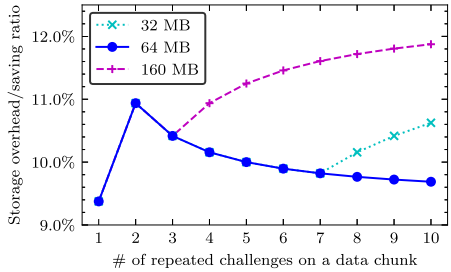


Fig. 9. Proportion of extra storage overhead to storage saved, for different levels of data sizes.

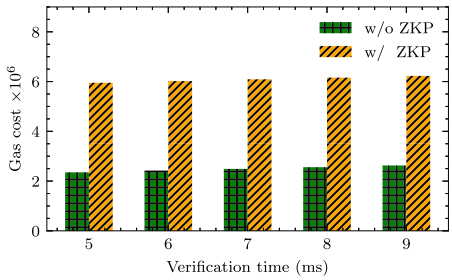


Fig. 10. Expected gas cost per auditing round with/without the use ZKP data hiding techniques.

gas cost by interpolating various amounts of verification time. To determine the range of verification time, we conduct experiments on multiple sets of mainstream devices for block mining and consequently the range is set 5 milliseconds to 9 milliseconds.

In addition to the basic ZKP comparison, we also evaluate the on-chain cost according to different stages within a life-cycle of the smart contract. According to an intuitive principle that cost amortization could favor larger number of auditing rounds, we prepare Fig. 11 to quantitatively display the relationship between the gas cost of the contract deployment stage as well as the actual auditing stage, and the number of auditing rounds. All experiments are conducted on the basis of Equation (2), where ZKP-adapted verification procedures are provided to mitigate the storage freeriding attack.

Finally, we present illustrative results to show the estimated explicit overall storage auditing cost using our proposed design, which is controlled by on-chain cost for public verification. In Fig. 12, we primarily consider the total amount of fees that are expected to be paid by data owners, which is measured in USD dollars using the latest conversion rate in late 2020. Similarly to the centralized cloud storage, the storage duration specified by the storage

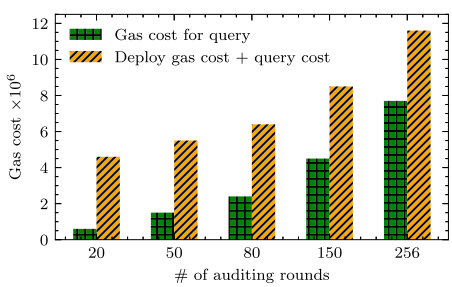


Fig. 11. Amortized gas cost per auditing round with extra on-chain query procedures using ZKP.

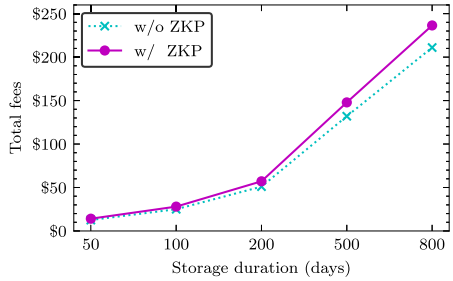


Fig. 12. Estimated overall explicit storage auditing cost in a deployable decentralized storage environment.

contract is measured in days. We obtain this figure using the combined results in Figs. 10 and 11. Though more expensive than the cloud storage solution for its implicit storage auditing cost, the total fees are of the same magnitude as that of cloud storage solution, which is bearable for users with strong security requirements. Notably, compared to our baseline solution without any extra protection against storage freeriding, our ZKP-adapted solution only incur a 10 to 15 percent increase in overall explicit storage auditing cost. The concrete numbers are also displayed in Fig. 12.

7.4 Overhead of Off-Chain Operations

Though the on-chain performance plays a leading role in affecting the overall auditing cost, the off-chain operations should also not be overlooked for its direct impact on the usability of the auditing framework. We now further perform the usability study from the perspective of storage providers, which is a crucial factor in determining the sustainability and scalability of the entire decentralized storage system. Using the evaluation results from Figs. 13 and 14, we can observe that similar to the one-shot on-chain storage size, the proof generation time on the side of storage providers grows linearly with the increase of the chunk size. As usual, it is preferable to set a medium-sized data chunk size, e.g., 32 kB, for the balance of data preprocessing time and the auditing overhead. Specifically, Fig. 13 compares the proof generation time under the influence of two different standards [8] of tampered data detection. The most commonly used standard is the ability to detect with 95 percent probability, when 1 percent data is tampered. Under this standard, a set of 300 random data chunks need to be challenged in each auditing round. The higher standard, on the other hand, requires 460 data chunks to be included in a proof to guarantee detection confidence of 99 percent probability. The proof generation time difference

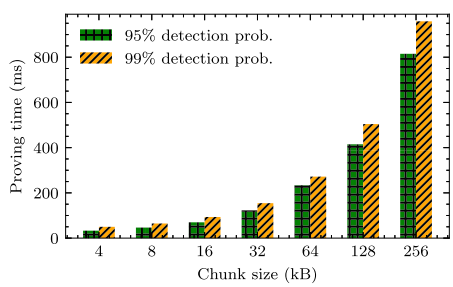


Fig. 13. Storage proof generation time under the influence of different storage guarantee criterion.

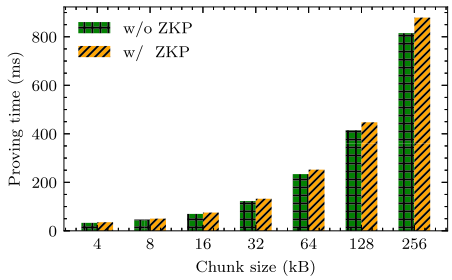


Fig. 14. Storage proof generation time comparison, with/without the ZKP adapted solution involved.

for two standards is constrained under 20 percent according to the results displayed in Fig. 13. Aside from the factors of detection ability, we also prepare Fig. 14 to see how the adoption of the ZKP auditing scheme affects the proof generation time. The observation is that the time difference is much smaller than the effect of a higher standard of detection ability. More concretely, for a 32 KB data chunk size that is widely usable, we can well constrain the proof generation time within 150 ms for one single storage proof.

8 Discussion and Future Work

Dynamism Support. Our currently proposed auditing design intends to provide a basic deployable decentralized auditing framework for static data, mostly for the use of data archiving systems. The support of storage auditing for more commonly used dynamic data is a delineate and prioritized objective that matters directly to the long-term support of DSN. Though a series of dynamic storage auditing schemes [22], [23], [57] have been proposed for centralized cloud storage settings, all the designs in general cannot be directly applied to on-chain auditing for the consideration of on-chain overhead. In order to achieve dynamic storage auditing for decentralized storage, we plan to first provide a revised definition that suits decentralized auditing for dynamic data and then refine the indexing schemes used for the data locating purpose in the dynamic settings. We will further consider the backward compatibility of the dynamic storage auditing scheme in future works, such that most design insights as well the prototype implementation experience of this work can be utilized.

Usability Support. In spite of the security guarantees and efficiency improvements of the storage auditing scheme provided in this work, the usability support from the perspective of data owners and storage providers is still an important aspect we have not covered. With a much improved storage auditing design in terms of usability, the entry barrier for interested participants could be significantly lowered. In future studies, we will focus on reducing the data preprocessing overhead for data owners and strengthening the scalability support of the proof generation procedures on the side of storage providers.

Expandability for PoST. With stronger security guarantees, Proof of Storage Time protocols enables the verifiers to efficiently audit the data storage in the manner that the outsourced data is continuously retrievable, thanks to the recent advancement of the primitive of verifiable delay function [58], [59], [60], [61]. In theory, any Proof of Storage scheme can be customized to construct PoSt. For future works, we also plan to study the feasibility of PoSt protocols in the decentralized settings. Concretely, we will look into three specific problems, following our current decentralized storage auditing framework. As an initial effort, we will study how to construct a sufficiently efficient PoST-based storage auditing protocol without incurring much overhead on the side of provers. Another notable problem is the extendability to the three-party settings, specified in our formalized decentralized storage auditing framework, as the primitive of PoST is designed to be a two-party protocol. Lastly, it is also worth studying a PoST-compatible auditing scheme for dynamic data by formalizing a periodical PoST-based storage auditing framework. Under this new framework, whenever the data is updated, a PoST proof is required for storage auditing purposes.

9 Conclusion

In this study, we analyze the current practices of the storage auditing frameworks used by decentralized storage solutions and present a generic storage auditing design, which suits our revised security framework. In the meantime, we also strive to derive a more efficient storage auditing scheme, making it more affordable for users to join in the ecosystem. We hope the this study offers a harbinger of feasible storage auditing designs in the decentralized settings and lays the solid ground for the continuous development and wider range of DSN employment boosted by the blockchain technology in the future.

Acknowledgments

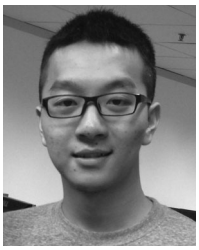
This work was supported in part by the National Key R&D Program of China under Grant 2020YFB1005500, in part by the Research Grants Council of Hong Kong under Grants CityU 11217819 and CityU 11217620, in part by the Innovation and Technology Commission of Hong Kong through ITF Project under Grant ITS/145/19, and in part by the National Natural Science Foundation of China under Grants 61572412, U20B2049, and 61822207.

References

[1] B. Cohen, "Incentives build robustness in bittorrent," 2003. [Online]. Avaiable: http://bittorrent.org/bittorrentecon.pdf
[2] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proc. IEEE/ACM IWQoS*, 2009, pp. 1–9.
[3] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling public verifiability and data dynamics for storage security in cloud computing," in *Proc. Eur. Symp. Res. Comput. Secur.*, 2009, pp. 355–370.
[4] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, 2010, pp. 1–9.
[5] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability and constant communication cost in cloud," in *Proc. Int. Workshop Secur. Cloud Comput.*, 2013, pp. 19–26.
[6] Y. Du, H. Duan, A. Zhou, C. Wang, M. H. Au, and Q. Wang, "Towards privacy-assured and lightweight on-chain auditing of decentralized storage," in *Proc. Int. Conf. Distrib. Comput. Syst.*, 2020, pp. 201–211.
[7] Y. Deswarte, J. Quisquater, and A. Saïdane, "Remote integrity checking — How to trust files stored on untrusted servers," in *Proc. Work. Conf. Integrity Intern. Control Inf. Syst.*, 2003, pp. 1–11.
[8] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.
[9] T. J. E. Schwarz and E. L. Miller, "Store, forget, and check: Using algebraic signatures to check remotely administered storage," in *Proc. Int. Conf. Distrib. Comput. Syst.*, 2006, pp. 12–12.

[10] G. Ateniese *et al.*, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 12:1–12:34, 2011.
[11] P. Golle, S. Jarecki, and I. Mironov, "Cryptographic primitives enforcing communication and storage complexity," in *Proc. Int. Conf. Financial Cryptogr.*, 2002, pp. 120–135.
[12] L. C. David Vorick, "Sia: Simple decentralized storage," 2014. [Online]. Available: https://sia.tech/sia.pdf
[13] P. Labs, "Filecoin: A decentralized storage network," 2017. [Online]. Available: https://filecoin.io/filecoin.pdf
[14] H. Shacham and B. Waters, "Compact proofs of retrievability," *J. Cryptol.*, vol. 26, no. 3, pp. 442–483, 2013.
[15] N. Kaaniche, E. E. Moustaine, and M. Laurent, "A novel zero-knowledge scheme for proof of data possession in cloud storage applications," in *Proc. IEEE/ACM CCGrid*, 2014, pp. 522–531.
[16] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. Int. Conf. Secur. Privacy Commun. Netow.*, 2008, pp. 1–10.
[17] J. Xu and E. Chang, "Towards efficient provable data possession," *IACR Cryptol. ePrint Arch.*, vol. 2011, 2011, Art. no. 574.
[18] J. Xu and E.-C. Chang, "Towards efficient proofs of retrievability," in *Proc. Int. Workshop Secur. Cloud Comput.*, 2012, pp. 79–80.
[19] A. Juels and B. S. K. Jr., "PORs: proofs of retrievability for large files," in *Proc. ACM Conf. Comput. Commun. Secur.*, P. Ning, S. D. C. di Vimercati, and P. F. Syverson, Eds., 2007, pp. 584–597.
[20] K. D. Bowers, A. Juels, and A. Oprea, "Proofs of retrievability: Theory and implementation," in *Proc. ACM Workshop Cloud Comput. Secur.*, 2009, pp. 43–54.
[21] M. B. Paterson, D. R. Stinson, and J. Upadhyay, "Multi-prover proof of retrievability," *J. Math. Cryptol.*, vol. 12, no. 4, pp. 203–220, 2018.
[22] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.
[23] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2013, pp. 325–336.
[24] R. Curtmola, O. Khan, R. Burns, and G. Ateniese, "MR-PDP: Multiple-replica provable data possession," in *Proc. Int. Conf. Distrib. Comput. Syst.*, 2008, pp. 411–420.
[25] K. D. Bowers, A. Juels, and A. Oprea, "Hail: A high-availability and integrity layer for cloud storage," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2009, pp. 187–198.
[26] D. Francati *et al.*, "Audita: A blockchain-based auditing framework for off-chain storage," 2019, *arXiv:1911.08515*.
[27] M. Campanelli, D. Fiore, N. Greco, D. Kolonelos, and L. Nizzardo, "Incrementally aggregatable vector commitments and applications to verifiable decentralized storage," in *Proc. ASIACRYPT*, 2020, pp. 3–35.
[28] G. Ateniese, L. Chen, M. Etemad, and Q. Tang, "Proof of storage-time: Efficiently checking continuous data availability," in *Proc. NDSS*, 2020.
[29] Filecoin, "Filecoin proving subsystem," 2020. [Online]. Available: https://github.com/filecoin-project/rust-fil-proofs
[30] B. Fisch, "Tight proofs of space and replication," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2019, pp. 324–348.
[31] I. Damgård, C. Ganesh, and C. Orlandi, "Proofs of replicated storage without timing assumptions," in *Proc. Annu. Int. Cryptol. Conf.*, 2019, pp. 355–380.
[32] T. Moran and D. Wichs, "Incompressible encodings," in *Proc. Annu. Int. Cryptol. Conf.*, 2020, pp. 494–523.
[33] R. Garg, G. Lu, and B. Waters, "New techniques in replica encodings with client setup," in *Proc. Theory Cryptogr. Conf.*, 2020, pp. 550–583.
[34] "Storjstat," 2019. [Online]. Available: https://storjstat.com/
[35] M. Bellare, S. Goldwasser, C. Lund, and A. Russell, "Efficient probabilistic checkable proofs and applications to approximation," in *Proc. Annu. ACM Symp. Theory Comput.*, 1994, pp. 294–304.
[36] B. Parno, J. Howell, C. Gentry, and M. Raykova, "Pinocchio: Nearly practical verifiable computation," in *Proc. IEEE Symp. Secur. Privacy*, 2013, pp. 238–252.
[37] E. Ben-Sasson, A. Chiesa, D. Genkin, E. Tromer, and M. Virza, "SNARKs for C: verifying program executions succinctly and in zero knowledge," in *Proc. Annu. Int. Cryptol. Conf.*, 2013, pp. 90–108.
[38] E. Ben-Sasson, A. Chiesa, E. Tromer, and M. Virza, "Succinct non-interactive zero knowledge for a von neumann architecture," in *Proc. USENIX Secur.*, 2014, pp. 781–796.
[39] J. Groth, "On the size of pairing-based non-interactive arguments," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2016, pp. 305–326.
[40] M. Maller, S. Bowe, M. Kohlweiss, and S. Meiklejohn, "Sonic: Zero-knowledge SNARKs from linear-size universal and updatable structured reference strings," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2019, pp. 2111–2128.
[41] A. Gabizon, Z. J. Williamson, and O. Ciobotaru, "PLONK: Permutations over lagrange-bases for oecumenical noninteractive arguments of knowledge," *IACR Cryptol. ePrint Arch.*, vol. 2019, 2019, Art. no. 953.
[42] S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf
[43] G. Wood, "Ethereum: A secure decentralised generalised transaction ledger," *Ethereum Project Yellow Paper*, vol. 151, pp. 1–32, 2014.
[44] I. Stoica *et al.*, "Chord: A scalable peer-to-peer lookup protocol for internet applications," *IEEE/ACM Trans. Netw.*, vol. 11, no. 1, pp. 17–32, Feb. 2003.
[45] Y. Dodis, S. P. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proc. Theory Cryptogr. Conf.*, 2009, pp. 109–127.
[46] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *Proc. Int. Conf. Theory Appl. Cryptol. Inf. Secur.*, 2010, pp. 177–194.
[47] R. W. F. Lai and G. Malavolta, "Subvector commitments with application to succinct arguments," in *Proc. Annu. Int. Cryptol. Conf.*, 2019, pp. 530–560.
[48] A. Tomescu, I. Abraham, V. Buterin, J. Drake, D. Feist, and D. Khovratovich, "Aggregatable subvector commitments for stateless cryptocurrencies," Cryptology ePrint Archive, Rep. 2020/527, 2020.
[49] D. Boneh and X. Boyen, "Short signatures without random oracles," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2004, pp. 56–73.
[50] V. Goyal, "Reducing trust in the PKG in identity based cryptosystems," in *Proc. Annu. Int. Cryptol. Conf.*, 2007, pp. 430–447.
[51] A. Miller, A. Juels, E. Shi, B. Parno, and J. Katz, "Permacoin: Repurposing bitcoin work for data preservation," in *Proc. IEEE Symp. Secur. Privacy*, 2014, pp. 475–490.
[52] D. Deuber, B. Magri, and S. A. K. Thyagarajan, "Redactable blockchain in the permissionless setting," in *Proc. IEEE Symp. Secur. Privacy*, 2019, pp. 124–138.
[53] J. Zhang, B. Wang, D. He, and X. A. Wang, "Improved secure fuzzy auditing protocol for cloud data storage," *Softw. Comput.*, vol. 23, no. 10, pp. 3411–3422, 2019.
[54] A. Miller, A. E. Kosba, J. Katz, and E. Shi, "Nonoutsourceable scratch-off puzzles to discourage bitcoin mining coalitions," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2015, pp. 680–691.
[55] M. Campanelli, R. Gennaro, S. Goldfeder, and L. Nizzardo, "Zero-knowledge contingent payments revisited: Attacks and payments for services," in *Proc. ACM Conf. Comput. Commun. Secur.*, 2017, pp. 229–243.
[56] "Tahoe-lafs," 2019. [Online]. Available: https://tahoe-lafs.org/trac/tahoe-lafs
[57] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, May 2010.
[58] D. Boneh, J. Bonneau, B. Bünz, and B. Fisch, "Verifiable delay functions," in *Proc. Annu. Int. Cryptol. Conf.*, 2018, pp. 757–788.
[59] K. Pietrzak, "Simple verifiable delay functions," in *Proc. Innovations Theor. Comput. Sci. Conf.*, 2019, pp. 60:1–60:15.
[60] B. Wesolowski, "Efficient verifiable delay functions," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2019, pp. 2113–2147.
[61] N. Ephraim, C. Freitag, I. Komargodski, and R. Pass, "Continuous verifiable delay functions," in *Proc. Annu. Int. Conf. Theory Appl. Cryptogr. Techn.*, 2020, pp. 125–154.

Yuefeng Du (Student Member, IEEE) received the BSc degree in 2018 in computer science from the City University of Hong Kong where he is currently working toward the PhD degree. His research interests include applied cryptography, blockchain, and network security.



Huayi Duan (Student Member, IEEE) received the BS degree with First Class Honors and the PhD degree from the City University of Hong Kong in 2015 and 2020, respectively. His research interests include network security, privacy and trust enhancing technologies, and applications and foundations of blockchain. He is particularly avid for usable techniques built from the choreography of elegant theoretical constructions and good engineering practices.

Anxin Zhou (Student Member, IEEE) received the BE degree in computer science and technology from the University of Science and Technology of China in 2018. He is currently working toward the PhD degree with the Deparment of Computer Sciecne, City University of Hong Kong. His research interests include secure computation, blockchain, data privacy.

Cong Wang (Fellow, IEEE) is currently a professor with the Department of Computer Science, City University of Hong Kong. His current research interests include data and network security, blockchain and decentralized applications, and privacy-enhancing technologies. He was an associate editor for the *IEEE Transactions on Dependable and Secure Computing*, *IEEE Internet of Things Journal, IEEE Networking Letters*, and *The Journal of Blockchain Research*, and TPC co-chairs for a number of IEEE conferences and workshops. He is one of the founding members of the Young Academy of Sciences of Hong Kong. He was the recipient of the Outstanding Researcher Award (junior faculty) in 2019, the Outstanding Supervisor Award in 2017, and the President's Awards in 2019 and 2016, all from City University of Hong Kong. He was the co-recipient of the IEEE INFOCOM Test of Time Paper Award 2020, Best Paper Award of IEEE ICDCS 2020, Best Student Paper Award of IEEE ICDCS 2017, and the Best Paper Award of IEEE ICPADS 2018 and MSN 2015. His research has been supported by multiple government research fund agencies, including National Natural Science Foundation of China, Hong Kong Research Grants Council, and Hong Kong Innovation and Technology Commission. He is member of the ACM.

Man Ho Au (Member, IEEE) received the BEng and MPhil degrees from the Department of Information Engineering, The Chinese University of Hong Kong, in 2003 and 2005, respectively, and the PhD degree from the School of Computer Science and Software Engineering, University of Wollongong, in 2009. He is currently an associate professor with the Department of Computer Science, The University of Hong Kong (HKU). Prior to joining HKU, he was an associate professor with the Department of Computing, The Hong Kong Polytechnic University. He has authored or coauthored more than 160 refereed articles in top journals and conferences, including CRYPTO, ASIACRYPT, ACM CCS, ACM SIGMOD, NDSS, *IEEE Transactions on Information Forensics and Security*, *IEEE Transactions on Computers*, and *IEEE Transactions on Knowledge and Data Engineering*. His research interests include applied cryptography, information security, blockchain technology, and related industrial applications. He was the recipient of the 2009 PET Runner-Up Award for outstanding research in privacy enhancing technologies. His digital signature technology has been adopted by the Hyperledger Fabric Project. He is also an expert member of the China delegation of ISO/IEC JTC 1/SC 27 Working Group 2: Cryptography and Security Mechanisms and a Committee Member of the Research and Development Division, Hong Kong Blockchain Society.

Qian Wang (Senior Member, IEEE) received the PhD degree from the Illinois Institute of Technology, USA. He is currently a professor with the School of Cyber Science and Engineering, Wuhan University. His research interests include AI security, data storage, search and computation outsourcing security and privacy, wireless systems security, big data security and privacy, and applied cryptography. He is a member of ACM. He is also an expert under National "1000 Young Talents Program" of China. He was the recipient of the National Science Fund for Excellent Young Scholars of China in 2018, the 2018 IEEE TCSC Award for Excellence in Scalable Computing (Early Career Researcher), and the 2016 IEEE Asia–Pacific Outstanding Young Researcher Award. He was also the co-recipient of several best paper and best student paper awards from the IEEE ICDCS'17, the IEEE TrustCom'16, WAIM'14, and the IEEE ICNP'11. He is an associate editors for the *IEEE Transactions on Dependable and Secure Computing* and the *IEEE Transactions on Information Forensics and Security* .