Privacy-Preserving Proof of Storage for the Pay-As-You-Go Business Model

Tong Wu, Guomin Yang, *Senior Member, IEEE*, Yi Mu, *Senior Member, IEEE*, Fuchun Guo, and Robert H. Deng, *Fellow, IEEE*

Abstract—Proof of Storage (PoS) enables a cloud storage provider to prove that a client's data is intact. However, existing PoS protocols are not designed for the pay-as-you-go business model in which payment is made based on both storage volume and duration. In this paper, we propose two PoS protocols suitable for the pay-as-you-go storage business model. The first is a time encapsulated Proof of Retrievability (PoR) protocol that ensures retrievability of the original file upon successful auditing by a client. Considering the large size of outsourced data, we then extend the protocol to a privacy-preserving public auditing protocol which allows a third party auditor to audit outsourced data on behalf of its clients without sacrificing the privacy of the data or the timestamp (i.e., time of storage). We formalize the definition, system model and security model of the proposed PoS system and prove the security of the proposed protocols by a sequence of games in the algebraic group model with a random oracle. We analyze the performance of the protocols both theoretically and experimentally and show that the protocols are practical.

Index Terms—Remote integrity checking, proof of storage, data security and privacy, pay-as-you-go, third party auditor

1 Introduction

Cloud computing deals with massive volume of data via powerful computation resources and elastic storage capability and brings significant benefits to cloud clients [18], such as allowing on-demand access to the outsourced data and relieving clients from the burden of local storage management and maintenance. However, it is also facing a range of internal/external attackers who illegally access, delete or corrupt the outsourced data; thus, it entails the security risks in terms of confidentiality, integrity and availability of the outsourced data and service. Among these security issues, the integrity of outsourced data is of great importance, since the clients do not possess their outsourced data locally. On the other hand, a cloud service provider (CSP) may be dishonest and attempts to hide data loss or corruption. For examples, the CSP might reclaim storage by discarding data that have not been or are rarely accessed, or even hide data loss incidents to maintain a reputation [17]. Thus it is crucial for clients to implement an effective mechanism to perform periodical integrity checking over the outsourced data to ensure that it is intact in the cloud [14].

Proof of Storage (PoS) is a generic primitive that allows a party to verify that the prover actually stores a file intact [3]. There are two popular PoS models, provable data possession (PDP) [2], [4], [6], [15], [16], [23], [30], [33], [35] and proof of retrievability (PoR) [9], [21], [27], [28], each has its pros and cons. PDP allows a client that has stored data at an untrusted server to verify that the server possesses the original data without retrieving it. The model generates probabilistic proofs of possession by sampling random sets of blocks. PoR ensures that a client can recover the entire file when an auditing is successful and is consider to be more robust than PDP model for remote integrity checking in cloud. On the other hand, considering the large size of outsourced data and clients' constraint on local storage capability, it is desirable to implement a public auditing protocol in which integrity checking is performed by a third party auditor (TPA) while without revealing the clients data during the auditing process.

To impel cloud services like public utilities such as water, gas and electricity, it is ideal to run the pay-as-you-go business model in which a client pays CSP based on the storage volume and storage duration, as illustrated in Fig. 1. It means a cloud client can update his/her files stored in the cloud on different dates and the CSP will calculate the cloud storage fee for each day according to the total volume occupied by the client on that day.

The new payment model can be integrated with a proof of storage system to handle file damage or corruption situations. The CSP has the responsibility to ensure the integrity of the files stored in the cloud. If a file corruption is detected in a periodical integrity checking, then the CSP should only charge the storage fee for that file until the last valid checking. More specifically, there are three possible situations: the first case is that no damage is detected in periodical integrity checking, then the client pay the bill in a regular way; the second case is that

- *T. Wu, G. Yang, and F. Guo is with the Institute of Cybersecurity and Cryptology, School of Computing and Information Technology, University of Wollongong, Wollongong, NSW 2522, Australia. E-mail: tw225@uowmail.edu.au, {gyang, fuchun}@uow.edu.au.*
- *Y. Mu is with the Fujian Provincial Key Laboratory of Network Security and Cryptology, Fujian Normal University, Fuzhou 350007, China. E-mail: ymu.ieee@gmail.com.*
- *R.H. Deng is with the School of Information Systems, Singapore Management University, Singapore. E-mail: robertdeng@smu.edu.sg.*

Manuscript received 25 June 2018; revised 15 June 2019; accepted 4 July 2019. Date of publication 29 July 2019; date of current version 12 Mar. 2021. (Corresponding author: Guomin Yang.)
Digital Object Identifier no. 10.1109/TDSC.2019.2931193

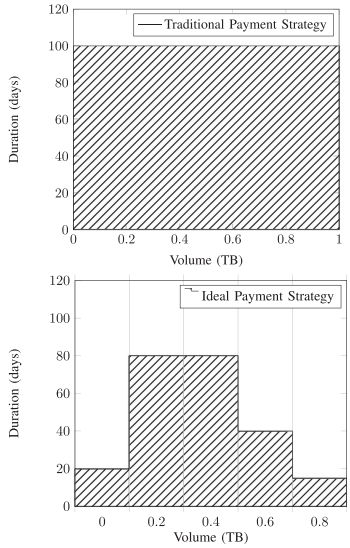


Fig. 1. Illustration of the difference between the traditional payment and the ideal payment: the financial cost is computed by the size of the shaded area.

an outsourced file is damaged and detected in the integrity checking, then the storage charge will be counted by the date of the last successful checking; the third case is the client removes a file from cloud storage, then the CSP calculates the storage fee by this date. The above possible situations are shown in Fig. 2.

However, to our knowledge, the existing PoS protocols only allow content integrity checking. To cater the pay-as-you-go model, PoS protocols need to support integrity checking of both content and timestamp (i.e., time of storage) of the outsourced data. A trivial solution is to append a timestamp at the end of the outsourced data or using special data blocks to store the timestamp. However, it has a loose relationship between the timestamp and individual data blocks and the timestamp may be lost or corrupted. Another solution is that the file name and the timestamp can be incorporated to generate an authentication tag for each block. This approach ensures a strong binding between the timestamp and individual blocks but the timestamp has to be known by the verifier which may be a privacy concern in the TPA setting. We should note that the timestamp may leak some information related to the data via inference. For example, if the TPA knows the date of a particular meeting, then from the timestamp the TPA is able to locate the set of documents that contain the meeting minutes. This work presents new PoS solutions that allow content and timestamp integrity checking without the aforementioned limitations. Our proposed PoS schemes also have other potential applications such as version control for outsourced data where the integrity of different versions of a file and their corresponding timestamps can be guaranteed.

1.1 Contributions

In this paper we propose two PoS protocols designed for the pay-as-you-go business model.

- We first present a PoR protocol which allows a client to verify the integrity of data and its timestamp and ensures retrievability of the data upon successful auditing.
- We then propose a privacy-preserving public auditing protocol in which integrity of outsourced data and the timestamp can be efficiently verified by a TPA while no information on the content or the timestamp is leaked to the TPA.
- We show that both protocols are sound, i.e., they can detect any modification or absence of the outsourced data with overwhelming probability. We prove their security properties formally against the algebraic adversaries [20], [22] with a random oracle.
- We compare our protocols with several related protocols in functionality and efficiency, and conduct experiments to measure their performances. Both the theoretical comparison and the experimental results confirm that our protocols are efficient for practical applications.

1.2 Related Work

The traditional data integrity checking techniques require a client to download an entire data file or store some meta-data locally [14] and hence are not suitable for the cloud storage setting. Considerable research under various security models have been carried out to address integrity checking of outsourced data without requiring the client to have a local copy or keep meta-data. Among them the most significant work are the PoR protocols and the PDP protocols. In 2007, Juels and Kaliski [21] proposed the notion of PoR which enables a (semi-trusted) storage server to produce a proof that the client can retrieve the entire file. However, PoR is a position-care scheme in that the number of queries is restricted by the number of sentinel blocks. Position-care means the integrity checking is

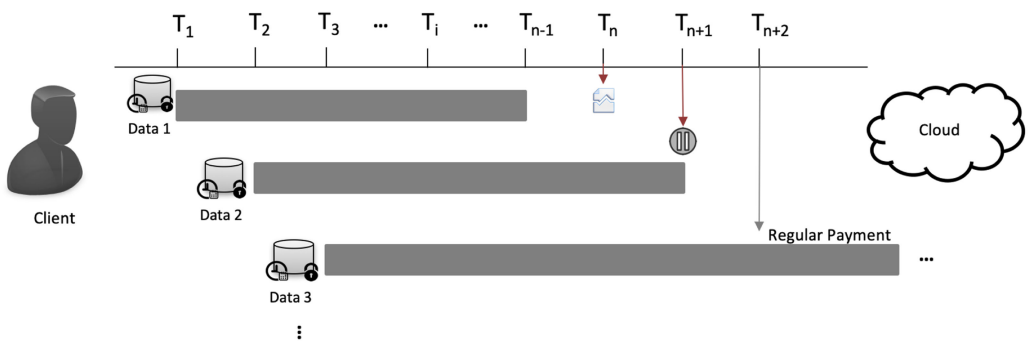


Fig. 2. Cloud data integrity checking integrated into the Pay-As-You-Go business model.

conducted by checking positions of the special blocks in the file. In the same year, Ateniese et al. [2] introduced the notion of PDP model and constructed two integrity checking protocols which are based on RSA cyptosystem and homomorphic linear authenticators (HLA). The two protocols can detect data corruption with high probabilities by checking some random sampling blocks. Since then, HLA has been widely applied in the design of many integrity checking protocols to achieve the blockless verification which dramatically reduces the protocol overhead in computation and communication. Blockless verification enables integrity checking over the outsourced data to be conducted without giving the data itself to the verifier. Inspired by [2], Shacham and Waters [27] proposed the compact PoR protocols and proved their security of the PoRs by a sequence of games in the random oracle model based on the intractability of Computational Diffie-Hellman (CDH) problem. For the purpose of achieving shortest query and response, they adopted the linear homomorphic property of BLS signatures [8] to construct the HLA. Later, Ateniese et al. [3] introduced a formal transformation from homomorphic identification protocols to the PoS protocols. In [3], the transformation requires the homomorphic identification protocols equipped with three combination functions. However, the generic construction is restricted by the combination algorithms, slightly differing from our construction. In [12], the authors construct the PoS by combining the message-locked encryption scheme and the Merkle Hash Tree. Their approach does not improve the efficiency or the security of PoS.

In fact, the aforementioned protocols may potentially leak information to auditors during the audit phase [30]. To address the privacy issue, Wang et al. [31] proposed the first privacy-preserving public auditing protocol based on the work in [27] and the random masking technique that the linear combination of sampled blocks in the response is masked with randomness [31]. The TPA performs the public auditing tasks on behalf of the clients. Furthermore, it cannot extract the original data during the audit phase, since the data is hidden by a zero-knowledge proof. To improved the efficiency, Worku et al. proposed a scheme [36]. Both [31] and [36] are vulnerable to existential forgeries using known message attacks from the malicious CSP [13], [24]. To improved the privacy, a privacy-enhanced protocol was introduced in [16] which achieves the indistinguishable privacy but at the cost of significantly increased computational overhead. Recently, Liu et al. [25] proposed a privacy-preserving public auditing protocol completely releases data owners from online burden by regenerating code [10], [11]. A proxy is introduced to generate authenticators. The audit is conducted by TPA. However, all the aforementioned privacy-preserving auditing mechanisms are content-only and cannot support the validation of timestamp.

Recently, there are some works aiming to improve the efficiency of the integrity checking service or offering a more user-friendly service, such as the lightweight privacy-preserving public auditing via online/offline strategy [23], [35] and identity-based remote integrity checking [32], [34], [40]. Additionally, some researchers became concerned to the key-exposure problem. To address this problem, they proposed the key-exposure resilience auditing scheme and key update strategy [37], [38], [39]. Unfortunately, the aforementioned works are also content-only integrity checking schemes and not yet offer any effective and practical approach to support the validation of timestamp.

1.3 Organization

The rest of the paper is organized as follows: In Section 2, we introduce the system architecture and security models. In Section 3, we provide some preliminaries. Then, we give concrete description of our protocols with/without TPA in Section 4 and Section 5, respectively. The theoretical comparison with related works and the experimental evaluation are shown in Section 6. Finally, we conclude our work in Section 7.

2 System Model and Security Definition

In this section, we give the system architecture, algorithm definition and security model of our protocols.

2.1 System Model

We consider two types of PoS protocols. In the first type of protocols, a cloud client verifies integrity of the outsourced data and validity of the timestamp, while in the second type, verification is performed by a TPA but without leaking any information to the TPA.

There are two parties in our PoS systems, namely the cloud client and the CSP. In the second setting, the PoS system additionally involves a TPA. Each entity has its own obligations and benefits. The CSP, for its own benefit, such as to maintain a good reputation, might decide to hide data corruption incidents to cloud clients. However, we assume that the CSP has no incentive to reveal the outsourced data to TPA. The cloud client in the PoR protocol is to perform the auditing. While in public auditing, the TPA performs the auditing on behalf cloud clients, but it is also curious on the content of outsourced data. The system models with/without TPA are illustrated in Fig. 3.

The significant difference between our PoS protocols and the related works is that our PoS protocols support the verification of timestamp in addition to the outsourced data, which allows our protocols to be integrated with the pay-as-you-go payment model.

System Components. Based on the work of Ateniese et al.'s [3], a proof of storage system $\Sigma = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{GenProof}, \mathsf{VerifyProof})$ is an interactive protocol that allows a verifier to verify that a prover is faithfully storing a file.

Setup. This probabilistic algorithm takes the security parameter and generates system parameters.

KeyGen. This probabilistic algorithm takes security parameter and public parameters and generates a key-pair for a client.

TagGen. This probabilistic algorithm takes a secret key, a file and timestamp as input. It processes the file to produce the encoded file and the auxiliary information including the file tag and handles for data blocks.

GenProof,VerifyProof. The probabilistic algorithms executed by the prover $\mathcal{P}$ and verifier $\mathcal{V}$ define a protocol for proving the integrity of stored data and the timestamp. During the protocol execution, both $\mathcal{P}$ and $\mathcal{V}$ take the public key pk and a file tag τ generated by TagGen as the input. $\mathcal{P}$ also takes the processed file and timestamp $F^*, \hat{t}$, and the handles $\{\sigma\}$ as input. At the end of the protocol, $\mathcal{V}$ outputs 0 or 1, where 1 means that the file is intact on the server. The protocol can be denoted as:

$$\{0,1\} \leftarrow (\mathcal{V}(pk,\tau) \rightleftharpoons \mathcal{P}(pk,\tau,F^*,\{\sigma\},\hat{t})).$$

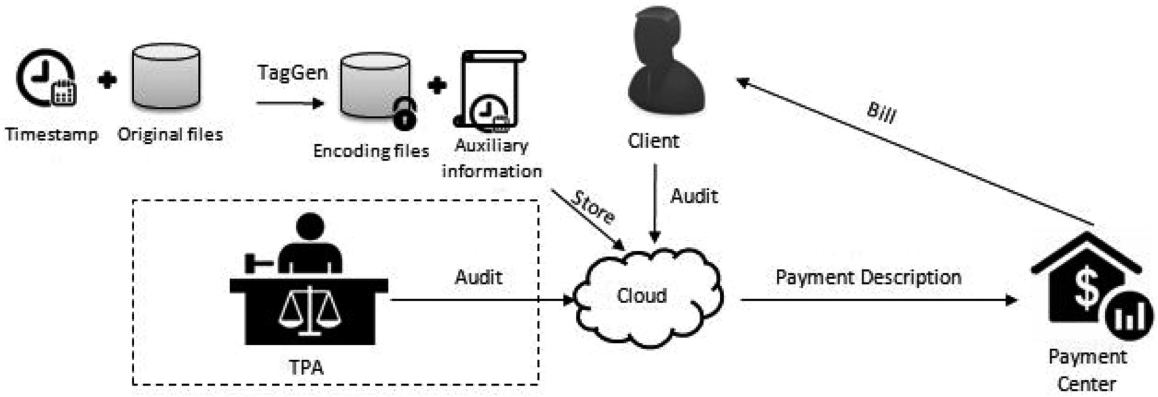


Fig. 3. System model of PoS with/without TPA.

2.2 Design Goals

Our protocols are designed to achieve the following goals:

- *Verifiability.* The verifier is able to verify the integrity of the content and the validation of the timestamp simultaneously, by the proof from the CSP, without storing a local copy.

Definition 1 (Verifiability). *If the prover's response is generated from the intact file and timestamp, then the verification algorithm accepts the proof made by such prover with overwhelming probability.*

- *Soundness.* The CSP should not be able to produce a proof accepted by VerifyProof without possession of the original outsourced data or timestamp.

We define the soundness of our protocol formally. The soundness is defined using the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

Setup. $\mathcal{C}$ runs Setup and KeyGen to initial the environment. It gives public key to $\mathcal{A}$, keeping the private key to itself.

Tag Query. $\mathcal{A}$ adaptively issues queries. When receiving each query for a file and a timestamp, $\mathcal{C}$ runs TagGen to get the output, and forwards the result to $\mathcal{A}$.

Proof-Verify Query. For any file on which it previously made a tag query, the adversary can undertake executions of the proof of storage protocol. In these protocol executions, $\mathcal{C}$ plays the part of the verifier and the adversary plays the part of the prover.

Challenge. $\mathcal{A}$ finally outputs the description of a cheating prover $\mathcal{P}'$ for a new/modified file-timestamp pair different from those appeared in Tag Queries. $\mathcal{C}$ generates a challenge *chal* and requests $\mathcal{P}'$ to respond a proof.

Definition 2 (Soundness). *A publicly verifiable PoS has soundness if the verification algorithm accepts the proof produced by $\mathcal{P}'$ only with negligible probability.*

- *Retrievability.* Informally, retrievability means the verifier is able to recover the file when the prover can general proofs that can pass verification. We say a prover $\mathcal{P}'$ is ϵ-admissible if it convincingly answers the verification challenges with ϵ probability, i.e., if $\Pr[(\mathcal{V}(pk,\tau) \rightleftharpoons \mathcal{P}') = 1] \geq \epsilon$.

Definition 3 (Retrievability). *[27] A PoS protocol with retrievability is ϵ-sound, if there exists a polynomial-time extraction algorithm denoted* Extr, *such that for any adversary that outputs an ϵ-admissible prover $\mathcal{P}'$ for a file M,* Extr *recovers M from $\mathcal{P}'$ except with negligible probability.*

- *Content and Timestamp Privacy.* The TPA should not be able to deduce data or timestamp of the outsourced data from the response given by CSP. We define the privacy of our PDP protocol formally. The content privacy agains TPA is defined using the following game between adversary $\mathcal{A}$ and challenger $\mathcal{C}$.

Setup. $\mathcal{C}$ runs Setup, KeyGen and TagGen to initial the environment. It gives the public key to $\mathcal{A}$, keeping the private key to itself. It also sends to $\mathcal{A}$ a list of public information over the outsourced files, including file tags and timestamps.

Proof-Verify. $\mathcal{C}$ and $\mathcal{A}$ perform the proof-verify protocol. $\mathcal{C}$ acts as the prover to generate the proof over the outsourced file after receiving the challenge from $\mathcal{A}$.

At the end of the game, $\mathcal{A}$ outputs the content of a file. We say $\mathcal{A}$ wins the game if $\mathcal{A}$ outputs the content correctly.

Definition 4 (Content Privacy). *A publicly verifiable PoS has the content privacy if receiving the response, the verifier can recover the file content with negligible probability.*

We also define the time privacy agains TPA formally. *Setup* and *Proof-Verify* are the same as the defined game for the content privacy except that $\mathcal{A}$ is given the file contents instead of timestamps. It finally outputs the timestamp of a file. We say $\mathcal{A}$ wins the game if $\mathcal{A}$ outputs the correct timestamp.

Definition 5 (Time Privacy). *A publicly verifiable PoS has the time privacy if receiving the response, the verifier can recover the timestamp with the probability no more than $\frac{1}{\#T} + \mathsf{negl}(\mathsf{k})$ where $\mathsf{negl}(\cdot)$ denotes a negligible function and $\#T$ denotes the size of the timestamp space.*

3 Preliminary

In this section, we review some preliminaries, including the bilinear map, the computational hardness assumptions and some security primitives.

Definition 6 (Bilinear Group). *$\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ constitute a bilinear group if there exists a bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $|\mathbb{G}_1| = |\mathbb{G}_2| = |\mathbb{G}_T| = p$.*

The bilinear map is an operation conducted on bilinear groups. Informally, two elements in such group are linearly related to the pairing result. The formal description of the bilinear map is given as follows.

Definition 7 (Bilinear Map). *Suppose that $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are three cyclic groups with the same prime order p. Suppose that g and h are generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. A bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ holds properties as follows:*

(1) Bilinearity: For any $x \in \mathbb{G}_1$, $y \in \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p^*$, $e(x^a, y^b) = e(x, y)^{ab}$.
(2) Non-degeneration: $e(g, h) \neq 1_{\mathbb{G}_T}$, where $1_{\mathbb{G}_T}$ is the generator of $\mathbb{G}_T$.
(3) Computability: There exists an efficient algorithm to compute $e(x, y)$, for any $x \in \mathbb{G}_1$ and $y \in \mathbb{G}_2$.

We say that a pairing is symmetric if $\mathbb{G}_1 = \mathbb{G}_2$. Otherwise, it is asymmetric. In asymmetric pairing, there are two types depending on whether there is an isomorphism from $\mathbb{G}_2$ to $\mathbb{G}_1$.

The security of our schemes will rely on the following computational assumptions.

Definition 8 (q-Strong Diffie-Hellman (q-SDH) Assumption). *Given $(g_1, g_1^x, g_1^{x^2}, \cdots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$ it is hard to compute $(c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}_1$.*

Definition 9 (Double Pairing (DP) Assumption [19]). *Given $(g_R, g_T) \in \mathbb{G}_1^2$ it is hard to find non-trivial $(R, T) \in \mathbb{G}_2^2$ satisfying $e(g_R, R)e(g_T, T) = 1$.*

Definition 10 (A Variant of Computational Diffie-Hellman (VoCDH) Assumption). *[29] Given $(g, g^a, g^{\frac{1}{a}}, g^b) \in \mathbb{G}_1^4$ it is hard to compute g^{ab}.*

Definition 11 (Extended Discrete-Logarithm (Extended-DL) Assumption). *Given $(g_1, g_1^x, g_2, g_2^{\frac{1}{x}}) \in \mathbb{G}_1^2 \times \mathbb{G}_2^2$ it is hard to compute x.*

The VoCDH assumption is reducible to the extended-DL assumption. The VoCDH solver sets $g_1^x = g^a$, $g_2 = g^c$ and $g_2^{\frac{1}{x}} = g^{\frac{c}{a}}$, where c is chosen from $\mathbb{Z}_p$. If the extended-DL solver can compute x that is equal to the value of a, the VoCDH solver can compute g^{ab}.

Definition 12 (Difference Lemma). *Let A, B, F be events defined in some probability distribution, and suppose that $\mathrm{A} \wedge \neg \mathrm{F} \Leftrightarrow \mathrm{B} \wedge \neg \mathrm{F}$. Then $|\Pr[\mathrm{A}] - \Pr[\mathrm{B}]| \leq \Pr[\mathrm{F}]$.*

Digital Signature. A digital signature scheme consists of four algorithms (Setup,KeyGen,Sign,Verify).

- Setup(1^λ): it takes security parameter λ and outputs system parameters pp.
- KeyGen(pp): it takes system parameters pp and outputs public/private key pairs $(\mathtt{spk}, \mathtt{ssk})$.
- Sign$(\mathtt{ssk}, m)$: it takes private key ssk and message m. Then, it produces a signature $\sigma = \mathsf{Sig}_{\mathtt{ssk}}(\mathsf{m})$.
- Verify$(\mathtt{spk}, m, \sigma)$: it takes the public key spk, message m and a signature σ and returns 1 for accept, 0 for reject.

4 Our Proposed PoR Protocol

Our PoR protocol follows the linear homomorphic authenticator structure used by the previous constructions. However, one challenging issue we need to address is to embed the timestamp in the authenticator such that it can prevent unauthorized modification of the timestamp while maintaining the aggregation property.

Our PoS protocol under PoR model $\Omega = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{GenProof}, \mathsf{VerifyProof})$ consists of five algorithms which are as follows:

Setup (1^λ). It takes the security parameter λ and outputs the public parameters including bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ with prime order p and bilinear map $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$. It sets $g_1 \in \mathbb{G}_1$, $g_2 \in \mathbb{G}_2$ as generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. It chooses collision-resistant hash functions (CRHFs) $H : \{0,1\}^* \to \mathbb{G}_1$ and $H_0 : \{0,1\}^* \to \mathbb{Z}_p$. It also chooses an existentially unforgeable signature scheme Sig. Then it publishes the public parameters

$$pp = (\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, p, g_1, g_2, H, H_0, \mathsf{Sig}).$$

KeyGen$(1^\lambda, pp)$. It takes the security parameter λ and public parameters pp. Then it outputs a pair of public/private key $\mathtt{pk} = (U, \mathtt{spk})$, $\mathtt{sk} = (\alpha, \mathtt{ssk})$, where $U = g_2^\alpha$. $(\mathtt{spk}, \mathtt{ssk})$ is the key pair of the signature scheme.

TagGen $(F, t, \mathtt{sk})$. It takes the file F with timestamp t to be stored in the cloud and client's sk, and outputs the encoded file and timestamp $(F^*, \hat{t})$, where $\hat{t} = H_0(t, \mathsf{Sig}_{\mathtt{ssk}}(t))$. Then, it breaks F^* into blocks, where m_i denotes the ith block of F^*. Each block contains s sectors that m_{ij} denotes the jth sector of ith block . It chooses a random file name $fname$ from a sufficiently large domain and chooses s random elements $u_1, u_2, \cdots, u_s \in_R \mathbb{G}_1^s$. Then compute the file tag denoted as $\tau = \tau_0 || \mathsf{Sig}_{\mathtt{ssk}}(\tau_0)$, where $\tau_0 = fname||n||u_1||\cdots||u_s$. Also, it generates σ_i as the handle to check that the CSP retains the corresponding block later

$$\sigma_i = (H(fname||i) \cdot \prod_{j=1}^{s} u_j^{m_{ij}})^{\frac{1}{\alpha+\hat{t}}}.$$

The CSP stores $(F^*, \hat{t}, \tau, \{\sigma_i\})$.

GenProof $(Q, \mathtt{pk}, \hat{t}, \{m_{ij}\}, \{\sigma_i\})$. The prover receives the c-element set $\{\theta_i, \nu_i\}$ to be challenged, denoted as Q, where $\theta_i \in_R I$, I is the indices set of blocks in file F^* and ν_i is a random number in $\mathbb{Z}_p$.

Using the stored information, it computes a proof $\pi = (\zeta, \{\mu_j\}, R, RI, d)$. Let $\mathcal{N}$ denote $\prod_{k=1}^{c} \nu_k$ and $\mathcal{N}_i$ denote $\prod_{k=1, k\neq i}^{c} \nu_k$.

$$R = g_2^r, \quad RI = g_1^{\frac{1}{r}}, \qquad r \in_R \mathbb{Z}_p,$$
$$d = r + \hat{t}\mathcal{N},$$
$$\mu_j = \sum_{i=1}^{c} m_{\theta_i j}\mathcal{N}_i, \qquad 1 \leq j \leq s,$$
$$\zeta = \prod_{i=1}^{c} \sigma_{\theta_i}^{\frac{1}{\nu_i}}, \qquad 1 \leq i \leq c.$$

VerifyProof$(Q, \mathtt{pk}, \pi, \tau)$. The verifier checks the validity of τ by spk and

$$e(RI, R) = e(g_1, g_2).$$

Then it verifies if

$$e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu_j}, g_2\right) = e\left(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right).$$

It outputs 1 if the above equations are held. Otherwise, output 0.

4.1 Verifiability

The correctness of the above verification equation is clear.

$$e\left(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} \sigma_i^{\frac{1}{\nu_i}}, g_2^{\hat{t}\mathcal{N}} \cdot g_2^{\alpha\mathcal{N}}\right)$$

$$= e\left(\prod_{i=1}^{c} (H(fname||\theta_i) \prod_{j=1}^{s} u_j^{m_{\theta_i j}})^{\frac{1}{\nu_i}}, g_2^{\mathcal{N}}\right)$$

$$= e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu_j}, g_2\right).$$

Note: There is an attack if the proof is without RI.

Specifically, the CSP randomly chooses $d, \{\mu_j\}$ and lets $R = g_2^{d-1} \cdot U^{\mathcal{N}}$ and $\zeta = \prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu_j}$. By this way, CSP can generate a proof $(R, d, \{\mu_j\}, \zeta)$ accepted by the verification algorithm without the knowledge of the stored data since

$$e(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu_j}, g_2\right).$$

This attack is caused by the CSP who does not need to prove the knowledge of the discrete logarithm of R. Hence, we provide $RI = g_1^{\frac{1}{r}}$ to prove the knowledge of the discrete logarithm of R [7].

4.2 Soundness against CSP

Theorem 1. *If the signature scheme in our PoS is existentially unforgeable and the q-SDH, DP assumptions hold in bilinear groups, then there is not any PPT adversary against the soundness of our verification algorithm in the algebraic group model with a random oracle, that causes $VerifyProof$ to accept a PoS instance, except that it is computed correctly.*

Following the security proof provided by Shacham and Waters in [27], we define a sequence of games Game 0, Game 1, Game 2 and Game 3, played between the simulator $\mathcal{B}$ and the adversary $\mathcal{A}$. Before playing games, $\mathcal{B}$ initializes the environment by running Setup.

Game 0. $\mathcal{A}$ interacts with $\mathcal{B}$ in the originally challenged game as defined in Section 2.2.

Game 1. It is the same as Game 0 expect that if $\mathcal{A}$ submits a file tag that is valid but not generated by $\mathcal{B}$. $\mathcal{B}$ then declares failure and aborts. It implies that $\mathcal{A}$ can break the underlying signature scheme.

Game 2. It is the same as Game 1, except that $\mathcal{B}$ keeps a list of its responses to $\mathcal{A}$'s Tag Queries. $\mathcal{B}$ observes $\mathcal{A}$'s responses over prove-verify or in the test made of $\mathcal{P}'$. If the adversary makes VerifyProof outputs 1 in any instance but the aggregated value ζ is not equal to $\prod_{(\theta_i,\nu_i)\in Q} \sigma_i^{\frac{1}{\nu_i}}$, where Q is the challenge issued by $\mathcal{B}$, $\mathcal{B}$ declares failure and aborts.

Game 3. As in Game 2, the challenger tracks Tag Queries and observes proof of storage protocol instances. This time, if in any of these instances the adversary is successful to make VerifyProof accept but at least one μ_j is not equal to the expected $\prod_{(\theta_i,\nu_i)\in Q} m_{\theta_i j}\mathcal{N}_i$, $\mathcal{B}$ declares failure and aborts.

Lemma 1. *The difference in success probabilities between Game 0 and Game 1 is negligible if the signature scheme is existentially unforgeable.*

Proof. If $\mathcal{A}$ submits a file tag on a file that is not generated by $\mathcal{B}$, $\mathcal{A}$ successfully breaks the existential unforgeability of the underlying signature scheme Sig. According to the difference lemma, we have that $|\mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game0}} - \mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game1}}| \leq \epsilon_{\mathsf{Sig}}$. □

Lemma 2. *The difference in success probabilities between Game 1 and Game 2 is negligible, if the q-SDH assumption holds in bilinear groups.*

Proof. The proof of Lemma 2 is given in Appendix A. □

Lemma 3. *The difference in success probabilities between Game 2 and Game 3 is negligible, if the DP assumption holds in bilinear groups.*

Proof. The proof of Lemma 3 is given in Appendix B. □

Wrapping up. From Lemmas 1, 2 and 3, by adopting difference lemma, assuming the signature scheme is secure and the q-SDH and the DP assumptions hold in bilinear groups, there is only a negligible difference in the success probabilities between Game 3 and Game 0. This completes the proof of Theorem 1.

4.3 Extractability

The extractability holds in our PoS under PoR model. It is to prove that the extraction procedure can efficiently reconstruct a ρ fraction of the outsourced data when interacting with a prover that provides correctly-computed $\{\mu_j\}$ responses for a non-negligible fraction of the query space. The above proofs guarantee that all adversaries that win the soundness game with non-negligible probability output cheating provers that are well-behaved. We say that a cheating prover is well-behaved if it never causes verification algorithm to accept in a prove-verify protocol instance except that is computed correctly. The proof of the extractability of $\{\mu_j\}$ over polynomial time queries is following the proof of Theorem 4.3 in [27].

4.4 Retrievability

To retrieve the checking file, the verifier follows the method used in SW PoR [27]. Let n be the number of blocks in the file. Suppose we use a ρ-rate erasure code, i.e., one in which any ρ-fraction of the blocks success for decoding. Our proofs guarantee that extraction will succeed from any adversary that convincingly answers an κ-fraction of queries, provided that $\kappa - \rho^l - 1/\#\{\nu\}$ is non-negligible in λ. It is this requirement that guides the choice of parameters. A conservative choice is $\rho = 1/2, l = \lambda$ and $\#\{\nu\} = 2^\lambda$. This guarantees extraction against any adversary. For applications that can tolerate a larger error rate these parameters can be reduced.

Erasure codes provide the property [1] that the verifier is able to recover the entire file with ρ fraction. Before storing it on the server, we would therefore like to encode an n-block file into a $2n$-block file or more when ρ is less than $\frac{1}{2}$. The traditional Reed-Solomon style erasure codes can be constructed for arbitrary rates allowing recovery of the original file [26].

TABLE 1
Comparison of Computation and Properties

Scheme	GenProof	VerifyProof	Retrievable	Privacy-Preserving	Timestamp
[27]	$cE_1 + (c+1)M_1$	$2P + (c+s)E_1 + (c+s-1)M_1$	✓	N/A	×
[30]	$P + cE_1 + (c-1)M_1$	$2P + (c+3)E_1 + (c+1)M_1$	×	✓	×
Our PoR	$(c+1)E_1 + (c-1)M_1 + E_2$	$3P + (c+s)E_1 + (c+s-2)M_1 + 2E_2 + 2M_2$	✓	N/A	✓
Our PDP	$(c+2)E_1 + (c-1)M_1 + 2E_2$	$5P + (c+1)E_1 + (c+1)M_1 + 2E_2 + 2M_2$	×	✓	✓

5 Our Proposed Protocol with TPA

In some scenarios, the cloud clients may employ a third party to do the audit on behalf of them in order to relieve the burden of doing periodical integrity checking. Meanwhile, the third party should be blind to what it is checking. Therefore, proof of retrievability is not suitable in this scenario. To satisfy the aforementioned requirement, we propose another PoS protocol without retrievability. Our PoS protocol under PDP model $\Omega = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{TagGen}, \mathsf{GenProof}, \mathsf{VerifyProof})$ consists of five algorithms. The Setup and KeyGen are the same as defined before in Section 4.

TagGen $(F, t, \mathtt{sk})$. It takes the file F with timestamp t to be stored in the cloud and client's $\mathtt{sk}$, and outputs the encoded file and timestamp $(F^*, \hat{t})$, where $\hat{t} = H_0(t, \mathsf{Sig}_{\mathtt{ssk}}(t))$. Then, it breaks F^* into blocks, where m_i denotes the ith block of F^*. Each block contains s sectors that m_{ij} denotes the jth sector of the ith block. For simplicity, in this scheme we assume $s = 1$. We then compute the file tag and block handles as follows. It chooses a random file name $fname$ from a sufficiently large domain and chooses a random element $u \in_R \mathbb{G}_1$. Then compute the tag denoted as $\tau = \tau_0 || \mathsf{Sig}_{\mathtt{ssk}}(\tau_0)$, where $\tau_0 = fname||n||u$. Also, it computes σ_i as the handle for m_i as follows

$$\sigma_i = (H(fname||i) \cdot u^{m_i})^{\frac{1}{\alpha+t}}.$$

The CSP stores $(F^*, \hat{t}, \tau, \{\sigma_i\})$.

GenProof $(Q, \mathtt{pk}, \hat{t}, \{m_i\}, \{\sigma_i\})$. The prover receives the c-element set $\{\theta_i, \nu_i\}$ to be challenged, denoted as Q, where $\theta_i \in_R I$, where I is the indices set of blocks in file F^* and ν_i is the prime chosen randomly in $\mathbb{Z}_p$. It computes a proof $\pi = (\zeta, A, AI, \mu, R, RI, d)$. Let $\mathcal{N}$ denote $\prod_{k=1}^{c} \nu_k$ and $\mathcal{N}_i$ denote $\prod_{k=1, k\neq i}^{c} \nu_k$.

$$\begin{aligned}
&A = g_2^a, \quad AI = g_1^{\frac{1}{a}}, && a \in_R \mathbb{Z}_p,\\
&\mu = a + \sum_{i=1}^{c} m_{\theta_i} \mathcal{N}_i,\\
&R = g_2^r, \quad RI = g_1^{\frac{1}{r}}, && r \in_R \mathbb{Z}_p,\\
&d = r + \hat{t}\mathcal{N},\\
&\zeta = \prod_{i=1}^{c} \sigma_{\theta_i}^{\frac{1}{\nu_i}}, && 1 \le i \le c.
\end{aligned}$$

VerifyProof$(Q, \mathtt{pk}, \pi, \tau)$. The verifier checks the validity of τ by $\mathtt{spk}$ and verifies if

$$e(AI, A) = e(g_1, g_2) \wedge e(RI, R) = e(g_1, g_2).$$

Then it verifies whether

$$e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu}, g_2\right) = PA \cdot e(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}})$$

when $PA = e(u, A)$. It outputs 1 if the above equations are held. Otherwise, output 0.

The correctness of the above verification equation is clear.

$$\begin{aligned}
PA \cdot e\left(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) &= e\left(\prod_{i=1}^{c} \sigma_{\theta_i}^{\frac{1}{\nu_i}}, g_2^{\hat{t}\mathcal{N}} \cdot g_2^{\alpha\mathcal{N}}\right)\\
&= e\left(u^a \cdot \prod_{i=1}^{c} u^{m_{\theta_i}\mathcal{N}_i}, g_2\right) \cdot\\
&\quad e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i}, g_2\right)\\
&= e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu}, g_2\right)
\end{aligned}$$

5.1 Soundness against CSP

We prove that the CSP is unable to generate a valid response to the TPA without storing the file and timestamp as it should be.

Theorem 2. *If the signature scheme in our protocol is existentially unforgeable and the q-SDH, DP assumption hold in bilinear groups, then, there is not any PPT adversary against the soundness of our verification algorithm in the algebraic group model with a random oracle, that causes $VerifyProof$ to accept a PoS instance, except that it is computed correctly.*

The proof of Theorem 2 is given in Appendix C.

5.2 Content/Time Privacy against TPA

Theorem 3. *There is not any adversary against the content/time privacy in our PDP protocol if the extended-DL problem is intractable and the signature scheme Sig is secure.*

The proof of Theorem 3 is given in Appendix D.

6 Performance

6.1 Theoretical Comparison

We summarize the computation cost of our protocols in Table 1, which also shows a comparison among our protocols, Wang et al. scheme [30] and the SW PoR scheme [27] over efficiency and functionality.

Computation. The bilinear pairings P, exponentiation E_1 and multiplication M_1 on $\mathbb{G}_1$, and exponentiation E_2 and multiplication M_2 on $\mathbb{G}_2$ contribute most computation cost.

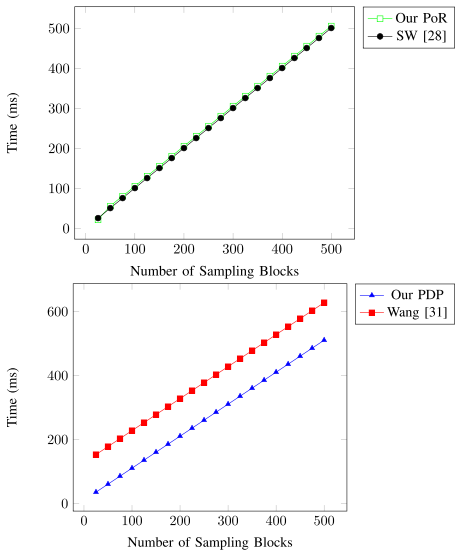


Fig. 4. Time cost of GenProof.

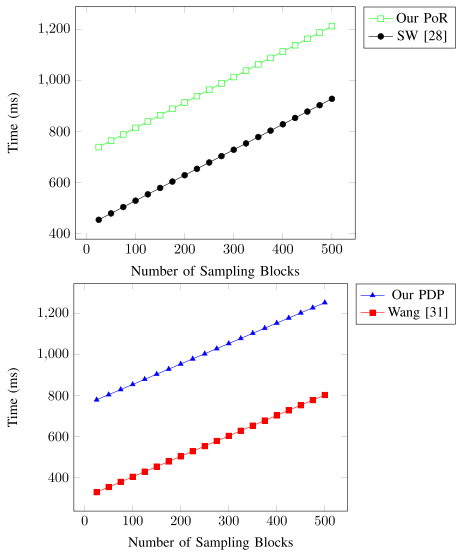


Fig. 5. Time cost of VerifyProof.

The other operations are much faster, such as hash function and the operations on finite fields.

Thus, we only consider bilinear pairings on bilinear groups, exponentiation, and multiplication on $\mathbb{G}_1$ and $\mathbb{G}_2$.

In audit phase of our PoR protocol, to generate a response on a challenge given by verifier, GenProof needs $c+1$ exponentiation and $c-1$ multiplication on $\mathbb{G}_1$, and 1 exponentiation on $\mathbb{G}_2$, respectively. VerifyProof needs $c+s$ exponentiation and $c+s-2$ multiplication on $\mathbb{G}_1$, 1 exponentiation and 1 multiplication on $\mathbb{G}_2$, and 3 bilinear pairings on bilinear groups. In audit phase of our PDP protocol, GenProof needs $c+2$ exponentiation and $c-1$ multiplication on $\mathbb{G}_1$, and 2 exponentiation on $\mathbb{G}_2$, respectively. VerifyProof needs $c+1$ exponentiation and $c+1$ multiplication on $\mathbb{G}_1$, 2 exponentiation and 2 multiplication on $\mathbb{G}_2$, and 5 bilinear pairings on bilinear groups.

Communication. In PoR, we use the aggregated tags as the response in our protocol, such that CSP returns $\pi = (\zeta, \{\mu_j\}, d, R, RI)$. The communication cost is of bit length $|\mathbb{G}_2| + 2 * |\mathbb{G}_1| + (s+1) * |\mathbb{Z}_p|$. We adopted the type f elliptic curve to generate our system. The group element in $\mathbb{G}_1$ is of 160-bit and the group element in $\mathbb{G}_2$ is of 320-bit. $\mathbb{Z}_p$ contains elements of 160-bit. In our experiments, we set the number of sector s in each block as 128. Thus the communication cost of the response is of bit length $21280 = 320 + 2 * 160 + 160 * 129$ for the PoR protocol. In PDP, the communication cost of the response is of bit length $2 * |\mathbb{G}_2| + 3 * |\mathbb{G}_1| + 2 * |\mathbb{Z}_p|$ when s is set to 1. Thus the communication cost of the response is of bit length $1440 = 2 * 320 + 3 * 160 + 2 * 160$.

Storage. In both PoR and PDP, for each file CSP stores the file tag τ and handles $\{\sigma_i\}_1^n$ for the data blocks, making the storage overhead to be roughly $(n+s) * |\mathbb{G}_1|$. During the experiments, we used files containing 500 blocks, 1000 blocks and 5000 blocks, which gives the storage overhead of 12 KB, 22 KB and 100 KB, respectively.

6.2 Experimental Result

We implement the prototype of our protocol and evaluate the time cost in audit phase. The implementation was conducted on a notebook with 2.7 GHz Intel Core i5 processor and 8 GB 1867 MHz DDR3 RAM. We use Java 1.8 and JPBC to implement the cryptographic algorithm. The implementation is done using type f elliptic curve. For more details about the curve, please refer to [5]. For 80-bits security, only 160 bits are needed to represent elements of $\mathbb{G}_1$, and 320 bits for $\mathbb{G}_2$.

In our experiments, we evaluated the performance of our protocols in audit phase by setting the number of the sampling blocks from 0 to 500. As expected, the computation time grows linearly as the verifier requests more sampling blocks. The comparisons between our protocols, the SW PoR [27] and the privacy-preserving PDP of Wang et al. [30] are given in Figs. 4 and 5.

Detection Rate. Assume the CSP hosts a n-block file, out of which x blocks are corrupted. The verifier checks the integrity of the entire file containing n blocks by randomly sampling c different blocks. In the experiment, we show detection probabilities for 1, 5, 10 and 20 percent corruption rates, respectively. Let Px denote the detection rate

$$Px = 1 - \frac{(n-x)(n-1-x)\cdots(n-c+1-x)}{n(n-1)\cdots(n-c+1)}.$$

When the corruption rate is of 1 percent, generating a proof with 400 blocks from a file with 500 blocks (Fig. 6) can detect such corruption with 99.9 percent probability. The probability will be declined to 99.4 percent when such file contains 5000 blocks (Fig. 8).

When the corruption rate is of 5 percent, generating a proof with 380 blocks from a 500-block file provides the detection rate close to 1. When the file contains 1000 blocks (Fig. 7), the detection rate of sampling 100 blocks is close to

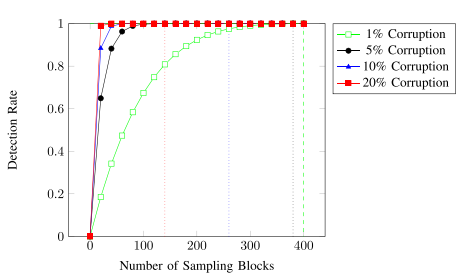


Fig. 6. Detection rate on 500-block file.

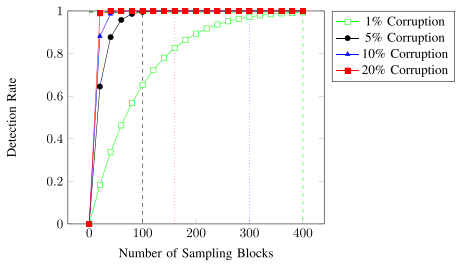


Fig. 7. Detection rate on 1000-block file.

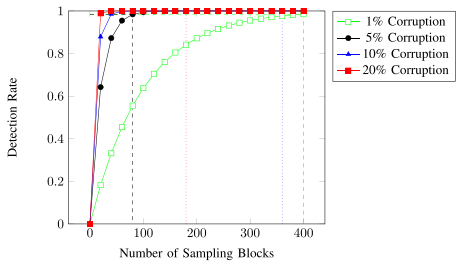


Fig. 8. Detection rate on 5000-block file.

99.4 percent. When sampling 80 blocks from a 5000-block file, the detection rate is 98.4 percent.

When the corruption rate is of 10 percent, to achieve the detection rate close to 1, the proof should contain 260 blocks, 300 blocks and 360 blocks on the 500-block file, 1000-block file and 5000-block file, respectively.

When the corruption rate is of 20 percent, to achieve the detection rate close to 1, the proof should contain 140 blocks, 160 blocks and 180 blocks on the 500-block file, 1000-block file and 5000-block file, respectively.

7 Conclusion

In this paper, we introduced a new Proof of Storage paradigm that supports the pay-as-you-go business model. We proposed two protocols, one allows data retrievability by the owner while the other supports public auditing by a third party auditor. Both protocols allow a verifier to check the integrity of the data content and the associated timestamp simultaneously. We proved that the two protocols are both sound and the protocol supporting third party auditing is privacy-preserving with regards to the data content and the timestamp. The performance analysis and experiment also demonstrate that our proposed protocols are efficient and practical.

Appendix A
Proof of Lemma 2

Before analyzing the difference in success probabilities between Game 1 and Game 2, we clarify some notation in order to present a summary. Supposed that $\mathcal{A}$ provides a proof $\pi' = (\zeta', \{\mu'_j\}, d, R, RI)$ of which $\zeta' \neq \zeta = \prod_{(\theta_i,\nu_i)\in Q} \sigma_{\theta_i}^{\frac{1}{\nu_i}}$ that $\{(\theta_i, \nu_i)\}$ are the challenge with c elements. By the correctness of the verification, we gain an equation that

$$e\left(\zeta', \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right),$$

where $\mathcal{N}$ denotes $\prod_{k=1}^{c} \nu_k$ and $\mathcal{N}_i$ denotes $\prod_{k=1,k\neq i}^{c} \nu_k$.

We show the security of our protocol can be reduced to the q-SDH problem in the AGM with random oracle. In the AGM, a forgery comes with a representation in the basis of all responses to queries.

Proof. We show that if there is a non-negligible difference in success probabilities between Game 1 and Game 2, we can construct a simulator $\mathcal{B}$ that can solve the q-SDH problem with non-negligible probability.

$\mathcal{B}$ is given a q-SDH problem instance $(g_1, g_1^x, g_1^{x^2}, \cdots, g_1^{x^q}, g_2, g_2^x) \in \mathbb{G}_1^{q+1} \times \mathbb{G}_2^2$. Its goal is to compute $(c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}_1$. There is an algebraic adversary $\mathcal{A}$ plays the game with $\mathcal{B}$ when $\mathcal{A}$ is allowed to query hash values and group elements via oracles. $\mathcal{B}$ plays a trick on $H(\cdot)$ to ensure that it can respond tag queries from adversary $\mathcal{A}$ successfully.

Setup. $\mathcal{B}$ sets g_1, g_2 as generators of $\mathbb{G}_1$ and $\mathbb{G}_2$. It sets $U = g_2^x$, then sends the public key to $\mathcal{A}$.

Tag Query. $\mathcal{A}$ is allowed to query for $q_1, \cdots, q_m$. For each query q_i, it runs as follows.

$\mathcal{A}$ sends a file F with the timestamp t as the input. $\mathcal{B}$ receives (F, t) and separates the encoded file F^* into n blocks. Then, for each block, it separates the block into s sectors, that the m_{ij} denotes the jth sector of the block m_i. It sets $u_j = g_1^{\gamma_j}$ $(1 \leq j \leq s, \gamma_j \in \mathbb{Z}_p)$ and randomly chooses $r_i \in \mathbb{Z}_p$ to compute the hash value for ith block

$$H(fname||i) = \frac{g_1^{r_i(f(x)+f(\hat{t}))}}{g_1^{\sum_{j=1}^{s} \gamma_j m_{ij}}}.$$

Here $f(\cdot)$ is a q-degree polynomial. $fname$ is a random string to identify files. When choosing a $fname$ for a given file, it must be queried for the first time.

$\mathcal{B}$ gains the hash value of each block, then computes the corresponding tag and handles. For each block m_i, $\mathcal{B}$ computes

$$\sigma_i = \left(H(fname||i) \prod_{j=1}^{s} u_j^{m_{ij}}\right)^{\frac{1}{x+\hat{t}}} = g_1^{r_i F(x)},$$

where $F(x) = \frac{f(x)+f(\hat{t})}{x+\hat{t}}$ is a (q-1)-degree polynomial.

Proof-Verify. $\mathcal{B}$ interacts with $\mathcal{A}$, if $\mathcal{A}$ submits a proof $\pi' = (\zeta', \{\mu'_j\}, d, R, RI)$ which is accepted by the verification algorithm that is different from the expected proof. $\mathcal{B}$ declares failure and aborts.

Game 1 guarantees that the parameters associated with this protocol instance $(fname, n, \{u_j\}, \{m_{ij}\}, \{\sigma_i\})$ is generated by $\mathcal{B}$; otherwise, execution would have already aborted. $\mathcal{A}$ gets the group elements for a response

$$R = g_2^r g_2^{xl} \wedge RI = g_1^{\frac{1}{r+xl}},$$
$$\zeta' = \prod_{k=0}^{q_G} g_1^{x^k r_k} \prod_{p=0}^{q_h} H_p^{a_p} \prod_{e=0}^{q_t} \sigma_e^{b_e},$$

where q_h, q_G and q_t are the number of responses that $\mathcal{A}$ gains from the hash oracle, group oracle and tags. Let H_p denote the pth response from the hash oracle. $r, l, \{a_p\}$

and $\{b_e\}$ compose representations of R, RI and ζ'. If $l \neq 0$, $(r/l, RI^{\frac{1}{l}})$ is the answer to the q-SDH problem. Otherwise, as we argued before, by the correctness of the verification, any response of $\mathcal{A}$ satisfies the equation that

$$e\left(\zeta', \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right) = e\left(\prod_{i=1}^{c} \frac{g_1^{\mathcal{N}_i r_{\theta_i}(f(x)+f(\hat{t}))}}{g_1^{\mathcal{N}_i \sum_{j=1}^{s} \gamma_j m_{\theta_i j}}} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right).$$

Therefore, we obtain that

$$e(\zeta'^{\mathcal{N}}, g_2) = e\left(g_1^{\frac{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i}(f(x)+f(\hat{t}))+\sum_{j=1}^{s} \gamma_j(\mu'_j - \sum_{i=1}^{c} \mathcal{N}_i m_{\theta_i j})}{x+\frac{d-r}{\mathcal{N}}}}, g_2\right).$$

Let $\sum_{j=1}^{s} \gamma_j(\mu'_j - \sum_{i=1}^{c} \mathcal{N}_i m_{\theta_i j})$ be A.

$$\zeta'^{\mathcal{N}} = g_1^{\frac{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i}(f(x)+f(\hat{t}))+A}{x+\frac{d-r}{\mathcal{N}}}} = g_1^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} F(x) + \frac{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i}(f(\hat{t}) - f(\frac{d-r}{\mathcal{N}}))+A}{x+\frac{d-r}{\mathcal{N}}}}.$$

It is concluded that

$$g_1^{\frac{1}{x+\frac{d-r}{\mathcal{N}}}} = \left(\frac{\zeta'^{\mathcal{N}}}{g_1^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} F(x)}}\right)^{\frac{1}{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i}(f(\hat{t}) - f(\frac{d-r}{\mathcal{N}}))+A}}.$$

Therefore, $\mathcal{B}$ obtains the answer to the given q-SDH problem that

$$\left(\frac{d-r}{\mathcal{N}}, \left(\frac{\zeta'^{\mathcal{N}}}{g_1^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} F(x)}}\right)^{\frac{1}{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i}(f(\hat{t}) - f(\frac{d-r}{\mathcal{N}}))+A}}\right).$$

Let E_2 denote the event that the adversary queries a proof on the file which $\zeta' \neq \zeta$. Hence, we have that $|\mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game2}} - \mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game1}}| \leq \Pr[\mathrm{E}_2] = \epsilon_{q-\mathrm{SDH}}$. □

APPENDIX B
PROOF OF LEMMA 3

Game 2 guarantees that we have ζ' is equal to the expected result ζ. The only difference in success probabilities is caused by $\{\mu'_j\}$. $\mathcal{A}$ submits a proof $\pi' = (\zeta, \{\mu'_j\}, d, R, RI)$ which is accepted by the verification algorithm. By the correctness of the verification algorithm, the equation is as follows

$$e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right) = e\left(\zeta', \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right),$$

where $\mathcal{N}$ denotes $\prod_{k=1}^{c} \nu_k$ and $\mathcal{N}_i$ denotes $\prod_{k=1, k\neq i}^{c} \nu_k$. $\{(\theta_i, \nu_i)\}$ are the challenge with c elements from the verifier.

We prove the difference in success probabilities between Game 2 and Game 3 is negligible if the DP assumption holds in bilinear groups.

Proof. $\mathcal{B}$ is given a DP problem instance $(g_r, g_t) \in \mathbb{G}_1^2$. Its goal is to find (R, T) that $e(g_r, R)e(g_t, T) = 1$ The only difference between Game 2 and Game 3 is to replace g_1 with g_r and u_j with $g_r^{\gamma_j} g_t^{\iota_j}$. It randomly chooses $x \in \mathbb{Z}_p$ as the part of private key and $U = g_2^x$ as a part of the public key. Then, it sends U to $\mathcal{A}$. By the correctness of our verification algorithm, $\mathcal{B}$ obtains that

$$e\left(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right).$$

It follows that

$$\begin{aligned}
1 =& e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} \prod_{j=1}^{s} u_j^{\mu'_j}, g_2\right) e\left(\zeta^{-1}, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) \\
=& e\left(g_r^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} - \sum_{j=1}^{s} \gamma_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)}, g_2\right) e\left(g_t^{\sum_{j=1}^{s} \iota_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)}, g_2\right) \\
& e\left(g_r^{-\sum_{i=1}^{c} \frac{r_{\theta_i}}{\nu_i(x+\hat{t})}}, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) \\
=& e\left(g_t, g_2^{\sum_{j=1}^{s} \iota_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)}\right) e\left(g_r, \left(\frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right)^{-\sum_{i=1}^{c} \frac{r_{\theta_i}}{\nu_i(x+\hat{t})}}\right) \\
& e\left(g_r, g_2^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} - \sum_{j=1}^{s} \gamma_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)}\right).
\end{aligned}$$

Let

$$R_1 = \left(\frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right)^{-\sum_{i=1}^{c} \frac{r_{\theta_i}}{\nu_i(x+\hat{t})}}, \quad R_2 = g_2^{\sum_{i=1}^{c} \mathcal{N}_i r_{\theta_i} - \sum_{j=1}^{s} \gamma_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)},$$

hence, the answer to the given DP problem instance is

$$\left(R_1 \cdot R_2, g_2^{\sum_{j=1}^{s} \iota_j \left(\sum_{i=1}^{c} m_{\theta_i j} - \mu'_j\right)}\right).$$

Let E_3 denote the event that the adversary queries a proof on the file which at least one $\mu'_j \neq \mu_j$. Hence, we have that $|\mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game3}} - \mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game2}}| \leq \Pr[\mathrm{E}_3] = \epsilon_{dp}$. □

APPENDIX C
SOUNDNESS AGAINST CSP

We define Game 0, Game 1 and Game 2 played between the simulator $\mathcal{B}$ and the adversary $\mathcal{A}$ to prove that our PoS under PDP model is sound against malicious CSP. Before playing games, $\mathcal{B}$ initial the environment by running Setup.

Game 0. $\mathcal{A}$ interacts with $\mathcal{B}$ in the challenged game as defined in the soundness game.

Game 1. It is the same as Game 0 expect that If $\mathcal{A}$ submits a file tag that is valid but not generated by $\mathcal{B}$, $\mathcal{B}$ declares failure and aborts.

The difference in success probabilities between Game 0 and Game 1 is negligible if the signature scheme is existentially unforgeable.

Game 2. It is the same as Game 1, except that if $\mathcal{A}$ makes VerifyProof outputs 1 in any proof instance but the aggregated value ζ' is not equal to $\prod_{(\theta_i,\nu_i)\in Q}\sigma_{\theta_i}^{\frac{1}{\nu_i}}$, where Q is the challenge issued by $\mathcal{B}$, $\mathcal{B}$ declares failure and aborts.

Game 3. As in Game 2, the challenger tracks Tag Queries and observes proof of storage protocol instances. This time, if in any of these instances the adversary is successful to make VerifyProof accept but at least one μ is not generated honestly, $\mathcal{B}$ declares failure and aborts.

Lemma 4. *The difference in success probabilities between Game 1 and Game 2 is negligible, if the q-SDH assumption holds in bilinear groups in AGM with a random oracle.*

Proof. Supposed that $\mathcal{A}$ provides a proof $\pi' = (\zeta', A, AI, \mu, R, RI, d)$ of which $\zeta' \neq \prod_{(\theta_i,\nu_i)\in Q}\sigma_{\theta_i}^{\frac{1}{\nu_i}}$. By the correctness of the verification, we have the following equations

$$e(AI, A) = e(g_1, g_2) \wedge e(RI, R) = e(g_1, g_2) \wedge$$
$$e(u, A)\cdot e\left(\zeta', \frac{g_2^d}{R}\cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i}u^{\mu}, g_2\right),$$

where $\mathcal{N}$ denotes $\prod_{k=1}^{c}\nu_k$ and $\mathcal{N}_i$ denotes $\prod_{k=1,k\neq i}^{c}\nu_k$. $\{(\theta_i,\nu_i)\}$ are the challenge with c elements from the verifier.

We show that if there is a non-negligible difference in success probabilities between Game 1 and Game 2, we can construct a simulator $\mathcal{B}$ that can solve the q-SDH problem with non-negligible probability.

$\mathcal{B}$ is given a q-SDH problem instance $(g_1, g_1^x, g_1^{x^2}, \cdots, g_1^{x^q}, g_2, g_2^x)$. Its goal is to compute $(c, g_1^{\frac{1}{x+c}}) \in \mathbb{Z}_p \times \mathbb{G}_1$. An algebraic adversary $\mathcal{A}$ plays the game with $\mathcal{B}$ when $\mathcal{A}$ is allowed to query hash values and group elements via oracles. $\mathcal{B}$ plays a trick on $H(\cdot)$ to ensure that it can respond tag queries from adversary $\mathcal{A}$ successfully.

Setup. $\mathcal{B}$ sets g_1, g_2 as the generators in this game and public key $U = g_2^x$. Then it returns the public key to $\mathcal{A}$.

Tag Query. $\mathcal{A}$ is allowed to query for $q_1, \cdots, q_m$. For each query q_i, it runs as follows.

$\mathcal{A}$ sends the file with the timestamp (F, t) as the input. $\mathcal{B}$ receives (F, t) and separates the encoded file F^* into n blocks. Then, for each block m_i, it randomly chooses $r_i \in \mathbb{Z}_p$ to computes the hash value for the ith block

$$H(fname||i) = \frac{g_1^{r_i(f(x)+f(\hat{t}))}}{g_1^{\gamma m_i}},$$

where $u = g_1^{\gamma}, \gamma \in \mathbb{Z}_p$, $f(\cdot)$ is a q-degree polynomial. $fname$ is a random string to identify a file.

$\mathcal{B}$ gains the hash value of each block, then computes the corresponding tag and handles. For each block m_i, $\mathcal{B}$ computes

$$\sigma_i = (H(fname||i)u^{m_i})^{\frac{1}{x+t}} = g_1^{r_i F(x)},$$

where $F(x) = \frac{f(x)+f(\hat{t})}{x+t}$ is a (q-1)-degree polynomial.

Proof-Verify. Game 1 guarantees that the parameters associated with this protocol instance $(fname, n, u, m_i, \{\sigma_i\})$ is generated by $\mathcal{B}$; otherwise, execution would have already aborted. $\mathcal{A}$ responds a proof that,

$$A = g_2^a g_2^{xl_1}, \quad AI = g_1^{\frac{1}{a+xl_1}},$$
$$R = g_2^r g_2^{xl_2}, \quad RI = g_1^{\frac{1}{r+xl_2}},$$
$$\zeta' = \prod_{k=0}^{q_G} g_1^{x^k r_k}\prod_{p=0}^{q_h} H_p^{a_p}\prod_{e=0}^{q_t}\sigma_e^{b_e}.$$

q_h, q_G and q_t are the number of responses that $\mathcal{A}$ gains from the hash oracle, group oracle and tag query. Let H_p denote the pth response from the hash oracle. $a, l_1, r, l_2, \{a_p\}$ and $\{b_e\}$ compose representations of A, AI, R, RI and ζ'. Either l_1 or l_2 is not equal to 0, $(a/l_1, AI^{\frac{1}{l_1}})$ or $(r/l_2, RI^{\frac{1}{l_2}})$ is an answers to the given q-SDH problem instance. Otherwise, as we argued before, by the correctness of the verification, any response of $\mathcal{A}$ satisfy the equation that

$$e(u, A)\cdot e\left(\zeta', \frac{g_2^d}{R}\cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i}u^{\mu}, g_2\right).$$

Then $\mathcal{B}$ is able to compute

$$e\left(\zeta', g_2^{d-r+x\mathcal{N}}\right) = e\left(g_1^{\sum_{i=1}^{c}\mathcal{N}_i(r_{\theta_i}(f(x)+f(\hat{t}))-\gamma m_{\theta_i})+\gamma(\mu'-a)}, g_2\right).$$

Hence, the equation shows that $e(\zeta'^{\mathcal{N}}, g_2)$ is equal to

$$e\left(g_1^{\frac{\sum_{i=1}^{c}\mathcal{N}_i(r_{\theta_i}(f(x)+f(\hat{t}))-\gamma m_{\theta_i})+\gamma(\mu'-a)}{x+(d-r)/\mathcal{N}}}, g_2\right).$$

Let $\gamma(\mu' - a\sum_{i=1}^{c}\mathcal{N}_i m_{\theta_i})$ be A.

$$\begin{aligned}\zeta'^{\mathcal{N}} &= g_1^{\frac{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}(f(x)+f(\hat{t}))+A}{x+\frac{d-r}{\mathcal{N}}}}\\ &= g_1^{\frac{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}\left(f(x)+f\left(\frac{d-r}{\mathcal{N}}\right)\right)+\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}\left(f(\hat{t})-f\left(\frac{d-r}{\mathcal{N}}\right)\right)+A}{x+\frac{d-r}{\mathcal{N}}}}\\ &= g_1^{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}F(x)+\frac{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}\left(f(\hat{t})-f\left(\frac{d-r}{\mathcal{N}}\right)\right)+A}{x+\frac{d-r}{\mathcal{N}}}}.\end{aligned}$$

Hence, we obtain that

$$g_1^{\frac{1}{x+\frac{d-r}{\mathcal{N}}}} = \left(\frac{\zeta'^{\mathcal{N}}}{g_1^{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}F(x)}}\right)^{\frac{1}{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}\left(f(\hat{t})-f\left(\frac{d-r}{\mathcal{N}}\right)\right)+A}}.$$

Therefore, $\mathcal{B}$ obtains the answer to the given q-SDH problem

$$\left(\frac{d-r}{\mathcal{N}}, \left(\frac{\zeta'^{\mathcal{N}}}{g_1^{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}F(x)}}\right)^{\frac{1}{\sum_{i=1}^{c}\mathcal{N}_i r_{\theta_i}\left(f(\hat{t})-f\left(\frac{d-r}{\mathcal{N}}\right)\right)+A}}\right).$$

Hence, we have that $|\mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game2}} - \mathrm{Adv}_{\mathcal{A}}^{\mathrm{Game1}}| \leq \epsilon_{q-\mathrm{SDH}}$. □

Lemma 5. *The difference in success probabilities between Game 2 and Game 3 is negligible, if the DP assumption holds in bilinear groups.*

Game 2 guarantees that we have ζ' is equal to the expected result ζ. The only difference in success probabilities is caused by μ'. $\mathcal{A}$ submits a proof $\pi' = (\zeta, \mu', d, R, RI)$ which is accepted by the verification algorithm. By the verification equation

$$e(u, A) \cdot e\left(\zeta, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu'}, g_2\right),$$

we can obtain an equation that is

$$1 = e\left(\zeta^{-1}, \frac{g_2^d}{R} \cdot U^{\mathcal{N}}\right) e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i}, g_2\right) e\left(u, \frac{g_2^{\mu'}}{A}\right).$$

The proof for Lemma 5 is similar to the proof for Lemma 3. Hence, we have that $|\mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game3}} - \mathrm{A}dv_{\mathcal{A}}^{\mathrm{Game2}}| \leq \Pr[\mathrm{E}_3] = \epsilon_{dp}$.

Appendix D
Content/Time Privacy against TPA

Proof. We prove that the TPA cannot learn any information of the content or timestamp from the CSP's response.

In our proof, we show that there does not exist any PPT adversary who can recovery the content or timestamp if the signature scheme Sig is secure and the extended-DL assumption holds.

Let $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ denote an isomorphism. The aggregated value of the content $\sum_c^{i=1} \mathcal{N}_i m_{\theta_i}$ is represented as M, where $\mathcal{N}_i$ denotes $\prod_{k=1,k\neq i}^{c} \nu_k$. $\{(\theta_i, \nu_i)\}$ are the challenge with c elements from the verifier.

Case 1. $\mathcal{A}$ has the knowledge of timestamp $\hat{t}'$. We prove that our protocol will not reveal the content under the extended-DL assumption.

In this case, we show the simulator can produce a valid response even without the knowledge of the content in random oracle model. Now, $\mathcal{A}$ is treated as a verifier. Given a challenge from $\mathcal{A}$, $\mathcal{S}$ sets that $A = g_2^{zx}$ and $AI = g_1^{\frac{1}{zx}}, z \in \mathbb{Z}_p$. Then, it randomly picks μ, r from $\mathbb{Z}_p$. d is computed with $\hat{t}$ and r that $d = r + \mathcal{N}\hat{t}$. It sets $u = g_1^{r_u}$ and $U = g_2^{x_{sk}}$, where $r_u, x_{sk} \in \mathbb{Z}_p$.$\mathcal{S}$ sets the value of ζ via the correctness equation

$$e(u, A) \cdot e\left(\zeta, g_2^{d-r+x_{sk}\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu}, g_2\right),$$

which gives

$$e\left(\zeta, g_2^{d-r+x_{sk}\mathcal{N}}\right) = e\left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu} \psi(A)^{-r_u}, g_2\right),$$

where $\mathcal{N} = \prod_{k=1}^{c} \nu_k$.

Hence, $\zeta = \left(\prod_{i=1}^{c} H(fname||\theta_i)^{\mathcal{N}_i} u^{\mu} \psi(A)^{-r_u}\right)^{\frac{1}{d-r+x_{sk}\mathcal{N}}}$. Therefore, if $\mathcal{A}$ can recover the content, $\mathcal{S}$ can compute $x = \frac{\mu - M}{z}$ in this condition.

Case 2. $\mathcal{A}$ has the knowledge of the content. We prove that the protocol will not reveal the timestamp due to the security of the signature scheme Sig.

In this case, we prove that the TPA cannot recover the timestamp with the probability more than $\frac{1}{\#T} + \epsilon_{\mathsf{Sig}}$. We prove the time privacy via the following games.

Game 0. The game is the challenged game.

Game 1. The difference between Game 1 and Game 0 is that if $\mathcal{A}$ sends any hash query to H_0 where the hash input $\mathsf{Sig}_{\mathsf{ssk}}(t)$ is not generated by $\mathcal{S}$ before, the game will be terminated. It means $\mathcal{A}$ can break the unforgeability of the signature scheme Sig. Thus, $|\Pr[\mathrm{G}ame\ 1] - \Pr[\mathrm{G}ame\ 0]| = \epsilon_{\mathsf{Sig}}$.

Game 2. $\mathcal{S}$ choose $\hat{t}$ from $\mathbb{Z}_p$. The probability is the same for Game 1 and Game 2 (when H_0 is modeled as a random oracle). The probability of adversary in Game 2 is $\frac{1}{\#T}$ since the simulation is not related to the timestamp t, so the chance the adversary can obtain t is $\frac{1}{\#T}$. The probability overall is no more than $\frac{1}{\#T} + \epsilon_{\mathsf{Sig}}$. This completes the proof. □

References

[1] N. Alon and M. Luby, "A linear time erasure-resilient code with nearly optimal recovery," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1732–1736, Nov. 1996.

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

[3] G. Ateniese, S. Kamara, and J. Katz, "Proofs of storage from homomorphic identification protocols," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2009, pp. 319–333.

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proc. 4th Int. Conf. Secur. Privacy Commun. Netw.*, 2008, Art. no. 9.

[5] P. S. L. M. Barreto and M. Naehrig, "Pairing-friendly elliptic curves of prime order," in *Proc. 12th Int. Workshop Select. Areas Cryptography*, 2005, pp. 319–331.

[6] A. F. Barsoum and M. A. Hasan, "Provable multicopy dynamic data possession in cloud computing systems," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 3, pp. 485–497, Mar. 2015.

[7] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, 2007, pp. 321–334.

[8] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the Weil pairing," in *Proc. 7th Int. Conf. Theory Appl. Cryptology Inf. Secur.: Adv. Cryptology*, 2001, pp. 514–532.

[9] D. Cash, A. Küpçü, and D. Wichs, "Dynamic proofs of retrievability via oblivious RAM," in *Proc. Annu. Int. Conf. Theory Appl. Cryptographic Tech.*, 2013, pp. 279–295.

[10] B. Chen, R. Curtmola, G. Ateniese, and R. C. Burns, "Remote data checking for network coding-based distributed storage systems," in *Proc. 2nd ACM Cloud Comput. Secur. Workshop*, 2010, pp. 31–42.

[11] H. C. H. Chen and P. P. C. Lee, "Enabling data integrity protection in regenerating-coding-based cloud storage: Theory and implementation," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 2, pp. 407–416, Feb. 2014.

[12] R. Chen, Y. Mu, G. Yang, and F. Guo, "BL-MLE: Block-level message-locked encryption for secure large file deduplication," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 12, pp. 2643–2652, Dec. 2015.

[13] C. Xu, X. He, and D. Abraha, "Cryptanalysis of auditing protocol proposed by Wang et al. for data storage security in cloud computing," *IACR Cryptology ePrint Archive*, vol. 2012, 2012, Art. no. 115.

[14] Y. Deswarte, J. Quisquater, and A. Saïdane, "Remote integrity checking - how to trust files stored on untrusted servers," in *Proc. Working Conf. Integrity Internal Control Inf. Syst.*, 2003, pp. 1–11.

[15] C. C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.

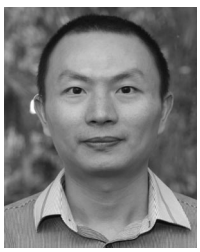
[16] X. Fan, G. Yang, Y. Mu, and Y. Yu, "On indistinguishability in remote data integrity checking," *Comput. J.*, vol. 58, no. 4, pp. 823–830, 2015.

[17] J. Feng, Y. Chen, W. Ku, and P. Liu, "Analysis of integrity vulnerabilities and a non-repudiation protocol for cloud data storage platforms," in *Proc. 39th Int. Conf. Parallel Process. Workshops*, 2010, pp. 251–258.
[18] B. Grobauer, T. Walloschek, and E. Stöcker, "Understanding cloud computing vulnerabilities," *IEEE Secur. Privacy*, vol. 9, no. 2, pp. 50–57, Mar./Apr. 2011.
[19] J. Groth, "Homomorphic trapdoor commitments to group elements," *IACR Cryptology ePrint Archive*, vol. 2009, 2009, Art. no. 7.
[20] T. Jager and A. Rupp, "The semi-generic group model and applications to pairing-based cryptography," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2010, pp. 539–556.
[21] A. Juels and B. S. K. Jr., "Pors: proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.
[22] E. Kiltz and J. Loss, "The algebraic group model and its applications," *IACR Cryptology ePrint Archive*, vol. 2017, 2017, Art. no. 620.
[23] J. Li, L. Zhang, J. K. Liu, H. Qian, and Z. Dong, "Privacy-preserving public auditing protocol for low-performance end devices in cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 11, pp. 2572–2583, Nov. 2016.
[24] H. Liu, L. Chen, Z. Davar, and M. R. Pour, "Insecurity of an efficient privacy-preserving public auditing scheme for cloud data storage," *J. Universal Comput. Sci.*, vol. 21, no. 3, pp. 473–482, 2015.
[25] J. Liu, K. Huang, H. Rong, H. Wang, and M. Xian, "Privacy-preserving public auditing for regenerating-code-based cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 7, pp. 1513–1528, Jul. 2015.
[26] L. Rizzo, "Effective erasure codes for reliable computer communication protocols," *Comput. Commun. Rev.*, vol. 27, no. 2, pp. 24–36, 1997.
[27] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur.*, 2008, pp. 90–107.
[28] E. Shi, E. Stefanov, and C. Papamanthou, "Practical dynamic proofs of retrievability," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2013, pp. 325–336.
[29] Y. Tian, S. Zhang, G. Yang, Y. Mu, and Y. Yu, "Privacy-preserving k-time authenticated secret handshakes," in *Proc. 22nd Australasian Conf. Inf. Secur. Privacy.*, 2017, pp. 281–300.
[30] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[31] 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. 525–533.
[32] H. Wang, D. He, and S. Tang, "Identity-based proxy-oriented data uploading and remote data integrity checking in public cloud," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1165–1176, Jun. 2016.
[33] 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. 14th Eur. Conf. Res. Comput. Secur.*, 2009, pp. 355–370.
[34] Y. Wang, Q. Wu, B. Qin, W. Shi, R. H. Deng, and J. Hu, "Identity-based data outsourcing with comprehensive auditing in clouds," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 940–952, Apr. 2017.
[35] Y. Wang, Q. Wu, B. Qin, S. Tang, and W. Susilo, "Online/offline provable data possession," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 5, pp. 1182–1194, May 2017.
[36] S. G. Worku, C. Xu, J. Zhao, and X. He, "Secure and efficient privacy-preserving public auditing scheme for cloud storage," *Comput. Elect. Eng.*, vol. 40, no. 5, pp. 1703–1713, 2014.
[37] J. Yu, K. Ren, and C. Wang, "Enabling cloud storage auditing with verifiable outsourcing of key updates," *IEEE Trans. Inf. Forensics Secur.*, vol. 11, no. 6, pp. 1362–1375, Jun. 2016.
[38] J. Yu, K. Ren, C. Wang, and V. Varadharajan, "Enabling cloud storage auditing with key-exposure resistance," *IEEE Trans. Inf. Forensics Secur.*, vol. 10, no. 6, pp. 1167–1179, Jun. 2015.
[39] J. Yu and H. Wang, "Strong key-exposure resilient auditing for secure cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 8, pp. 1931–1940, Aug. 2017.
[40] Y. Yu, M. H. Au, G. Ateniese, X. Huang, W. Susilo, Y. Dai, and G. Min, "Identity-based remote data integrity checking with perfect data privacy preserving for cloud storage," *IEEE Trans. Inf. Forensics Secur.*, vol. 12, no. 4, pp. 767–778, Apr. 2017.

Tong Wu received the bachelor's degrees from the School of Computer Science and Technology, Beijing Institute Technology, China in 2014. Currently, she is currently working toward the PhD degree in the School of Computing and Information Technology, University of Wollongong, Australia, under the supervision of Dr. Guomin Yang and Dr. Fuchun Guo. Her research interests include applied cryptography, cloud security.

Guomin Yang received the PhD degree in computer science from the City University of Hong Kong, in 2009. He worked as a research scientist with the Temasek Laboratories of the National University of Singapore (NUS) from September 2009 to May 2012. He is currently an associated professor with the School of Computing and Information Technology, University of Wollongong, Australia. His research mainly focuses on applied cryptography and network security. He received the Australian Research Council Discovery Early Career Researcher Award in 2015.

Yi Mu received the PhD degree from the Australian National University, in 1994. He is currently a professor with the College of Mathematics and Informatics, Fujian Normal University, China. Before joining Fujian Normal University in 2018, he was a professor with the School of Computing and Information Technology and the co-director of Centre for Computer and Information Security Research, University of Wollongong, Australia. His research interest includes cryptography and cyber security. He is a senior member of the IEEE.



Fuchun Guo received the BS and MS degrees from Fujian Normal University China, in 2005 and 2008, respectively, and the PhD degree from the University of Wollongong, Australia, in 2013. He is currently a lecturer with the School of Computing and Information Technology, University of Wollongong. He has been awarded the prestigious Australian Research Council DECRA Fellowship (2017-2020). His primary research interests include public-key cryptography; in particular, protocols, encryption and signature schemes, and security proof.

Robert H. Deng is an AXA chair professor of cybersecurity and the director of the Secure Mobile Centre, School of Information Systems, Singapore Management University (SMU). His research interests include data security and privacy, cloud security and internet of things security. He received the Outstanding University Researcher Award from National University of Singapore, Lee Kuan Yew Fellowship for Research Excellence from SMU, and Asia-Pacific Information Security Leadership Achievements Community Service Star from International Information Systems Security Certification Consortium. His professional contributions include an extensive list of positions in several industry and public services advisory boards, editorial boards and conference committees. These include the editorial boards of the *IEEE Security & Privacy Magazine*, the *IEEE Transactions on Dependable and Secure Computing*, the *IEEE Transactions on Information Forensics and Security*, the *Journal of Computer Science and Technology*, and Steering Committee Chair of the ACM Asia Conference on Computer and Communications Security. He is a fellow of the IEEE.