Lightweight Delegatable Proofs of Storage

Jia Xu[1(✉)], Anjia Yang[1,2(✉)], Jianying Zhou[1], and Duncan S. Wong[3]

[1] Infocomm Security Department, Institute for Infocomm Research, Singapore, Singapore
{xuj,jyzhou}@i2r.a-star.edu.sg
[2] City University of Hong Kong, Hong Kong, China
ayang3-c@my.cityu.edu.hk
[3] Hong Kong Applied Science and Technology Research Institute, Hong Kong, China
duncanwong@astri.org

Abstract. Proofs of Storage (including Proofs of Retrievability and Provable Data Possession) is a cryptographic tool, which enables data owner or third party auditor to audit integrity of data stored remotely in a cloud storage server, without keeping a local copy of data or downloading data back during auditing. We observe that all existing publicly verifiable POS schemes suffer from a serious drawback: It is extremely slow to compute authentication tags for all data blocks, due to many expensive group exponentiation operations. Surprisingly, it is even much slower than typical network uploading speed, and becomes the bottleneck of the setup phase of the POS scheme. We propose a new variant formulation called "Delegatable Proofs of Storage". In this new relaxed formulation, we are able to construct a POS scheme, which on one side is as efficient as privately verifiable POS schemes, and on the other side can support third party auditor and can efficiently switch auditors at any time, close to the functionalities of publicly verifiable POS schemes. Compared to traditional publicly verifiable POS schemes, we speed up the tag generation process by at least several hundred times, without sacrificing efficiency in any other aspect. Like many existing schemes, we can also speed up our tag generation process by approximately N times using N CPU cores in parallel, before I/O cost becomes the bottleneck. We prove that our scheme is sound under Bilinear Strong Diffie-Hellman Assumption in standard model.

Keywords: Proof of Storage · Proof of Retrievability · Third party verifier · Lightweight homomorphic authentication tag · Applied cryptography

(1) The full version [46] with all details of proof is available at http://eprint.iacr.org/2014/395.

(2) A. Yang contributed to this work when he took his internship in Infocomm Security Department, Institute for Infocomm Research, Singapore.

I. Askoxylakis et al. (Eds.): ESORICS 2016, Part I, LNCS 9878, pp. 324–343, 2016.
DOI: 10.1007/978-3-319-45744-4_16

1 Introduction

Since Proofs of Retrievability (POR [22]) and Provable Data Possession (PDP [4]) are proposed in 2007, a lot of effort of research community has been devoted to constructing proofs of storage schemes with more advanced features. The new features include, public key verifiability [30], supporting dynamic operations [10,16,35] (i.e. inserting/deleting/editing a data block), supporting multiple cloud servers [13], privacy-preserving against auditor [42], and supporting data sharing [37], etc.

1.1 Drawbacks of Publicly Verifiable Proofs of Storage

Expensive Setup Preprocessing. We look back into the very first feature—public verifiability, and observe that all existing publicly verifiable POS schemes suffer from serious drawbacks: (1) Merkle Hash Tree based method is not disk IO-efficient and not even a sub-linear memory authenticator [24]: Every bit of the file has to be accessed by the cloud storage server in each remote integrity auditing process. (2) By our knowledge, all other publicly verifiable POS schemes employ a lot of expensive operation (e.g. group exponentiation) to generate authentication tags for data blocks. As a result, it is prohibitively expensive to generate authentication tags for medium or large size data file. For example, Wang *et al.* [38] achieves throughput of data pre-processing (i.e. generating authentication tag) at speed 17.2 KB/s with an Intel Core 2 1.86 GHz workstation CPU, which means it will take about 17 h to generate authentication tags for a 1 GB file. Even if the user has a CPU with 8 cores, it still requires more than 2 h heavy computation. Such amount of heavy computation is not appropriate for a laptop, not to mention tablet computer (e.g. iPad) or smart phone. It might be weird to tell users that, *mobile device should be only used to verify or download data file stored in cloud storage server, and should not be used to upload (and thus pre-process) data file to cloud.* Unless a formal lower bound is proved and shows that existing study of POS has reach optimality, it is the responsibility of our researchers to make both pre-processing and verification of (third party verifiable) POS practically efficient, although the existing works have already reached good amortized complexity. In this paper, we make our effort towards this direction, improving pre-processing speed by several hundred times without sacrificing efficiency on other aspects.

In many publicly verifiable POS (POR/PDP) scheme (e.g. [4,30,38,42]), publicly verifiable authentication tag function, which is a variant of signing algorithm in a digital signature scheme, is applied directly over every block of a large user data. This is one of few application scenarios that a public key cryptography primitive is directly applied over large user data. In contrast, (1) public key encryption scheme is typically employed to encrypt a short symmetric cipher key, and the more efficient symmetric cipher (e.g. AES) will encrypt the user data; (2) digital signature scheme is typically applied over a short hash digest of large user data, where the hash function (e.g. SHA256) is much more efficient (in term of throughput) than digital signature signing algorithm.

Lack of Control on Auditing. The benefit of publicly verifiable POS schemes is that, anyone with the public key can audit the integrity of data in cloud storage, to relieve the burden from the data owner. However, one should not allow any third party to audit his/her data at their will, and delegation of auditing task should be in a controlled and organized manner. Otherwise, we cannot prevent extreme cases: (1) on one hand, some data file could attract too much attention from public, and are audited unnecessarily too frequently by the public, which might actually result in distributed denial of service attack against the cloud storage server; (2) on the other hand, some unpopular data file may be audited by the public too rarely, so that the possible data loss event might be detected and alerted to the data owner too late and no effective countermeasure can be done to reduce the damage at that time.

1.2 Existing Approaches to Mitigate Drawbacks

Outsourcing Expensive Operations. To reduce the computation burden on data owner for preprocessing in setup phase, the data owner could outsource expensive operations (e.g. group exponentiation) to some cloud computing server during authentication tag generation, by using existing techniques (e.g. [12,21]) as black-box, and verify the computation result.

However, this approach just shifts the computation burden from data owner to cloud storage server, instead of reducing the amount of expensive operations. Furthermore, considering the data owner and cloud computing server as a whole system, much more cost in network communication and computation will be incurred: (1) uploading (possibly transformed) data file, to the cloud computing server, and downloading computation results from the cloud computing server; (2) extra computation cost on both data owner side and cloud computing server side, in order to allow data owner to verify the computation result returned by the cloud computing server and maintain data privacy against cloud computing server.

One may argue that it could save much of the above cost, if the outsourcing of expensive operations and proofs of storage scheme are integrated together and letting cloud storage server takes the role of cloud computing server. But in this case, simple black-box combination of existing proofs of storage scheme and existing privacy-preserving and verifiable outsource scheme for expensive operations, may not work. Thus, a new sophisticated proofs of storage scheme is required to be constructed following this approach, which remains an open problem.

Dual Instantiations of Privately Verifiable Proof of Storage. The data owner could independently apply an existing privately verifiable POS scheme over an input file twice, in order to generate two key pairs and two authentication tags per each data block, where one key pair and authentication tag (per data block) will be utilized by data owner to perform data integrity check, and the other key pair and authentication tag (per data block) will be utilized by auditor to perform data integrity check, using the interactive proof algorithm in the

privately verifiable POS scheme. The limitation of this approach is that, in order to add an extra auditor or switch the auditor, the data owner has to download the whole data file to refresh the key pair and authentication tags for auditor.

Recently, [2] gave an alternative solution. The data owner runs privately verifiable POS scheme (i.e. Shacham-Water's scheme [30] as in [2]) over a data file to get a key pair and authentication tag per each data block, and uploads the data file together with newly generated authentication tags to cloud storage server. Next, the auditor downloads the whole file from cloud storage server, and independently runs the same privately verifiable POS scheme over the downloaded file, to get another key pair and another set of authentication tags. The auditor uploads these authentication tags to cloud storage server. For each challenge query provided by the auditor, the cloud storage server will compute two responses, where one is upon data owner's authentication tags and the other is upon auditor's authentication tags. Then the auditor can verify the response generated upon his/her authentication tags, and keeps the other response available for data owner.

Since [2] aims to resolve possible framing attack among the data owner, cloud storage server and auditor, all communication messages are digitally signed by senders, and the auditor has to prove to the data owner that, his/her authentication tags are generated *correctly*, where this proof method is very expensive, and comparable to tag generation complexity of publicly verifiable POS scheme (e.g. [4,30,38,42]). Furthermore, in this scheme, in the case of revoking or adding an auditor, the new auditor has to download the whole file, then compute authentication tags, and prove that these tags are correctly generated to the data owner.

We remark that our early version of this work appeared as a private internal technique report in early 2014, before [2] became available to public.

Program Obfuscation. Very recently, [19] proposed to construct publicly verifiable POR from privately verifiable POR using indistinguishability obfuscation technique [17]. This obfuscation technique is able to embed the data owner's secret key in a verifier program, in a way such that it is hard to recover the secret key from the obfuscated verifier program. Therefore, this obfuscated verifier program could be treated as public key and given to the auditor to perform data integrity check. However, both [17,19] admit that indistinguishability obfuscation is currently impractical. Particularly, [1] implements the scheme of [17] and shows that, it requires about 9 h to obfuscate a simple function which contains just 15 AND gates, and resulted obfuscated program has size 31.1 GB. Furthermore, it requires around 3.3 h to evaluate the obfuscated program on a single input.

1.3 Our Approach

To address the issues of existing publicly verifiable POS schemes, we propose a *hybrid* POS scheme, which on one hand supports delegation of data auditing task and switching/adding/revoking an auditor, like publicly verifiable POS schemes, and on the other hand is as efficient as a privately verifiable POS scheme.

Unlike in publicly verifiable POS scheme, the data owner could delegate the auditing task to some semi-trusted third party auditor, and this auditor is responsible to audit the data stored in cloud storage on behalf of the data owner, in a controlled way and with proper frequency. We call such an exclusive auditor as *Owner-Delegated-Auditor* or ODA for short. In real world applications, ODA could be another server that provides free or paid auditing service to many cloud users.

Our bottom line is that, even if all auditors colluded with the dishonest cloud storage server, our formulation and scheme should guarantee that the data owner still retains the capability to perform POR auditing by herself.

Overview of Our Scheme. Our scheme generates two pairs of public/private keys: (pk, sk) and (vpk, vsk). The verification public/private key pair (vpk, vsk) is delegated to the ODA. Our scheme proposes a novel linear homomorphic authentication tag function [5], which is extremely lightweight, without any expensive operations (e.g. group exponentiation or bilinear map). Our tag function generates two tags (σ_i, t_i) for each data block, where tag σ_i is generated in a way similar to Shacham and Waters' privately verifiable POR scheme [30], and tag t_i is generated in a completely new way. Each of tag σ_i and tag t_i is of length equal to $1/m$-fraction of length of a data block, where the data block is treated as a vector of dimension m[1]. ODA is able to verify data integrity remotely by checking consistency among the data blocks and both tags $\{(\sigma_i, t_i)\}$ that are stored in the cloud storage server, using the verification secret key vsk. The data owner retains the capability to verify data integrity by checking consistency between the data blocks and tags $\{\sigma_i\}$, using the master secret key sk. When an ODA is revoked and replaced by a new ODA, data owner will update all authentication tags $\{t_i\}$ and the verification key pair (vpk, vsk) without downloading the data file from cloud, but keep tags $\{\sigma_i\}$ and master key pair (pk, sk) unchanged.

Furthermore, we customize the polynomial commitment scheme proposed by Kate *et al.* [23] and integrate it into our homomorphic authentication tag scheme, in order to reduce proof size from $O(m)$ to $O(1)$.

1.4 Contributions

Our main contributions can be summarized as below:

- We propose a new formulation called "Delegatable Proofs of Storage" (DPOS), as a relaxed variant of publicly verifiable POS. Our formulation allows data owner to delegate auditing task to a third party auditor, and meanwhile retains the capability to perform audit task by herself, even if the auditor colluded with the cloud storage server. Our formulation also supports revoking and switching auditors efficiently.
- We design a new scheme under this formulation. Our scheme is as efficient as privately verifiable POS: The tag generation throughput is slightly larger

[1] System parameter m can take any positive integer value and typical value is from a hundred to a thousand.

Table 1. Performance Comparison of Proofs of Storage (POR,PDP) Schemes. In this table, publicly verifiable POS schemes appear above our scheme, and privately verifiable POS schemes appear below our scheme.

Scheme	Computation (Data Pre-process)			Communication bits		Storage Overhead (Server)	Computation (Verifier)				Computation (Prover)			
	$exp.$	$mul.$	$add.$	Challenge	Response		$exp.$	$mul.$	$pair.$	$add.$	$exp.$	$mul.$	$pair.$	$add.$
[3, 4]	$2\frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{m\lambda}$	0	$\log \ell + 2^{\kappa}$	2λ	$\frac{\lvert F\rvert}{m}$	ℓ	ℓ	0	0	ℓ	2ℓ	0	ℓ
[30, 31]-pub.	$\frac{\lvert F\rvert}{\lambda} + \frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	$(m+1)\lambda$	$\frac{\lvert F\rvert}{m}$	$\ell + m$	$\ell + m$	2	0	ℓ	$m\ell + \ell$	0	$m\ell$
[43, 44]	$2\frac{\lvert F\rvert}{\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	$(\ell+3)\lambda + \ell(\lceil \log(\frac{\lvert F\rvert}{m\lambda}) \rceil - 1)\ \lvert h\rvert$	$\lvert F\rvert$	ℓ	ℓ	4	0	ℓ	2ℓ	0	ℓ
[42]	$2\frac{\lvert F\rvert}{\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	3λ	$\lvert F\rvert$	ℓ	ℓ	2	0	ℓ	2ℓ	0	ℓ
[38][a]	$\frac{\lvert F\rvert}{\lambda} + \frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	$(2m+1)\lambda$	$\frac{\lvert F\rvert}{m}$	$\ell + m$	$\ell + 2m$	2	0	$\ell + m$	$m\ell + \ell$	1	$m\ell$
[53]	$\frac{\lvert F\rvert}{m\lambda} + m$	$2\frac{\lvert F\rvert}{\lambda} + m$	$\frac{\lvert F\rvert}{\lambda} + m$	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	$(m+3)\lambda$	$\frac{\lvert F\rvert}{m}$	$\ell + m$	$\ell + m$	3	0	$\ell + m$	$m\ell + 2\ell + 2m$	1	$m\ell$
[20]	$\frac{\lvert F\rvert}{m\lambda}$	0	0	$\lambda + k$	λ	0[b]	$\frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{m\lambda}$	0	0	$\log(\frac{\lvert F\rvert}{m\lambda}) + m$	$\frac{\lvert F\rvert}{m\lambda}$	0	$\frac{\lvert F\rvert}{m\lambda}$
[49]	$\frac{\lvert F\rvert}{\lambda} + \frac{\lvert F\rvert}{m\lambda} + m$	$\frac{\lvert F\rvert}{\lambda}$	0	$(\ell+1)\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	2λ	$\frac{\lvert F\rvert}{m}$	ℓ	2ℓ	2	0	$\ell + m$	$m\ell + m + \ell$	m	$m\ell$
[50]	$\frac{\lvert F\rvert}{\lambda} + \frac{2\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$2\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	3λ	$\frac{\lvert F\rvert}{m}$	ℓ	2ℓ	4	0	$\ell + m$	$m\ell + m + \ell$	0	$m\ell$
[34]	$\frac{2\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{m\lambda}$	0	$\frac{\lvert F\rvert}{m} + 2\lambda$	5λ	$\frac{\lvert F\rvert}{m}$	$\frac{\lvert F\rvert}{m\lambda} + 4$	$\frac{\lvert F\rvert}{m\lambda} + 1$	5	0	$\frac{\lvert F\rvert}{m\lambda} + 5$	$3\frac{\lvert F\rvert}{m\lambda}$	3	$2\frac{\lvert F\rvert}{m\lambda}$
Our Scheme	0	$\frac{2\lvert F\rvert}{\lambda}$	$\frac{2\lvert F\rvert}{\lambda}$	$3\lambda + 280$	6λ	$\frac{2\lvert F\rvert}{m}$	6	ℓ	7	ℓ	$3m$	$m\ell + 2\ell + 6m$	0	$m\ell + 2\ell + 2m$
[30, 31]-pri.[c]	0	$\frac{\lvert F\rvert}{\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	$\ell\lambda + \ell \log(\frac{\lvert F\rvert}{m\lambda})$	$(m+1)\lambda$	$\frac{\lvert F\rvert}{m}$	0	$\ell + m$	0	$\ell + m$	0	$m\ell + \ell$	0	$m\ell + \ell$

[a] [38] is a journal version of [42], and the main scheme is almost the same as [42]. We now consider the one that divides each data block into m sectors.
[b] In Hao *et al.*'s paper [20], the authentication tags are stored at both the client and the verifier side, rather than the server side.
[c] The private key verifiable POR scheme of Shacham and Waters [30,31]. Notice that the public key verifiable POS scheme of [30,31] also appears in this table.
κ, k are system parameters, $\lvert h\rvert$ is the length of a hash output. $\lvert F\rvert$ is the data file size. λ is group element size. m is the number of sectors in each data block. ℓ is the sampling size.

than 10 MB/s per CPU core on a mobile CPU released in Year 2008. On the other side, our scheme allows delegation of auditing task to a semi-trusted third party auditor, and also supports switching and revoking an auditor at any time, like a publicly verifiable POS scheme. We compare the performance complexity of our scheme with the state of arts in Table 1, and experiment shows the tag generation speed of our scheme is more than hundred times faster than the state of art of publicly verifiable POS schemes.
- We prove that our scheme is sound (Theorems 1 and 2) under Bilinear Strong Diffie-Hellman Assumption in standard model.

2 Related Work

Recently, much growing attention has been paid to integrity check of data stored at untrusted servers [3–6,8,9,11,13–16,20,22,28–33,36–45,47–53]. In CCS'07, Ateniese *et al.* [4] defined the *provable data possession* (PDP) model and proposed the first publicly verifiable PDP scheme. Their scheme used RSA-based homomorphic authenticators and sampled a number of data blocks rather than the whole data file to audit the outsourced data, which can reduce the communication complexity significantly. However, in their scheme, a linear combination of sampled blocks are exposed to the third party auditor (TPA) at each auditing, which may leak the data information to the TPA. At the meantime, Juels and Kaliski [22] described a similar but stronger model: *proof of retrievability* (POR), which enables auditing of not only the integrity but also the retrievability of remote data files by employing spot-checking and error-correcting codes. Nevertheless, their proposed scheme allows for only a bounded number of auditing services and does not support public verification.

Shacham and Waters [30,31] proposed two POR schemes, where one is private key verifiable and the other is public key verifiable, and gave a rigorous proof of security under the POR model [22]. Similar to [4], their scheme utilized homomorphic authenticators built from BLS signatures [7]. Subsequently, Zeng *et al.* [51], Wang *et al.* [43,44] proposed some similar constructions for publicly verifiable remote data integrity check, which adopted the BLS based homomorphic authenticators. With the same reason as [4], these protocols do not support data privacy. In [38,42], Wang *et al.* extended their scheme to be privacy preserving. The idea is to mask the linear combination of sampled blocks in the server's response with some random value. With the similar masking technique, Zhu *et al.* [53] introduced another privacy-preserving public auditing scheme. Later, Hao *et al.* [20] and Yang *et al.* [49] proposed two privacy-preserving public auditing schemes without applying the masking technique. Yuan *et al.* [50] gave a POR scheme with public verifiability and constant communication cost. Ren [26] designed mutual verifiable public POS application.

However, by our knowledge, all of the publicly verifiable PDP/POR protocols require to do a large amount of computation of exponentiation on big numbers for generating the authentication tags upon preprocessing the data file. This makes these schemes impractical for file of medium or large size, especially limiting the usage on mobile devices.

Although delegable POS has been studied by [25,27,34], unfortunately these works have the same drawback with public POS, i.e., the cost of tag generation is extremely high.

3 Formulation

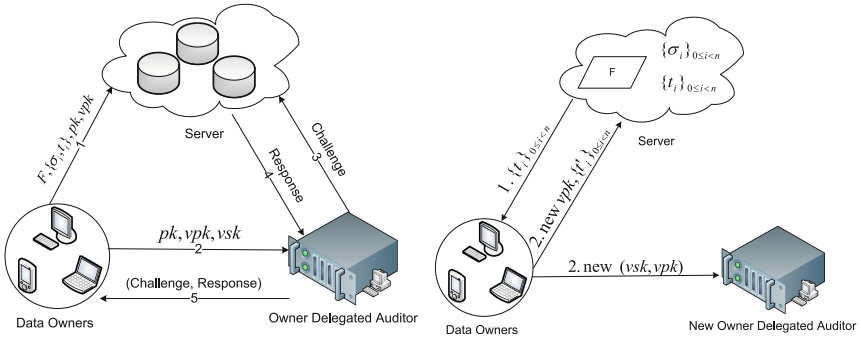
We propose a formulation called "Delegatable Proofs of Storage" scheme (DPOS for short), based on existing POR [22,30] and PDP [4] formulations. We provide the system model in Sect. 3.1 and the trust model in Sect. 3.2. We will defer the security definition to Sect. 5, where the security analysis of our scheme will be provided.

3.1 System Model

Definition 1. *A* Delegatable Proofs of Storage *(*DPOS*) scheme consists of algorithms (*KeyGen, Tag, UpdVK, OwnerVerify*), and a pair of interactive algorithms* $\langle \mathsf{P}, \mathsf{V} \rangle$*, where each algorithm is described as below*

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk, vpk, vsk)$: *Given a security parameter* 1^λ*, this randomized key generating algorithm generates a pair of public/private master keys* (pk, sk) *and a pair of public/private verification keys* (vpk, vsk)*.*
- $\mathsf{Tag}(sk, vsk, F) \rightarrow (\mathtt{Param}_F, \{(\sigma_i, t_i)\})$: *Given the master secret key* sk*, the verification secret key* vsk*, and a data file* F *as input, the tag algorithm generates a file parameter* $\mathtt{Param}_F$ *and authentication tags* $\{(\sigma_i, t_i)\}$*, where a unique file identifier* $\mathtt{id}_F$ *is a part of* $\mathtt{Param}_F$*.*
- $\mathsf{UpdVK}(vpk, vsk, \{t_i\}) \rightarrow (vpk', vsk', \{t'_i\})$: *Given the current verification key pair* (vpk, vsk) *and the current authentication tags* $\{t_i\}$*, this updating algorithm generates the new verification key pair* (vpk', vsk') *and the new authentication tags* $\{t'_i\}$*.*
- $\langle \mathsf{P}(pk, vpk, \{(\vec{\boldsymbol{F}}_i, \sigma_i, t_i)\}_i), \mathsf{V}(vsk, vpk, pk, \mathtt{Param}_F) \rangle \rightarrow (b, \mathtt{Context}, \mathtt{Evidence})$*: The verifier algorithm* V *interacts with the prover algorithm* P *to output a decision bit* $b \in \{1, 0\}$*,* Context *and* Evidence*, where the input of* P *consists of the master public key* pk*, the verification public key* vpk*, and file blocks* $\{\vec{\boldsymbol{F}}_i\}$ *and authentication tags* $\{\sigma_i, t_i\}$*, and the input of* V *consists of the verification secret key* vsk*, verification public key* vpk*, master public key* pk*, and file information* $\mathtt{Param}_F$*.*
- $\mathsf{OwnerVerify}(sk, pk, \mathtt{Context}, \mathtt{Evidence}, \mathtt{Param}_F)) \rightarrow (b_0, b_1)$: *The owner verifier algorithm* OwnerVerify *takes as input the master key pair* (sk, pk) *and* Context *and* Evidence*, and outputs two decision bits* $b_0, b_1 \in \{0, 1\}$*, where* b_0 *indicates accepting or rejecting the storage server, and* b_1 *indicates accepting or rejecting the ODA.*

A DPOS system is described as below and illustrated in Fig. 1(a) and (b).



(a) The framework of DPOS (b) Updating Verification key (vpk, vsk)

Fig. 1. Illustration of system model of DPOS.

Definition 2. *A* DPOS *system among three parties—data owner, cloud storage server and auditor, can be implemented by running a* DPOS *scheme (*KeyGen, Tag, UpdVK,$\langle$P, V$\rangle$, OwnerVerify*) in the following three phases, where the setup phase will execute at the very beginning, for only once (for one file); the proof phase and revoke phase can execute for multiple times and in any (interleaved) order.*

Setup phase. *The data owner runs the key generating algorithm* KeyGen(1^λ) *for only once across all files, to generate the per-user master key pair* (pk, sk) *and the verification key pair* (vpk, vsk)*. For every input data file, the data owner runs the tag algorithm* Tag *over the (possibly erasure encoded) file, to generate authentication tags* $\{(\sigma_i, t_i)\}$ *and file parameter* $\texttt{Param}_F$*. At the end of setup phase, the data owner sends the file* F*, all authentication tags* $\{(\sigma_i, t_i)\}$*, file parameter* $\texttt{Param}_F$*, and public keys* (pk, vpk) *to the cloud storage server. The data owner also chooses an exclusive third party auditor, called* Owner-Delegated-Auditor *(ODA, for short), and delegates the verification key pair* (vpk, vsk) *and file parameter* $\texttt{Param}_F$ *to the ODA. After that, the data owner may keep only keys* (pk, sk, vpk, vsk) *and file parameter* $\texttt{Param}_F$ *in local storage, and delete everything else from local storage.*

Proof phase. *The proof phase consists of multiple proof sessions. In each proof session, the ODA, who runs algorithm* V*, interacts with the cloud storage server, who runs algorithm* P*, to audit the integrity of data owner's file, on behalf of the data owner. Therefore, ODA is also called* verifier *and cloud storage server is also called* prover*. ODA will also keep all outputs of algorithm* V*, i.e. tuples (b,* `Context`*,* `Evidence`*), and allow data owner to fetch and verify these tuples using algorithm* OwnerVerify *at any time.*

Revoke phase. *In the revoke phase, the data owner downloads all tags* $\{t_i\}$ *from cloud storage server, revokes the current verification key pair, and generates a fresh verification key pair and new tags* $\{t'_i\}$*, by running algorithm* UpdVK*. The data owner also chooses a new ODA, and delegates the new verification key pair*

to this new ODA, and sends the updated tags $\{t'_i\}$ *to the cloud storage server to replace the old tags* $\{t_i\}$.

Definition 3 (Completeness). *A* DPOS *scheme (*KeyGen, Tag, UpdVK, $\langle \mathsf{P}, \mathsf{V} \rangle$, OwnerVerify*) is* complete*, if the following condition holds: For any keys* (*pk,sk, vpk,vsk*) *generated by* KeyGen*, for any file* F*, if all parties follow our scheme exactly and the data stored in cloud storage is intact, then interactive proof algorithms* $\langle \mathsf{P}, \mathsf{V} \rangle$ *will always output* $(1, \ldots)$ *and* OwnerVerify *algorithm will always output* $(1, 1)$.

3.2 Trust Model

In this paper, we aim to protect data integrity of data owner's file. The data owner is fully trusted, and the cloud storage server and ODA are semi-trusted in different sense: (1) The cloud storage server is trusted in maintaining service availability and is *not* trusted in maintaining data integrity (e.g. the server might delete some rarely accessed data for economic benefits, or hide the data corruption events caused by server failures or attacks to maintain reputation). (2) Before he/she is revoked, the ODA is trusted in performing the delegated auditing task and protecting his/her verification secret key securely. A revoked ODA could be potentially malicious and might surrender his/her verification secret key to the cloud storage server.

4 Our Proposed Scheme

4.1 Preliminaries

Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative cyclic groups of prime order p. Let g be a randomly chosen generator of group $\mathbb{G}$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a non-degenerate and efficiently computable bilinear map. For vector $\vec{\boldsymbol{a}} = (a_1, \ldots, a_m)$ and $\vec{\boldsymbol{b}} = (b_1, \ldots, b_m)$, the notation $\left\langle \vec{\boldsymbol{a}}, \vec{\boldsymbol{b}} \right\rangle \stackrel{\text{def}}{=} \sum_{j=1}^{m} a_j b_j$ denotes the dot product (a.k.a inner product) of the two vectors $\vec{\boldsymbol{a}}$ and $\vec{\boldsymbol{b}}$. For vector $\vec{\boldsymbol{v}} = (v_0, \ldots, v_{m-1})$ the notation $\mathsf{Poly}_{\vec{\boldsymbol{v}}}(x) \stackrel{\text{def}}{=} \sum_{j=0}^{m-1} v_j x^j$ denotes the polynomial in variable x with $\vec{\boldsymbol{v}}$ being the coefficient vector.

4.2 Construction of the Proposed DPOS Scheme

We define our DPOS scheme (KeyGen, Tag, UpdVK, $\langle \mathsf{P}, \mathsf{V} \rangle$, OwnerVerify) as below, and these algorithms will run in the way as specified in Definition 2. We remind that in the following description of algorithms, some equations have inline explanation highlighted in box, which is not a part of algorithm procedures but could be useful to understand the correctness of our algorithms.

$\underline{\mathsf{KeyGen}(1^\lambda) \to (pk, sk, vpk, vsk)}$ Choose at random a λ-bits prime p and a bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G}$ and $\mathbb{G}_T$ are both multiplicative cyclic

groups of prime order p. Choose at random a generator $g \in \mathbb{G}$. Choose at random $\alpha, \gamma, \rho \in_R \mathbb{Z}_p^*$, and $(\beta_1, \beta_2, \ldots, \beta_m) \in_R (\mathbb{Z}_p)^m$. For each $j \in [1, m]$, define $\alpha_j := \alpha^j \mod p$, and compute $g_j := g^{\alpha_j}$, $h_j := g^{\rho \cdot \beta_j}$. Let $\alpha_0 := 1$, $\beta_0 := 1$, $g_0 = g^{\alpha^0} = g$, $h_0 = g^\rho$, vector $\vec{\boldsymbol{\alpha}} := (\alpha_1, \alpha_2, \ldots, \alpha_m)$, and $\vec{\boldsymbol{\beta}} := (\beta_1, \beta_2, \ldots, \beta_m)$. Choose two random seeds s_0, s_1 for pseudorandom function $\mathcal{PRF}_{\mathtt{seed}} : \{0,1\}^\lambda \times \mathbb{N} \to \mathbb{Z}_p$.

The secret key is $sk = (\alpha, \vec{\boldsymbol{\beta}}, s_0)$ and the public key is $pk = (g_0, g_1, \ldots, g_m)$. The verification secret key is $vsk = (\rho, \gamma, s_1)$ and the verification public key is $vpk = (h_0, h_1, \ldots, h_m)$.

$\underline{\mathsf{Tag}(sk, vsk, F) \to (\mathtt{Param}_F, \{(\sigma_i, t_i)\})}$ Split file[2] F into n blocks, where each block is treated as a vector of m elements from $\mathbb{Z}_p$: $\{\vec{\boldsymbol{F}}_i = (F_{i,0}, \ldots, F_{i,m-1}) \in \mathbb{Z}_p^m\}_{i \in [0,n-1]}$. Choose a unique identifier $\mathtt{id}_F \in \{0,1\}^\lambda$ for file F. Define a customized[3] pseudorandom function w.r.t. the file F: $\mathcal{R}_s(i) = \mathcal{PRF}_s(\mathtt{id}_F, i)$.

For each block $\vec{\boldsymbol{F}}_{\boldsymbol{i}}$, $0 \le i \le n-1$, compute

$$\sigma_i := \left\langle \vec{\boldsymbol{\alpha}}, \vec{\boldsymbol{F}}_{\boldsymbol{i}} \right\rangle + \mathcal{R}_{s_0}(i) \quad \boxed{= \alpha \cdot \mathsf{Poly}_{\vec{\boldsymbol{F}}_{\boldsymbol{i}}}(\alpha) + \mathcal{R}_{s_0}(i) \mod p} \tag{1}$$

$$t_i := \rho \left\langle \vec{\boldsymbol{\beta}}, \vec{\boldsymbol{F}}_{\boldsymbol{i}} \right\rangle + \gamma \mathcal{R}_{s_0}(i) + \mathcal{R}_{s_1}(i) \mod p \tag{2}$$

The general information of F is $\mathtt{Param}_F := (\mathtt{id}_F, n)$.

$\underline{\mathsf{UpdVK}(vpk, vsk, \{t_i\}_{i \in [0,n-1]}) \to (vpk', vsk', \{t'_i\}_{i \in [0,n-1]})}$ Parse vpk as $(h_0, \ldots, h_m)$ and vsk as (ρ, γ, s_1). Verify the integrity of all tags $\{t_i\}$ (*We will discuss how to do this verification later*), and abort if the verification fails. Choose at random $\gamma' \in_R \mathbb{Z}_p^*$ and choose a random seed s'_1 for pseudorandom function $\mathcal{R}$. For each $j \in [0, m]$, compute $h'_j := h_j^{\gamma'} = g^{(\rho \cdot \gamma') \cdot \beta_j} \in \mathbb{G}$. For each $i \in [0, n-1]$, compute a new authentication tag

$$\begin{aligned} t'_i :=& \gamma' (t_i - \mathcal{R}_{s_1}(i)) + \mathcal{R}_{s'_1}(i) \mod p. \\ =& \boxed{\gamma' \cdot \rho \left\langle \vec{\boldsymbol{\beta}}, \vec{\boldsymbol{F}}_{\boldsymbol{i}} \right\rangle + (\gamma' \cdot \gamma) \mathcal{R}_{s_0}(i) + \mathcal{R}_{s'_1}(i) \mod p} \end{aligned}$$

The new verification public key is $vpk' := (h'_0, \ldots, h'_m)$ and the new verification secret key is $vsk' := (\gamma' \cdot \rho, \ \gamma' \cdot \gamma, \ s'_1)$.

$\underline{\langle \mathsf{P}(pk, vpk, \{\vec{\boldsymbol{F}}_i, \sigma_i, t_i\}_{i \in [0,n-1]}), \mathsf{V}(vsk, vpk, pk, \mathtt{Param}_F) \rangle \to (b, \mathtt{Context}, \mathtt{Evidence})}$

V1: Verifier parses $\mathtt{Param}_F$ as $(\mathtt{id}_F, n)$. Verifier chooses a random subset $\mathbf{C} = \{i_1, i_2, \ldots, i_c\} \subset [0, n-1]$ of size c, where $i_1 < i_2 < \ldots < i_c$. Choose at random $w, \xi \in_R \mathbb{Z}_p^*$, and compute $w_{i_\iota} := w^\iota \mod p$ for each $\iota \in [1, c]$. Verifier sends $(\mathtt{id}_F, \{(i, w_i) : i \in \mathbf{C}\}, \xi)$ to the prover to initiate a proof session.

[2] Possibly, the input has been encoded by the data owner using some error erasure code.

[3] With such a customized function $\mathcal{R}$, the input $\mathtt{id}_F$ will become *implicit* and this will make our expression short.

P1: Prover finds the file and tags $\{(\vec{\boldsymbol{F}}_i, \sigma_i, t_i)\}_i$ corresponding to $\mathtt{id}_F$. Prover computes $\vec{\boldsymbol{\mathcal{F}}} \in \mathbb{Z}_p^m$, and $\bar{\sigma}, \bar{t} \in \mathbb{Z}_p$ as below.

$$\vec{\boldsymbol{\mathcal{F}}} := \left(\sum_{i \in \mathbf{C}} w_i \vec{\boldsymbol{F}}_i\right) \mod p; \tag{3}$$

$$\bar{\sigma} := \left(\sum_{i \in \mathbf{C}} w_i \sigma_i\right) \mod p; \tag{4}$$

$$\bar{t} := \left(\sum_{i \in \mathbf{C}} w_i t_i\right) \mod p. \tag{5}$$

Evaluate polynomial $\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(x)$ at point $x = \xi$ to obtain $z := \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi) \mod p$. Divide the polynomial (in variable x) $\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(x) - \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi)$ with $(x - \xi)$ using polynomial long division, and denote the coefficient vector of resulting quotient polynomial as $\vec{\boldsymbol{v}} = (v_0, \ldots v_{m-2})$, that is, $\mathsf{Poly}_{\vec{\boldsymbol{v}}}(x) \equiv \frac{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(x) - \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi)}{x - \xi} \mod p$. (*Note* : $(x - \xi)$ *can divide polynomial* $\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(x) - \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi)$ *perfectly, since the latter polynomial evaluates to 0 at point* $x = \xi$.)

Compute $(\psi_\alpha, \psi_\beta, \phi_\alpha) \in \mathbb{G}^3$ as below

$$\psi_\alpha := \prod_{j=0}^{m-1} g_j^{\vec{\boldsymbol{\mathcal{F}}}[j]} \boxed{= \prod_{j=0}^{m-1} \left(g^{\alpha^j}\right)^{\vec{\boldsymbol{\mathcal{F}}}[j]} = g^{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)};} \tag{6}$$

$$\psi_\beta := \prod_{j=0}^{m-1} h_{j+1}^{\vec{\boldsymbol{\mathcal{F}}}[j]} \boxed{= \prod_{j=0}^{m-1} \left(g^{\rho \cdot \beta_{j+1}}\right)^{\vec{\boldsymbol{\mathcal{F}}}[j]} = g^{\rho \langle \vec{\boldsymbol{\beta}},\, \vec{\boldsymbol{\mathcal{F}}} \rangle};} \tag{7}$$

$$\phi_\alpha := \prod_{j=0}^{m-2} g_j^{v_j} \boxed{= \prod_{j=0}^{m-2} \left(g^{\alpha^j}\right)^{v_j} = g^{\mathsf{Poly}_{\vec{\boldsymbol{v}}}(\alpha)}.} \tag{8}$$

Prover sends $(z, \phi_\alpha, \bar{\sigma}, \bar{t}, \psi_\alpha, \psi_\beta)$ to the verifier.

V2: Let $\mathtt{Context} := (\xi, \{(i, w_i) : i \in \mathbf{C}\})$ and $\mathtt{Evidence} := (z,\ \phi_\alpha,\ \bar{\sigma})$. Verifier sets $b := \mathtt{1}$ if the following equalities hold and sets $b := \mathtt{0}$ otherwise.

$$e(\psi_\alpha, g) \stackrel{?}{=} e(\phi_\alpha,\ g^\alpha / g^\xi) \cdot e(g, g)^z \tag{9}$$

$$\left(\frac{e(\psi_\alpha,\ g^\alpha)}{e\,(g,\ g^{\bar{\sigma}})}\right)^\gamma \stackrel{?}{=} \frac{e(\psi_\beta,\ g)}{e\left(g,\ g^{\bar{t}} \cdot g^{-\sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_1}(i)}\right)} \tag{10}$$

Output $(b, \mathtt{Context}, \mathtt{Evidence})$.

$\underline{\mathsf{OwnerVerify}(sk, pk, b, \mathtt{Context}, \mathtt{Evidence}, \mathtt{Param}_F)) \to (b_0, b_1)}$

Parse $\mathtt{Context}$ as $(\xi, \{(i, w_i) : i \in \mathbf{C}\})$ and parse $\mathtt{Evidence}$ as $(z, \phi_\alpha, \bar{\sigma})$. Verifier

will set $b_0 := 1$ if the following equality hold; otherwise set $b_0 := 0$.

$$\left(e(\phi_\alpha, g^\alpha / g^\xi) e(g,g)^z\right)^\alpha \stackrel{?}{=} e(g,\ g^{\bar{\sigma}}) \cdot e(g,g)^{\left(-\sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)\right)} \tag{11}$$

If ODA's decision b equals to b_0, then set $b_1 := 1$; otherwise set $b_1 := 0$. Output (b_0, b_1).

The completeness of the above scheme is proved in the full paper [46].

4.3 Discussion

How to verify the integrity of all tag values $\{t_i\}$ *in algorithm* UpdVK*?* A straightforward method is that: The data owner keeps tack a hash (e.g. SHA256) value of $t_0 \| t_1 \ldots \| t_{n-1}$ in local storage, and updates this hash value when executing UpdVK.

How to reduce the size of challenge $\{(i, w_i) : i \in \mathbf{C}\}$*?* Dodis *et al.* [15]'s result can be used to represent a challenge $\{(i, w_i) : i \in \mathbf{C}\}$ compactly as below: Choose the subset $\mathbf{C}$ using Goldreich [18]'s (δ, ϵ)-hitter[4], where the subset $\mathbf{C}$ can be represented compactly with only $\log n + 3\log(1/\epsilon)$ bits. Assume $n < 2^{40}$ (sufficient for practical file size) and let $\epsilon = 2^{-80}$. Then $\mathbf{C}$ can be represented with 280 bits. Recall that $\{w_i : i \in \mathbf{C}\}$ is derived from some single value $w \in \mathbb{Z}_p^*$.

4.4 Experiment Result

We implement a prototype of our scheme in C language and using GMP[5] and PBC[6] library. We run the prototype in a Laptop PC with a 2.5 GHz Intel Core 2 Duo mobile CPU (model T9300, released in 2008). Our test files are randomly generated and of size from 128 MB to 1 GB. We achieve a throughput of data preprocessing at speed slightly larger than 10 megabytes per second, with $\lambda = 1024$.

In contrast, Atenesis *et al.* [3,4] achieves throughput of data preprocessing at speed 0.05 megabytes per second with a 3.0 GHz desktop CPU [4]. Wang *et al.* [38] achieves throughput of data pre-processing at speed 9.0KB/s and 17.2KB/s with an Intel Core 2 1.86 GHz workstation CPU, when a data block is a vector of dimension $m = 1$ and $m = 10$, respectively. According to the pre-processing complexity of [38] shown in Table 1, the theoretical optimal throughput speed of [38] is twice of the speed for dimension $m = 1$, which can be approached only when m tends to $+\infty$.

Therefore, the data pre-processing in our scheme is 200 times faster than Atenesis et al. [3,4], and 500 times faster than Wang *et al.* [38], using a single

[4] Goldreich [18]'s (δ, ϵ)-hitter guarantees that, for any subset $W \subset [0, n-1]$ with size $|W| \geq (1-\delta)n$, $\Pr[\mathbf{C} \cap W \neq \emptyset] \geq 1 - \epsilon$. Readers may refer to [15] for more details.

[5] GNU Multiple Precision Arithmetic Library: https://gmplib.org/.

[6] The Pairing-Based Cryptography Library: http://crypto.stanford.edu/pbc/.

CPU core. We remark that, all of these schemes (ours and [3,4,38]) and some others can be speedup by N times using N CPU cores in parallel. However, typical cloud user who runs the data pre-processing task, might have CPU cores number ≤ 4.

5 Security Analysis

5.1 Security Formulation

We will define soundness security in two layers. Intuitively, if a cloud storage server can pass auditor's verification, then there exists an efficient extractor algorithm, which can output the challenged data blocks. Furthermore, if a cloud storage server with knowledge of verification secret key can pass data owner's verification, then there exists an efficient extractor algorithm, which can output the challenged data blocks. If the data file is erasure encoded in advance, the whole data file could be decoded from sufficiently amount of challenged data blocks.

5.1.1 Definition of Soundness w.r.t Verification of Auditor

Based on the existing Provable Data Possession formulation [4] and Proofs of Retrievability formulation [22,30], we define DPOS soundness security game $\mathsf{Game}_{\mathtt{sound}}$ between a *probabilistic polynomial time* (PPT) adversary $\mathcal{A}$ (i.e. dishonest prover/cloud storage server) and a PPT challenger $\mathcal{C}$ w.r.t. a DPOS scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{Tag}, \mathsf{UpdVK}, \langle \mathsf{P}, \mathsf{V} \rangle, \mathsf{OwnerVerify})$ as below.

Setup: The challenger $\mathcal{C}$ runs the key generating algorithm $\mathsf{KeyGen}(1^\lambda)$ to obtain two pair of public-private keys (pk, sk) and (vpk, vsk). The challenger $\mathcal{C}$ gives the public key (pk, vpk) to the adversary $\mathcal{A}$ and keeps the private key (sk, vsk) securely.

Learning: The adversary $\mathcal{A}$ adaptively makes polynomially many queries, where each query is one of the following:

- STORE-QUERY($\mathtt{F}$): Given a data file $\mathtt{F}$ chosen by $\mathcal{A}$, the challenger $\mathcal{C}$ runs tagging algorithm $(\mathtt{Param}_F, \{(\sigma_i, t_i)\}) \leftarrow \mathsf{Tag}(sk, vsk, \mathtt{F})$, where $\mathtt{Param}_F = (\mathtt{id}_F, n)$, and sends the data file $\mathtt{F}$, authentication tags $\{(\sigma_i, t_i)\}$, public keys (pk, vpk), and file parameter $\mathtt{Param}_F$, to $\mathcal{A}$.
- VERIFY-QUERY($\mathtt{id}_F$): Given a file identifier $\mathtt{id}_F$ chosen by $\mathcal{A}$, if $\mathtt{id}_F$ is not the (partial) output of some previous STORE-QUERY that $\mathcal{A}$ has made, ignore this query. Otherwise, the challenger $\mathcal{C}$ initiates a proof session with $\mathcal{A}$ w.r.t. the data file $\mathtt{F}$ associated to the identifier $\mathtt{id}_F$ in this way: The adversary $\mathcal{C}$, who runs the verifier algorithm $\mathsf{V}(vsk, vpk, pk, \mathtt{Param}_F)$, interacts with the adversary $\mathcal{A}$, who replaces the prover algorithm P with any PPT algorithm of its choice, and obtains an output $(b, \mathtt{Context}, \mathtt{Evidence})$, where $b \in \{1, 0\}$. The challenger runs the algorithm $\mathsf{OwnerVerify}(b, \mathtt{Context}, \mathtt{Evidence})$ to obtain output $(b_0, b_1) \in \{0, 1\}^2$. The challenger sends the two decision bits (b, b_0) to the adversary as feedback.

- RevokeVK-Query: To respond to this query, the challenger runs the verification key update algorithm to obtain a new pair of verification keys $(vpk', vsk', \{t_i'\}) := \mathsf{UpdVK}(vpk, vsk, \{t_i\})$, and sends the revoked verification secret key vsk and the new verification public key vpk' and new authentication tags $\{t_i'\}$ to the adversary $\mathcal{A}$, and keeps vsk' private.

Commit: Adversary $\mathcal{A}$ outputs and commits on $(\mathtt{id}^*, \mathtt{Memo}, \tilde{\mathsf{P}})$, where each of them is described as below:

- a file identifier $\mathtt{id}^*$ among all file identifiers it obtains from $\mathcal{C}$ by making Store-Queries in **Learning** phase;
- a bit-string $\mathtt{Memo}$;
- a description of PPT prover algorithm $\tilde{\mathsf{P}}$ (e.g. an executable binary file).

Challenge: The challenger randomly chooses a subset $\mathbf{C}^* \subset [0,\ n_{F^*} - 1]$ of size $c < \lambda^{0.9}$, where $\mathtt{F}^*$ denotes the data file associated to identifier $\mathtt{id}^*$, and n_{F^*} is the number of blocks in file F^*.

Extract: Let $\mathcal{E}^{\tilde{\mathsf{P}}(\mathtt{Memo})}(vsk, vpk, pk, \mathtt{Param}_{F^*}, \mathtt{id}^*, \mathbf{C}^*)$ denote a knowledge-extractor algorithm with oracle access to prover algorithm $\tilde{\mathsf{P}}(\mathtt{Memo})$. More precisely, the extractor algorithm $\mathcal{E}$ will revoke the verifier algorithm $\mathsf{V}(vsk, vpk, pk, \mathtt{Param}_{F^*})$ to interact with $\tilde{\mathsf{P}}(\mathtt{Memo})$, and observes all communication between the prover and verifier. It is worthy pointing out that: (1) the extractor $\mathcal{E}$ can feed input (including random coins) to the verifier V, and cannot access the internal states (e.g. random coins) of the prover $\tilde{\mathsf{P}}(\mathtt{Memo})$, unless the prover $\tilde{\mathsf{P}}$ sends its internal states to verifier; (2) the extractor $\mathcal{E}$ can rewind the algorithm $\tilde{\mathsf{P}}$, as in formulation of Shacham and Waters [30,31]. The goal of this knowledge extractor is to output data blocks $\{(i, \mathtt{F}_i') : i \in \mathbf{C}^*\}$.

The adversary $\mathcal{A}$ wins this DPOS soundness security game $\mathsf{Game}_{\mathtt{Sound}}$, if the verifier algorithm $\mathsf{V}(vsk, vpk, pk, \mathtt{Param}_{F^*})$ accepts the prover algorithm $\tilde{\mathsf{P}}(\mathtt{Memo})$ with some noticeable probability $1/\lambda^\tau$ for some positive integer τ, where the sampling set is fixed as $\mathbf{C}^*$. More precisely,

$$\mathsf{Pr}\left[\langle \tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{V}(vsk, vpk, pk, \mathtt{Param}_{F^*})\rangle = (1, \ldots) \,\middle|\, \begin{array}{c}\mathtt{Sampling}\\ \mathtt{set\ is}\ \mathbf{C}^*\end{array}\right] \geq 1/\lambda^\tau. \quad (12)$$

The challenger $\mathcal{C}$ wins this game, if these exists PPT knowledge extractor algorithm $\mathcal{E}$ such that the extracted blocks $\{(i, \mathtt{F}_i') : i \in \mathbf{C}^*\}$ are identical to the original $\{(i, \mathtt{F}_i) : i \in \mathbf{C}^*\}$ with overwhelming high probability. That is,

$$\mathsf{Pr}\left[\mathcal{E}^{\tilde{\mathsf{P}}(\mathtt{Memo})}(vsk, vpk, pk, \mathtt{Param}_{F^*}, \mathtt{id}^*, \mathbf{C}^*) = \{(i, \mathtt{F}_i) : i \in \mathbf{C}^*\}\right] \geq 1 - negl(\lambda). \quad (13)$$

Definition 4 (Soundness-1). *A* DPOS *scheme is* sound *against dishonest cloud storage server w.r.t. auditor, if for any PPT adversary $\mathcal{A}$, $\mathcal{A}$ wins the above* DPOS *security game* $\mathsf{Game}_{\mathtt{Sound}}$ *implies that the challenger $\mathcal{C}$ wins the same security game.*

5.1.2 Definition of Soundness w.r.t Verification of Owner

We define $\mathsf{Game2}_{\mathtt{sound}}$ by modifying the DPOS soundness security game $\mathsf{Game}_{\mathtt{sound}}$ as below: (1) In the **Setup** phase, the verification private key vsk is given to the adversary $\mathcal{A}$; (2) in the **Extract** phase, the knowledge extractor has oracle access to $\mathsf{OwnerVerify}(sk, \ldots)$, additionally.

Definition 5 (Soundness-2). *A* DPOS *scheme is* sound *against dishonest cloud storage server w.r.t. owner, if for any PPT adversary $\mathcal{A}$, $\mathcal{A}$ wins the above* DPOS *security game* $\mathsf{Game2}_{\mathtt{sound}}$*, i.e.*

$$\mathsf{Pr}\left[\begin{array}{l|l} \mathsf{OwnerVerify}(sk, pk, \langle \tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{V}(vsk, vpk, pk, \mathtt{Param}_{F^*})\rangle) & \textit{Sampling} \\ = (1, \ldots) & \textit{set is } \mathbf{C}^* \end{array}\right] \geq 1/\lambda^{\tau}, \quad \textit{for some positive integer constant } \tau, \tag{14}$$

implies that the challenger $\mathcal{C}$ wins the same security game, i.e. there exists PPT knowledge extractor algorithm $\mathcal{E}$ such that

$$\mathsf{Pr}\left[\mathcal{E}^{\tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{OwnerVerify}(sk, \ldots)}(vsk, vpk, pk, \mathtt{Param}_{F^*}, \mathtt{id}^*, \mathbf{C}^*) = \{(i, \mathtt{F}_i) : i \in \mathbf{C}^*\}\right] \geq 1 - negl(\lambda) \tag{15}$$

Remarks

- The two events "adversary $\mathcal{A}$ wins" and "challenger $\mathcal{C}$ wins" are not *mutually exclusive*.
- The above knowledge extractor formulates the notion that "**data owner is *capable* to recover data file efficiently (i.e. in polynomial time) from the cloud storage server**", if the cloud storage sever can pass verification with noticeable probability and its behavior will not change any more. The knowledge extractor might also serve as the contingency plan[7] (or last resort) to recover data file, when downloaded file from cloud is always corrupt but the cloud server can always pass the verification with high probability.
- Unlike POR [30,31], our formulation separates "error correcting code" out from POS scheme, since error correcting code is orthogonal to our design of homomorphic authentication tag function. If required, error correcting code can be straightforwardly combined with our DPOS scheme, and the analysis of such combination is almost identical to previous works.

5.2 Security Claim

Definition 6. (m-Bilinear Strong Diffie-Hellman Assumption). *Let $e : \mathbb{G} \times \mathbb{G} \rightarrow \mathbb{G}_T$ be a bilinear map where $\mathbb{G}$ and $\mathbb{G}_T$ are both multiplicative cyclic groups of prime order p. Let g be a randomly chosen generator of group $\mathbb{G}$. Let $\varsigma \in_R \mathbb{Z}_p^*$ be chosen at random. Given as input a $(m+1)$-tuple*

[7] Cloud server's cooperation might be required.

$\mathbf{T} = (g, g^{\varsigma}, g^{\varsigma^2} \ldots, g^{\varsigma^m}) \in \mathbb{G}^{m+1}$, *for any PPT adversary* $\mathcal{A}$, *the following probability is negligible*

$$\mathsf{Pr}\left[d = e(g,g)^{1/(\varsigma+c)} \textit{ where } (c,d) = \mathcal{A}(\mathbf{T})\right] \leq negl(\log p).$$

Theorem 1. *Suppose m-BSDH Assumption hold, and* PRF *is a secure pseudorandom function. The* DPOS *scheme constructed in Sect. 4 is sound w.r.t. auditor, according to Definition 4 (Proof is given in the full paper [46]).*

Theorem 2. *Suppose m-BSDH Assumption hold, and* PRF *is a secure pseudorandom function. The* DPOS *scheme constructed in Sect. 4 is sound w.r.t. data owner, according to Definition 5 (Proof is given in the full paper [46]).*

6 Conclusion

We proposed a novel and efficient POS scheme. On one side, the proposed scheme is as efficient as privately verifiable POS scheme, especially very efficient in authentication tag generation. On the other side, the proposed scheme supports third party auditor and can revoke an auditor at any time, close to the functionality of publicly verifiable POS scheme. Compared to existing publicly verifiable POS scheme, our scheme improves the authentication tag generation speed by more than 100 times. How to prevent data leakage to ODA during proof process and how to enable dynamic operations (e.g. inserting/deleting a data block) in our scheme are in future work.

References

1. Apon, D., Huang, Y., Katz, J., Malozemoff, A.J.: Implementing cryptographic program obfuscation. Cryptology ePrint Archive, Report 2014/779 (2014). http://eprint.iacr.org/
2. Armknecht, F., Bohli, J.M., Karame, G.O., Liu, Z., Reuter, C.A.: Outsourced proofs of retrievability. In: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, CCS 2014, pp. 831–843 (2014)
3. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Khan, O., Kissner, L., Peterson, Z., Song, D.: Remote data checking using provable data possession. ACM Tran. Inf. Sys. Sec. TISSEC 2011 **14**(1), 12:1–12:34 (2011)
4. Ateniese, G., Burns, R., Curtmola, R., Herring, J., Kissner, L., Peterson, Z., Song, D.: Provable data possession at untrusted stores. In: ACM CCS 2007, pp. 598–609. ACM (2007)
5. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: Matsui, M. (ed.) ASIACRYPT 2009. LNCS, vol. 5912, pp. 319–333. Springer, Heidelberg (2009)
6. Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: SecureComm 2008, pp. 9:1–9:10. ACM (2008)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. J. Cryptology **17**(4), 297–319 (2004)

8. Bowers, K.D., Juels, A., Oprea, A.: HAIL: A high-availability and integrity layer for cloud storage. In: ACM CCS 2009, pp. 187–198. ACM (2009)
9. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: theory and implementation. In: CCSW 2009, pp. 43–54. ACM (2009)
10. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. In: Johansson, T., Nguyen, P.Q. (eds.) EUROCRYPT 2013. LNCS, vol. 7881, pp. 279–295. Springer, Heidelberg (2013)
11. Chang, E.-C., Xu, J.: Remote integrity check with dishonest storage server. In: Jajodia, S., Lopez, J. (eds.) ESORICS 2008. LNCS, vol. 5283, pp. 223–237. Springer, Heidelberg (2008)
12. Chen, X., Li, J., Ma, J., Tang, Q., Lou, W.: New algorithms for secure outsourcing of modular exponentiations. In: Foresti, S., Yung, M., Martinelli, F. (eds.) ESORICS 2012. LNCS, vol. 7459, pp. 541–556. Springer, Heidelberg (2012)
13. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: MR-PDP: multiple-replica provable data possession. In: ICDCS 2008, pp. 411–420. IEEE (2008)
14. Deswarte, Y., Quisquater, J.J., Saïdane, A.: Remote integrity checking: how to trust files stored on untrusted servers. In: Jajodia, S., Strous, L. (eds.) IICIS 2003. IFIP, vol. 140, pp. 1–11. Springer, Heidelberg (2004)
15. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of retrievability via hardness amplification. In: Reingold, O. (ed.) TCC 2009. LNCS, vol. 5444, pp. 109–127. Springer, Heidelberg (2009)
16. Erway, C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: ACM CCS 2009, pp. 213–222. ACM (2009)
17. Garg, S., Gentry, C., Halevi, S., Raykova, M., Sahai, A., Waters, B.: Candidate indistinguishability obfuscation and functional encryption for all circuits. In: Proceedings of the 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, FOCS 2013, pp. 40–49 (2013)
18. Goldreich, O.: A sample of samplers: a computational perspective on sampling. In: Goldreich, O. (ed.) Studies in Complexity and Cryptography. LNCS, vol. 6650, pp. 302–332. Springer, Heidelberg (2011)
19. Guan, C., Ren, K., Zhang, F., Kerschbaum, F., Yu, J.: A symmetric-key based proofs of retrievability supporting public verification. In: Proceedings of 20th European Symposium on Research in Computer Security, ESORICS 2015, pp. 203–223 (2015). www.fkerschbaum.org/esorics15b.pdf
20. Hao, Z., Zhong, S., Yu, N.: A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. In:TKDE 2011, vol. 23(9), pp. 1432–1437 (2011)
21. Hohenberger, S., Lysyanskaya, A.: How to securely outsource cryptographic computations. In: Kilian, J. (ed.) TCC 2005. LNCS, vol. 3378, pp. 264–282. Springer, Heidelberg (2005)
22. Juels, A., Kaliski, B.S.,J.: PORs: Proofs of retrievability for large files. In: ACM CCS 2007, pp. 584–597. ACM (2007)
23. Kate, A., Zaverucha, G.M., Goldberg, I.: Constant-size commitments to polynomials and their applications. In: ASIACRYPT 2010, pp. 177–194 (2010)
24. Naor, M., Rothblum, G.N.: The complexity of online memory checking. J. ACM, **56**(1) (2009)
25. Ren, Y., Shen, J., Wang, J., Fang, L.: Outsourced data tagging via authority and delegable auditing for cloud storage. In: 49th Annual IEEE International Carnahan Conference on Security Technology, ICCST 2015, pp. 131–134. IEEE (2015)
26. Ren, Y., Shen, J., Wang, J., Han, J., Lee, S.: Mutual verifiable provable data auditing in public cloud storage. J. Internet Technol. **16**(2), 317–324 (2015)

27. Ren, Y., Xu, J., Wang, J., Kim, J.U.: Designated-verifier provable data possession in public cloud storage. Int. J. Secur. Appl. **7**(6), 11–20 (2013)
28. Schwarz, T.J.E., Miller, E.L.: Store, forget, and check: using algebraic signatures to check remotely administered storage. In: ICDCS 2006. IEEE (2006)
29. Sebé, F., Domingo-Ferrer, J., Martínez-Ballesté, A., Deswarte, Y., Quisquater, J.J.: Efficient remote data possession checking in critical information infrastructures. In: TKDE 2008, vol. 20, no. 8, pp. 1034–1038 (2008)
30. Shacham, H., Waters, B.: Compact proofs of retrievability. In: Pieprzyk, J. (ed.) ASIACRYPT 2008. LNCS, vol. 5350, pp. 90–107. Springer, Heidelberg (2008)
31. Shacham, H., Waters, B.: Compact proofs of retrievability. J. Cryptology **26**(3), 442–483 (2013)
32. Shah, M.A., Baker, M., Mogul, J.C., Swaminathan, R.: Auditing to keep online storage services honest. In: HotOS 2007. USENIX Association (2007)
33. Shah, M.A., Swaminathan, R., Baker, M.: Privacy-preserving audit and extraction of digital contents. Cryptology ePrint Archive, Report 2008/186 (2008). http://eprint.iacr.org/2008/186
34. Shen, S.-T., Tzeng, W.-G.: Delegable provable data possession for remote data in the clouds. In: Qing, S., Susilo, W., Wang, G., Liu, D. (eds.) ICICS 2011. LNCS, vol. 7043, pp. 93–111. Springer, Heidelberg (2011)
35. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: ACM CCS 2013, pp. 325–336. ACM (2013)
36. Wang, B., Li, B., Li, H.: Oruta: Privacy-preserving public auditing for shared data in the cloud. In: IEEE Cloud 2012, pp. 295–302. IEEE (2012)
37. Wang, B., Li, B., Li, H.: Public auditing for shared data with efficient user revocation in the cloud. In: INFOCOM 2013, pp. 2904–2912. IEEE (2013)
38. Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Trans. Comput. **62**(2), 362–375 (2013)
39. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Network Mag. **24**(4), 19–24 (2010)
40. Wang, C., Wang, Q., Ren, K., Cao, N., Lou, W.: Towards secure and dependable storate services in cloud computing. IEEE Trans. Serv. Comput. **5**(2), 220–232 (2012)
41. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring data storage security in cloud computing. In: Proceedings of IWQoS 2009, pp. 1–9. IEEE (2009)
42. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: INFOCOM 2010, pp. 525–533. IEEE (2010)
43. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: Backes, M., Ning, P. (eds.) ESORICS 2009. LNCS, vol. 5789, pp. 355–370. Springer, Heidelberg (2009)
44. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. TPDS **22**(5), 847–859 (2011)
45. Xu, J., Chang, E.C.: Towards efficient proofs of retrievability. In: ACM Symposium on Information, Computer and Communications Security, AsiaCCS 2012 (2012)
46. Xu, J., Yang, A., Zhou, J., Wong, D.S.: Lightweight and privacy-preserving delegatable proofs of storage. Cryptology ePrint Archive, Report 2014/395 (2014). http://eprint.iacr.org/2014/395
47. Xu, J., Zhou, J.: Leakage resilient proofs of ownership in cloud storage, revisited. In: Boureanu, I., Owesarski, P., Vaudenay, S. (eds.) ACNS 2014. LNCS, vol. 8479, pp. 97–115. Springer, Heidelberg (2014)

48. Yang, K., Jia, X.: Data storage auditing service in cloud computing: challenges, methods and opportunities. World Wide Web **15**(4), 409–428 (2012)
49. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. TPDS **24**(9), 1717–1726 (2013)
50. Yuan, J., Yu, S.: Proofs of retrievability with public verifiability and constant communication cost in cloud. In: Proceedings of the 2013 International Workshop on Security in Cloud Computing, Cloud Computing 2013, pp. 19–26. ACM (2013)
51. Zeng, K.: Publicly verifiable remote data integrity. In: Chen, L., Ryan, M.D., Wang, G. (eds.) ICICS 2008. LNCS, vol. 5308, pp. 419–434. Springer, Heidelberg (2008)
52. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in multicloud storage. TPDS **23**(12), 2231–2244 (2012)
53. Zhu, Y., Wang, H., Hu, Z., Ahn, G.J., Hu, H., Yau, S.S.: Dynamic audit services for integrity verification of outsourced storages in clouds. In: Proceedings of SAC 2011, pp. 1550–1557. ACM (2011)