Lightweight Delegatable Proofs of Storage*

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

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

Since Proofs of Retrievability (POR [26]) and Provable Data Possession (PDP [4]) are proposed in 2007, a lot of effort of research community is devoted to constructing proofs of storage schemes with more advanced features. The new features include, public key verifiability [34], supporting dynamic operations [11, 20, 39] (i.e. inserting/deleting/editing a data block), supporting multiple cloud servers [16], privacy-preserving against auditor [46], and supporting data sharing [41], etc.

1.1 Drawback 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 [27]: 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.* [42] achieves throughput of data pre-processing (i.e. generating authentication tag) at speed 17.2KB/s with an Intel Core 2 1.86GHz workstation CPU, which means it will take about 17 hours to generate authentication tags for a 1GB file. Even if the user has a CPU with 8 cores, it still requires more than 2 hours' heavy

* (1) The full version [51] with all details of proof is available at http://eprint.iacr.org/2014/395. (2) Anjia Yang contributes to this work when he takes his internship in Institute for Infocomm Research, Singapore. (3) Both Jia Xu and Anjia Yang are corresponding authors.

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 shoud 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 hundreds times without sacrificing effiency on other aspects.

In many publicly verifiable POS (POR/PDP) scheme (e.g. [4, 34, 42, 46]), 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 has to 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 Approaches to Mitigate Drawback

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. [25, 49]) 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 [34] 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, 34, 42, 46]). 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, [14] proposed to construct publicly verifiable POR from privately verifiable POR using indistinguishability obfuscation technique [21]. 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 [14] and [21] admit that indistinguishability obfuscation is currently impractical. Particularly, [17] implements the scheme of [21] and shows that, it requires about 9 hours 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 hours 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 [34], 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

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

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.* [1] 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 support 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 than 10MB/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 (Theorem 1 and Theorem 2 on page 13) 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, 13, 16, 18–20, 24, 26, 32–37, 40–48, 50, 52–57]. 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 [26] 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 [34, 35] 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 [26]. Similar to [4], their scheme utilized homomorphic authenticators built from BLS signatures [7]. Subsequently, Zeng *et al.* [55], Wang *et al.* [47, 48] 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 [42, 46], 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.* [57] introduced another privacy-preserving public auditing scheme. Later, Hao *et al.* [24] and Yang *et al.* [53] proposed two privacy-preserving public auditing schemes without applying the masking technique. Yuan *et al.* [54] gave a POR scheme with public verifiability and constant communication cost. Ren [30] designs 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

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
[34, 35]-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	$m\ell$
[47, 48]	$2\frac{\lvert F\rvert}{\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	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
[46]	$2\frac{\lvert F\rvert}{\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	0	$\ell\lambda + \ell\log(\frac{\lvert F\rvert}{m\lambda})$	3λ	$\lvert F\rvert$	ℓ	ℓ	2	0	ℓ	2ℓ	0	ℓ
[42]†	$\frac{\lvert F\rvert}{\lambda} + \frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	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	$m\ell$
[57]	$\frac{\lvert F\rvert}{m\lambda} + m$	$2\frac{\lvert F\rvert}{\lambda} + m$	$\frac{\lvert F\rvert}{\lambda} + m$	$\ell\lambda + \ell\log(\frac{\lvert F\rvert}{m\lambda})$	$(m+3)\lambda$	$\frac{\lvert F\rvert}{m}$	$\ell + m$	$\ell + m$	3	0	$\ell + m$	$m\ell + 2\ell + 2m$	1	$m\ell$
[24]	$\frac{\lvert F\rvert}{m\lambda}$	0	0	$\lambda + k$	λ	0††	$\frac{\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{m\lambda}$	0	0	$\log(\frac{\lvert F\rvert}{m\lambda}) + m$	$\frac{\lvert F\rvert}{m\lambda}$	0	$\frac{\lvert F\rvert}{m\lambda}$
[53]	$\frac{\lvert F\rvert}{\lambda} + \frac{\lvert F\rvert}{m\lambda} + m$	$\frac{\lvert F\rvert}{\lambda}$	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	$m\ell$
[54]	$\frac{\lvert F\rvert}{\lambda} + \frac{2\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{\lambda}$	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	$m\ell$
[38]	$\frac{2\lvert F\rvert}{m\lambda}$	$\frac{\lvert F\rvert}{m\lambda}$	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	$m\ell + 2\ell + 2m$
[34, 35]-pri.†††	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	$m\ell + \ell$

† [42] is a journal version of [46], and the main scheme is almost the same as [46]. We now consider the one that divides each data block into m sectors.

†† In Hao *et al.*'s paper [24], the authentication tags are stored at both the client and the verifier side, rather than the server side.

††† The private key verifiable POR scheme of Shacham and Waters [34, 35]. Notice that the public key verifiable POS scheme of [34, 35] also appears in this table.

κ, k are system parameters, $|h|$ is the length of a hash output. $|F|$ 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.

preprocessing the data file. This makes these schemes impractical for file of medium or large size, especially limiting the usage on mobile device.

Although delegable POS has been studied by [29, 31, 38], unfortunately these works have the same drawback with public POS, i.e., the cost of tag generation is extremely high.

In other related work, Curtmola *et al.* [16] extended Ateniese *et al.*'s protocol [4] to a multiple-replica PDP scheme in which a client stores multiple replicas of a file across distributed storage servers and can verify the possession of each replica from the different servers. Bowers et al. [8] improved the POR model to distributed scenario with high availability and integrity, in which a set of distributed servers can prove to a client that a stored file is intact and retrievable. Similarly, Wang *et al.* [44, 45] and Zhu *et al.* [56] considered the auditing service for distributed data storage in cloud computing, in which the data file is encoded and divided into multiple blocks and store these blocks on different storage servers. In 2012, Wang *et al.* [40] put forward a new scenario where multiple users share data in the cloud and they proposed a privacy-preserving public auditing mechanism on the shared data. Later, they designed a public auditing scheme [41] for shared data with efficient user revocation.

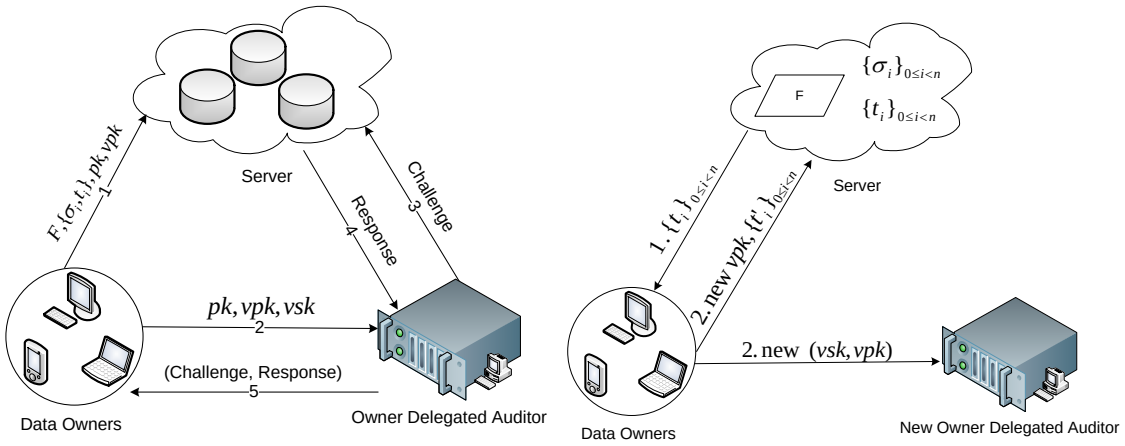
3 Formulation

We propose a formulation called "Delegatable Proofs of Storage" scheme (DPOS for short), based on existing POR [26, 34] and PDP [4] formulations. We provide the system model in Section 3.1 and the trust model in Section 3.2. We will defer the security definition to Section 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) The framework of DPOS (b) Updating Verification key (vpk, vsk)

Fig. 1. Illustration of system model of DPOS.

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

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,$\langle \mathsf{P}, \mathsf{V} \rangle$, 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* $\mathsf{KeyGen}(1^\lambda)$ *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* $\mathtt{Param}_F$*. At the end of setup phase, the data owner sends the file* F*, all authentication tags* $\{(\sigma_i, t_i)\}$*, file parameter* $\mathtt{Param}_F$*, 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* $\mathtt{Param}_F$ *to the ODA. After that, the data owner may keep only keys* (pk, sk, vpk, vsk) *and file parameter* $\mathtt{Param}_F$

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 (*`Context`*,* `Evidence`*) pairs, and allow data owner to fetch and verify these pairs 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* $(1, 1)$*.*

Application of DPOS **System: A Full Story.** In a real world application, with our scheme (and some other tools), data owner can delegate auditing task to ODA and ensure that, ODA performs the verification task at the right time and with the right challenge (e.g. the challenge message is really randomly chosen).

We require three additional building blocks: (1) error correcting code (e.g Reed-Solomon code [28]), which can correct errors in data blocks up to a certain amount; (2) timed-release encryption (e.g. [10,12]), which can encrypt a message w.r.t. a future time point t, such that only after time t, the holder of ciphertext can decrypt it successfully; (3) `ReceiveServer` (e.g. a trusted email server) with persistent buffer storage and synchronized time clock, which can receive a message via Internet and record the message together with the message-arrival-time.

Now let us put all pieces together: During the setup phase, the data owner runs the KeyGen algorithm to generate master key pair (sk, pk) and verification key pair (vsk, vpk), where sk will be kept private by data owner, vsk will be kept private by the ODA, and (pk, vpk) will be publicly available. To upload a file F to a cloud storage server, the data owner will divide file F into equal-length blocks, and apply error correcting code on each block independently. Then each erasure encoded block will be divided into equal-length slices, and all slices will be permutated randomly across different blocks. For each newly formed data block denoted as $\vec{\boldsymbol{F}}_i$, the data owner applies Tag algorithm to generate authentication tag (σ_i, t_i) using key (sk, vsk, pk). Then all data blocks $\{\vec{\boldsymbol{F}}_i\}$ together with their authentication tags $\{\sigma_i, t_i\}$ are uploaded to the cloud storage server. Optionally, the data owner may choose to delete the local copy of data file and tags, in order to save local storage space. To delegate the auditing task over file F, the data owner chooses properly N future time points $(\mathtt{t}_0, \mathtt{t}_1, \ldots, \mathtt{t}_{N-1})$ and N random seeds $(\mathfrak{S}_0, \ldots, \mathfrak{S}_{N-1})$, where $\mathtt{t}_0 < \mathtt{t}_1 < \ldots < \mathtt{t}_{N-1}$. Data owner encrypts the seed $\mathfrak{S}_i$ w.r.t time point $\mathtt{t}_i$, to obtain ciphertext $\mathtt{CText}_i$. Data owner delegates auditing task to the ODA by sending all timed-release ciphertexts $\{\mathtt{CText}_i\}$ and time points $\mathtt{t}_i$'s to the ODA at once. The security property of timed-release encryption will ensure that, only after time point $\mathtt{t}_i$, the ODA is able to decrypt the ciphertext $\mathtt{CText}_i$ and thus recover the seeds $\mathfrak{S}_i$. ODA then derives a challenge from seed $\mathfrak{S}_i$ and starts an interactive proof session with cloud storage server. After receiving response from server, ODA verifies whether the response is valid using verification secret key vsk and sends her decision bit and challenge-response pair to `ReceiveServer`, where `ReceiveServer` will keep the received challenge-response pairs together with message-arrival-time $\mathtt{t}'_i$ in its persistent storage device. Data owner may download all outstanding challenge-response pairs from `ReceiveServer`, at any time he/she likes. Data owner can verify the response using algorithm OwnerVerify and check whether the difference $(\mathtt{t}'_i - \mathtt{t}_i)$ is within a reasonable range. Optionally, data owner can verify multiple challenge-response pairs at once using batch-verification, to reduce computation time. If ODA's decision bit does not match data owner's decision bit or the timing check fails, then data owner can switch to another auditor and runs UpdVK to update the verification key pair and authentication tags $\{t_i\}$. In addition, the data owner may also add an extra auditor, and generates a pair of new verification keys and a set of new authentication tags $\{t'_i\}$ by running UpdVK.

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.

We assume that each communication party is authenticated and all communication data among the data owner, the cloud storage server and ODA is via some secure channel (i.e. channel privacy and integrity are protected). Privacy-preserving against auditor and framing attack among these three parties could be dealt with existing techniques and is out of scope of this paper.

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 $\mathbb{G}$. A bilinear map is a map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties:

(1) Bilinearity: $e(u^a, v^b) = e(u, v)^{ab}$ for all $u, v \in \mathbb{G}$ and $a, b \in \mathbb{Z}_p$.
(2) Non-degeneracy: If g is a generator of $\mathbb{G}$, then $e(g, g)$ is a generator of $\mathbb{G}_T$, i.e. $e(g, g) \neq 1$.
(3) Computable: There exists an efficient algorithm to compute $e(u, v)$ for all $u, v \in \mathbb{G}$.

In the rest of this paper, the term "bilinear map" will refer to the non-degenerate and efficiently computable bilinear map only.

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, $\langle \mathsf{P}, \mathsf{V} \rangle$, $\mathsf{OwnerVerify}$) as below, and these algorithms will run in the way as specified in Definition 2 (on page 6). 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 $\mathtt{id}_F \in \{0,1\}^\lambda$. Define a customized[3] pseudorandom function w.r.t. the file F: $\mathcal{R}_s(i) = \mathcal{PRF}_s(\mathtt{id}_F, i)$.

[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.

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

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

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.

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

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

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

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

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

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

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

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

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

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

Parse $\texttt{Context}$ as $(\xi, \{(i, w_i) : i \in \mathbf{C}\})$ and parse $\texttt{Evidence}$ as $(z, \phi_\alpha, \bar{\sigma})$. Verifier will set $b_0 := \texttt{1}$ if the following equality hold; otherwise set $b_0 := \texttt{0}$.

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

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

4.2.1 Batch Verification Alternatively, the data owner can also perform batch verification on a set of challenge-response pairs $\{(\texttt{Context}^{(\iota)}, \texttt{Evidence}^{(\iota)})\}_{\iota \in [1,n]}$ as below to achieve even lower amortized complexity: Choose weight value $\mu_\iota \in_R \mathbb{Z}_p^*$ for each $\iota \in [1, n]$ and verify the following equality

$$\left(\prod_{\iota \in [1,n]} e(\phi_\alpha^{(\iota)}, g^\alpha / g^{\xi^{(\iota)}})^{\mu_\iota} \quad \cdot \quad e(g, g)^{\sum\limits_{\iota \in [1,n]} \mu_\iota \cdot z^{(\iota)}}\right)^\alpha \stackrel{?}{=}$$

$$e(g,\ g^{\sum\limits_{\iota \in [1,n]} \mu_\iota \cdot \bar{\sigma}^{(\iota)}}) \cdot e(g, g)^{\left(-\sum\limits_{\iota \in [1,n]} \left(\mu_\iota \cdot \sum\limits_{i \in \mathbf{C}^{(\iota)}} w_i \mathcal{R}_{s_0}(i)\right)\right)} \tag{12}$$

4.2.2 Completeness We prove in Appendix A, that if all parties are honest and data is intact, then equalities in Eq (9), (10) and (11) will hold for certain. In addition, Eq 11 implies Eq 12 directly.

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.* [19]'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 [22]'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 2.5GHz Intel Core 2 Duo mobile CPU (model T9300, released in 2008). Our test files are randomly generated and of size from 128MB to 1GB. We achieve a throughput of data preprocessing at speed slightly larger than 10 megabytes per second, with $\lambda = 1024$. Detailed experiment data will be provided in the full paper.

In contrast, Atenesis *et al.* [3, 4] achieves throughput of data preprocessing at speed 0.05 megabytes per second with a 3.0GHz desktop CPU [4]. Wang *et al.* [42] achieves throughput of data pre-processing at speed 9.0KB/s and 17.2KB/s with an Intel Core 2 1.86GHz workstation CPU, when a data block is a

[4] Goldreich [22]'s (δ, ϵ)-hitter guarantees that, for any subset $W \subset [0, n-1]$ with size $|W| \geq (1-\delta)n$, $\mathsf{Pr}[\mathbf{C} \cap W \neq \emptyset] \geq 1 - \epsilon$. Readers may refer to [19] for more details.

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

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

vector of dimension $m = 1$ and $m = 10$, respectively. According to the pre-processing complexity of [42] shown in Table 1, the theoretical optimal throughput speed of [42] 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.* [42], using a single CPU core. We remark that, all of these schemes (ours and [3, 4, 42]) 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 [26,34], 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:

- StoreQuery($\mathtt{F}$): 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 $\mathcal{A}$.
- VerifyQuery($\mathtt{id}_F$): Given a file identifier $\mathtt{id}_F$ chosen by $\mathcal{A}$, if $\mathtt{id}_F$ is not the (partial) output of some previous StoreQuery 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 $b \in \{\mathtt{1}, \mathtt{0}\}$. 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.
- RevokeVKQuery: 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 StoreQueries 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 [34, 35]. 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,

$$\mathsf{Pr}\left[\begin{array}{l|l} \langle \tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{V}(vsk, & \mathtt{Sampling} \\ vpk, pk, \mathtt{Param}_{F^*})\rangle = (\mathtt{1}, \ldots) & \mathtt{set\ is\ } \mathbf{C}^* \end{array}\right] \geq 1/\lambda^\tau. \tag{13}$$

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,

$$\mathsf{Pr}\left[\begin{array}{l} \mathcal{E}^{\tilde{\mathsf{P}}(\mathtt{Memo})}(vsk, vpk, pk, \mathtt{Param}_{F^*}, \mathtt{id}^*, \mathbf{C}^*) \\ = \{(i, \mathtt{F}_i) : i \in \mathbf{C}^*\} \end{array}\right] \geq 1 - negl(\lambda). \tag{14}$$

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.*

$$\mathsf{Pr}\left[\begin{array}{l|l} \mathsf{OwnerVerify}(sk, pk, \langle \tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{V}(vsk, & \mathtt{Sampling} \\ vpk, pk, \mathtt{Param}_{F^*})\rangle) = (\mathtt{1}, \ldots) & \mathtt{set\ is\ } \mathbf{C}^* \end{array}\right] \tag{15}$$

$$\geq 1/\lambda^\tau, \quad \textit{for some positive integer constant } \tau, \tag{16}$$

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

$$\mathsf{Pr}\left[\begin{array}{l} \mathcal{E}^{\tilde{\mathsf{P}}(\mathtt{Memo}), \mathsf{OwnerVerify}(sk, \ldots)}(vsk, vpk, pk, \mathtt{Param}_{F^*}, \mathtt{id}^*, \mathbf{C}^*) \\ = \{(i, \mathtt{F}_i) : i \in \mathbf{C}^*\} \end{array}\right] \tag{17}$$

$$\geq 1 - negl(\lambda) \tag{18}$$

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 [34, 35], 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 easy.

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

5.2 Security Claim

Definition 6 (m-Bilinear Strong Diffie-Hellman Assumption) *Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ 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 $\mathbf{T} = (g, g^{\varsigma}, g^{\varsigma^2} \ldots, g^{\varsigma^m}) \in \mathbb{G}^{m+1}$, for any PPT adversary $\mathcal{A}$, the following probability is negligible*

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

Theorem 1 *Suppose m-BSDH Assumption hold, and* PRF *is a secure pseudorandom function. The* DPOS *scheme constructed in Section 4 is sound w.r.t. auditor, according to Definition 4.*

Theorem 2 *Suppose m-BSDH Assumption hold, and* PRF *is a secure pseudorandom function. The* DPOS *scheme constructed in Section 4 is sound w.r.t. data owner, according to Definition 5.*

5.3 Proof of Theorem 1—Soundness w.r.t Auditor

5.3.1 Unforgeability of Our Authentication Tag under Verification Private Key

Game 1 The first game is the same as soundness security game $\mathsf{Game}_{\mathtt{sound}}$, except that the pseudorandom function $\mathcal{R}_{s_0}$ outputs true randomness. Precisely, for each given seed s_0, the function $\mathcal{R}$ is evaluated in the following way:

1. The challenger keeps a table to store all previous encountered input-output pairs $(v, \mathcal{R}_{s_0}(v))$.
2. Given an input v, the challenger lookups the table for v, if there exists an entry (v, u), then return u as output. Otherwise, choose u at random from the range of $\mathcal{R}$, insert $(v, \mathcal{R}_{s_0}(v) := u)$ into the table and return u as output.

Game 2 The second game is the same as **Game 1**, except that the pseudorandom function $\mathcal{R}_{s_1}$ with seed s_1 outputs true randomness. The details are similar as in **Game 1**.

For ease of exposition, we clarify two related but distinct concepts: valid proof and genuine proof (or value).

Genuine Proof A proof is *genuine*, if it is the same as the one generated by an honest (deterministic) prover on the same query and the same context ().

Valid Proof A proof is *valid*, if it is accepted by the honest verifier.

In our scheme, for each query and given context, we will show that it is hard to find two distinct proofs (or responses) $(z, \ldots)$ and $(z', \ldots)$, which are both accepted by ODA and $z \neq z' \mod p$.

Lemma 3 *Suppose m-BSDH Assumption holds. In* **Game 2**, *any PPT adversary is unable to find two* distinct *tuples $T_0 = (z, \phi_\alpha;\ \bar{\sigma}, \bar{t}, \psi_\alpha, \psi_\beta)$ and $T_1 = (z', \phi'_\alpha;\ \bar{\sigma}, \bar{t}, \psi_\alpha, \psi_\beta)$, such that both are accepted by ODA w.r.t the same challenge and $z \neq z' \mod p$.*

$$\mathsf{Pr}\left[\begin{array}{c} T_0 \text{ and } T_1 \text{ are both accepted by ODA and } T_0[0] \neq T_1[0] \text{ and} \\ T_0[2,3,4,5] = T_1[2,3,4,5], \text{ where } (T_0,T_1) = \mathcal{A}^{\mathbf{Game\ 2}} \end{array}\right] \tag{19}$$

$$\leq \mathsf{Pr}\left[\mathcal{B} \text{ solves } m\text{-BSDH } \textit{Problem}\right] \tag{20}$$

(Proof of Lemma 3 is given in Appendix B.2)

Lemma 4 *Suppose m-BSDH Assumption holds. In* **Game 2**, *any PPT adversary is unable to find two* distinct *tuples $T_0 = (z, \phi_\alpha;\ \bar{\sigma}, \bar{t}, \psi_\alpha, \psi_\beta)$ and $T_1 = (z', \phi'_\alpha;\ \bar{\sigma}', \bar{t}', \psi'_\alpha, \psi'_\beta)$, such that both T_0 and T_1 are accepted by ODA w.r.t the same challenge, $z \neq z'$ and $(\bar{\sigma}, \bar{t}, \psi_\alpha, \psi_\beta) \neq (\bar{\sigma}', \bar{t}', \psi'_\alpha, \psi'_\beta)$. Precisely, for any PPT adversary $\mathcal{A}$ in* **Game 2**, *there exists a PPT algorithm $\mathcal{D}$, such that*

$$\mathsf{Pr}\left[\begin{array}{c} T_0 \text{ and } T_1 \text{ are both accepted by ODA and } T_0[0] \neq T_1[0] \text{ and} \\ T_0[2,3,4,5] \neq T_1[2,3,4,5], \text{ where } (T_0,T_1) = \mathcal{A}^{\mathbf{Game\ 2}} \end{array}\right] \tag{21}$$

$$\leq \mathsf{Pr}\left[\mathcal{D} \text{ solves } m\text{-BSDH } \textit{Problem}\right] \tag{22}$$

(Proof of Lemma 4 is given in Appendix B.3).

Algorithm 1: SIMULATOR

Input: $(\tilde{\mathsf{P}}, \mathtt{Memo}, \mathsf{V}, pk, vsk, vpk, \mathbf{C}^*, N_1, N_2, N_3)$, where $\mathbf{C}^*$ is a set of block indices, and N_1, N_2, N_3 are integers

Output: A set $\mathbf{S}$, representing transcript of an execution of this algorithm

1 Initiate the set $\mathbf{S} \leftarrow \emptyset$ as empty set;
2 **foreach** *i from* 1 *to* N_1 **do**
3 Choose weight vector $W_i = (w_{i,1}, \ldots, w_{i,c})$ at random as in Step V1 of algorithm V ;
4 **foreach** *j from* 1 *to* N_2 **do**
5 Choose value $\xi_{i,j}$ at random as in Step V1 of algorithm V;
6 **foreach** *k from* 1 *to* N_3 **do**
7 Send $(\mathbf{C}^*, W_i, \xi_{i,j})$ as challenge query to algorithm $\tilde{\mathsf{P}}(\mathtt{Memo})$ and get response $(z_{i,j,k}, \ldots)$;
8 Send this response to algorithm $\mathsf{V}(vsk, vpk, pk, \mathtt{Param}_{F^*})$, which outputs decision bit $b_{i,j,k}$ at Step V2;
9 Update $\mathbf{S} \leftarrow \mathbf{S} \cup \{(W_i,\ \xi_{i,j},\ z_{i,j,k},\ b_{i,j,k})\}$;
10 *Note: Different instances of revocation of* $\tilde{\mathsf{P}}(\mathtt{Memo})$ *are independent.*
11 **end**
12 **end**
13 **end**

5.3.2 Construction of Knowledge Extractor We explicitly construct the knowledge extractor algorithm using two subroutines, named SIMULATOR Algorithm 1 (on page 14) and SOLVER Algorithm 2 (on page 15).

Before introducing SOLVER Algorithm 2 on page 15, let us define sets $\mathbf{S}_{\boldsymbol{\xi}}$, $\mathbf{S}_{\mathcal{W}}$ based on set $\mathbf{S}$ as below:

$$\mathbf{S}_{\boldsymbol{\xi}} = \left\{ (W, \xi) : \exists \text{witness set } \mathbb{W} \text{ s.t. } {}^{|\mathbb{W}| \geq 1 \text{ and } \forall z \in \mathbb{W},}_{(W,\xi,z,1) \in \mathbf{S}} \right\}; \tag{23}$$

$$\mathbf{S}_{\mathcal{W}} = \left\{ W : \exists \text{witness set } \mathbb{W} \text{ s.t. } {}^{|\mathbb{W}| \geq m \text{ and } \forall \xi \in \mathbb{W},}_{(W,\xi) \in \mathbf{S}_{\boldsymbol{\xi}}} \right\}. \tag{24}$$

5.3.3 Analysis of Knowledge Extractor

Lemma 5 *If* $|\mathbf{S}_{\mathcal{W}}| \geq c$*, then the algorithm* SOLVER *can output* $\{\vec{\boldsymbol{F}}_i\}_{i \in \mathbf{C}^*}$ *correctly, with o.h.p.*

Lemma 5 directly follows from the unforgeability of our authentication tag (i.e. Lemma 3 and Lemma 4) and Claim 1 and Claim 2, where the two claims are as below.

Claim 1 *If* $|\mathbf{S}_{\mathcal{W}}| \geq c$*, then procedure* SOLVER *will abort (i.e. output* `Failure`*) with negligible probability (in* λ*). (Proof is given in Appendix B.4.1)*

Claim 2 *If for every tuple* $(W, \xi, z, 1) \in \mathbf{S}$*, the value* z *is genuine (i.e. the same as computed by an honest prover w.r.t. the same query and context* $(\mathbf{C}^*, W, \xi)$*), then the output of* SOLVER *Algorithm 2 at Line 36 is correct, i.e.* $\{A_i^*\} = \{\vec{\boldsymbol{F}}_i\}$*. (Proof is given in Appendix B.4.2)*

Given the challenge $(\mathbf{C}^*, W = \{w_i : i \in \mathbf{C}^*\}, \xi)$ in Step V1 of verifier algorithm V, and random coin $\mathfrak{C}_{\mathcal{A}}$ of adversary $\mathcal{A}$ in Step P1, the response of adversary in Step P1 will be deterministic and so is the output of the verifier in Step V2 of algorithm V. Let notations $\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}$ denote the random variables, respectively.

Let function $f_{\mathbf{C}^*}(W, \xi, \mathfrak{C}_{\mathcal{A}})$ denote the boolean output of verifier in Step V2.

Definition 7 *For any* $\delta \in (0, 1)$*,*

- W *is* δ*-good, if* $\Pr_{\boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(W, \quad \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}) = 1] \geq \delta$*;*
- ξ *is* δ*-good w.r.t* W*, if* $\Pr_{\mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(W, \xi, \quad \mathfrak{C}_{\mathcal{A}}) = 1] \geq \delta$*.*

Lemma 6 *Let parameters* $N_1 = \lceil 2\lambda\epsilon^{-1} \rceil, N_2 = N_3 = \lceil 4\lambda\epsilon^{-1} \rceil$ *in* SIMULATOR *Algorithm 1. Recall that the set* $\mathbf{S}_{\mathcal{W}}$ *is constructed at Line 5 of* SOLVER *Algorithm 5. If adversary* $\mathcal{A}$ *can pass auditor's verification with probability* $\geq \epsilon$*, i.e.* $\Pr_{\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}) = 1] \geq \epsilon$*, then* $|\mathbf{S}_{\mathcal{W}}| \geq c$*.*

Lemma 6 directly follows from Claim 3 and Claim 4 as below.

Algorithm 2: SOLVER

Input: Set $\mathbf{S}$, which is the output of SIMULATOR.
Output: A guess of data blocks $\{\vec{\boldsymbol{F}}_{i_\iota} : \iota \in [1,c]\}$ where $\{i_1, \ldots, i_c\} = \mathbf{C}^*$.

Find set $\mathbf{S}_{\boldsymbol{\xi}}$, by filtering $\mathbf{S}$;
foreach $(W, \xi) \in \mathbf{S}_{\boldsymbol{\xi}}$ **do**
  find witness set $\mathbb{W}_{[W,\xi]} := \{z : (W, \xi, z, 1) \in \mathbf{S}\}$;
end
Find set $\mathbf{S}_{\mathcal{W}}$, by sorting and filtering $\mathbf{S}_{\boldsymbol{\xi}}$;
foreach $W \in \mathbf{S}_{\mathcal{W}}$ **do**
  find witness set $\mathbb{W}_{[W]} := \{\xi : (W, \xi) \in \mathbf{S}_{\boldsymbol{\xi}}\}$;
end
if $|\mathbf{S}_{\mathcal{W}}| < |\mathbf{C}^*|$ **then**
  Abort and output `Failure1`;
else
  Find subset $\{W_1^*, \ldots, W_c^*\} \subset \mathbf{S}_{\mathcal{W}}$;
end
foreach *i from* 1 *upto* c **do**
  if $|\mathbb{W}_{[W_i^*]}| < m$ **then**
    Abort and output `Failure2`;
  else
    Find subset $\{\xi_{i,1}^*, \ldots, \xi_{i,m}^*\} \subset \mathbb{W}_{[W_i^*]}$;
  end
  foreach *j from* 1 *to* m **do**
    if $|\mathbb{W}_{[W_i^*,\ \xi_{i,j}^*]}| < 1$ **then**
      Abort and output `Failure3`;
    else
      Choose $z_{i,j}^* \in_R \mathbb{W}_{[W_i^*,\ \xi_{i,j}^*]}$;
    end
  end
  Find the polynomial $\mathsf{Poly}_{\vec{\boldsymbol{F}}_i}$ of degree $(m-1)$ passing all of m points $\{(\xi_{i,j}^*,\ z_{i,j}^*)\}_{j=1}^m$;
  if *number of polynomials found* $\neq 1$ **then**
    Abort and output `Failure4` ;
  end
end
Solve the linear system $(A_1^\top, \ldots, A_c^\top) \times (W_1^{*\top}, \ldots, W_c^{*\top}) = (\vec{\boldsymbol{\mathcal{F}}}_1^\top, \ldots, \vec{\boldsymbol{\mathcal{F}}}_c^\top)$, where A_i's are unknown, and W_i^* and $\vec{\boldsymbol{\mathcal{F}}}_i$ are given, $i \in [1,c]$. *Note:* $W_i^* = (w_i, w_i^2, \ldots, w_i^c)$ *and the coefficient matrix* $(W_1^{*\top}, \ldots, W_c^{*\top})$ *is a simple variant of Vandermonde matrix*;
if *number of solutions of above linear system* $\neq 1$ **then**
  Abort and output `Failure5` ;
else
  return *the unique solution* $\{A_i^*\}_{i=1}^c$;
end

Claim 3 *If* $\Pr_{\mathcal{W},\boldsymbol{\xi},\mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}) = 1] \geq \epsilon$, *there exist at least* c *distinct* $\epsilon/2$*-good* W_i*'s among all of* N_1 *instances of* W_i*'s generated at Line 3 of Algorithm 1. (Proof is given in Appendix B.5.1)*

Claim 4 *If* W_i *is* $\epsilon/2$*-good, then* $W_i \in \mathbf{S}_{\mathcal{W}}$ *with o.h.p. (Proof is given in Appendix B.5.2)*

Proof (Proof of Theorem 1). First of all, the complexity of the extractor is $O(N_1 \cdot N_2 \cdot N_3 \cdot (T_{\mathsf{P},\mathsf{V}} + \log(N_1 \cdot N_2 \cdot N_3)))$ $= O(\frac{32\lambda^3}{\epsilon^3} \times \left(T_{\mathsf{P},\mathsf{V}} + \log \frac{32\lambda^3}{\epsilon^3}\right))$, which is polynomial in λ, if $\epsilon > 1/poly(\lambda)$ is noticeable, where $T_{\mathsf{P},\mathsf{V}}$ denotes the time complexity of the interactive algorithm $\langle \mathsf{P}, \mathsf{V} \rangle$ and integer parameters N_1, N_2, N_3 are specified as in Lemma 6.

Since $\mathcal{R}$ is a secure pseudorandom function, the soundness security game $\mathtt{Game}_{\mathtt{sound}}$ and **Game 1** are computationally indistinguishable. So are **Game 1** and **Game 2**. Therefore, Lemma 3 and Lemma 4 also hold in $\mathtt{Game}_{\mathtt{sound}}$ with negligible difference in success probability. Consequently, the premise (i.e.

if-part) of Claim 2 holds with o.h.p. Claim 1 and Claim 2 together imply Lemma 5. Lemma 6 ensures that the premise of Lemma 5 will hold. Therefore, the conclusion (then-part) of Lemma 5 holds, that is, the extractor algorithm constructed previous can output the selected file blocks with overwhelming high probability as desired.

$$\left.\begin{array}{r}\left.\begin{array}{r}\text{Claim 1}\\ \text{Claim 2}\\ \text{Lemma 3}\\ \text{Lemma 4}\end{array}\right\}\Rightarrow \text{Lemma 5}\\ \left.\begin{array}{r}\text{Claim 3}\\ \text{Claim 4}\end{array}\right\}\Rightarrow \text{Lemma 6}\end{array}\right\}\Rightarrow \text{Theorem 1} \tag{25}$$

□

5.4 Proof of Theorem 2—Soundness w.r.t Owner

5.4.1 Unforgeability of Our Authentication Tag under Master Private Key **Game 1'** The same as $\mathsf{Game2}_{\mathtt{Sound}}$, except that pseudo random function $\mathcal{R}_{s_0}$ is replaced by true randomness. The detail is similar to definition of **Game 1**.

Game 2' The same as **Game 1'**, except that pseudo random function $\mathcal{R}_{s_1}$ is replaced by true randomness.

Lemma 7 *Suppose m-BSDH Assumption holds. In* **Game 2'***, any PPT adversary is unable to find two* distinct *tuples* $T_0 = (z, \phi_\alpha, \bar{\sigma})$ *and* $T_1 = (z', \phi'_\alpha, \bar{\sigma}')$*, such that both* T_0 *and* T_1 *are accepted by data owner w.r.t the same challenge and* $T_0 \neq T_1$*. Precisely, for any PPT adversary* $\mathcal{A}$ *in* **Game 2***, there exists a PPT algorithm* $\mathcal{B}$*, such that*

$$\mathsf{Pr}\left[\begin{array}{c}T_0\neq T_1 \text{ are both accepted by owner}\\ \text{where } (T_0,T_1)=\mathcal{A}^{\textbf{Game 2'}}\end{array}\right] \tag{26}$$

$$\leq \quad \mathsf{Pr}\left[\mathcal{B} \text{ solves } m\text{-BSDH } \textit{Problem}\right] \tag{27}$$

(Proof is given in Appendix C.1)

5.4.2 Construction and Analysis of Knowledge Extractor In this case, the construction and analysis of knowledge extractor algorithm is identical as for auditor case, except that: *at Line 9 of* Simulator *Algorithm 1, the decision bit* $b_{i,j,k}$ *should represent data owner's first decision bit (i.e* b_0 *— indicating whether data owner accepts storage server) upon response provided by* $\tilde{\mathsf{P}}(\mathtt{Memo})$, instead of auditor's decision bit.

The rest of proof for Theorem 2 is identical to proof for Theorem 1.

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. Aniket Kate, Gregory M. Zaverucha, I.G.: Constant-Size Commitments to Polynomials and Their Applications. In: ASIACRYPT'10. pp. 177–194
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. pp. 831–843. CCS '14 (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. on Info. and 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'07. pp. 598–609. ACM (2007)
5. Ateniese, G., Kamara, S., Katz, J.: Proofs of storage from homomorphic identification protocols. In: ASIACRYPT'09. LNCS, vol. 5912, pp. 319–333. Springer (2009)
6. Ateniese, G., Pietro, R.D., Mancini, L.V., Tsudik, G.: Scalable and efficient provable data possession. In: SecureComm'08. pp. 9:1–9:10. ACM (2008)
7. Boneh, D., Lynn, B., Shacham, H.: Short signatures from the weil pairing. Journal of 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'09. pp. 187–198. ACM (2009)
9. Bowers, K.D., Juels, A., Oprea, A.: Proofs of retrievability: Theory and implementation. In: CCSW'09. pp. 43–54. ACM (2009)
10. Casassa Mont, Marco; Harrison, K.S.M.: The HP Time Vault Service: Innovating the way confidential information is disclosed, at the right time, `http://www.hpl.hp.com/techreports/2002/HPL-2002-243.pdf`
11. Cash, D., Küpçü, A., Wichs, D.: Dynamic proofs of retrievability via oblivious RAM. In: EUROCRYPT'13. LNCS, vol. 7881, pp. 279–295. Springer (2013)
12. Chan, A.C., Blake, I.F.: Scalable, Server-Passive, User-Anonymous Timed Release Cryptography. In: 25th International Conference on Distributed Computing Systems (ICDCS 2005). pp. 504–513 (2005)
13. Chang, E.C., Xu, J.: Remote integrity check with dishonest storage server. In: ESORICS'08. LNCS, vol. 5283, pp. 223–237. Springer (2008)
14. Chaowen Guan, Kui Ren, F.Z.F.K., Yu, J.: A Symmetric-Key Based Proofs of Retrievability Supporting Public Verification. In: (To appear in) ESORICS 2015. `www.fkerschbaum.org/esorics15b.pdf`
15. Chernoff, H.: A Measure of Asymptotic Efficiency for Tests of a Hypothesis Based on the sum of Observations. Annals of Mathematical Statistics 23(4), 493507 (1952)
16. Curtmola, R., Khan, O., Burns, R., Ateniese, G.: MR-PDP: Multiple-replica provable data possession. In: ICDCS'08. pp. 411–420. IEEE (2008)
17. Daniel Apon, Yan Huang, J.K.A.J.M.: Implementing Cryptographic Program Obfuscation. Cryptology ePrint Archive, Report 2014/779 (2014), `http://eprint.iacr.org/`
18. Deswarte, Y., Quisquater, J.J., Saïdane, A.: Remote integrity checking: How to trust files stored on untrusted servers. In: Integrity and Internal Control in Information Systems VI. LNCS, vol. 140, pp. 1–11. Springer (2004)
19. Dodis, Y., Vadhan, S., Wichs, D.: Proofs of Retrievability via Hardness Amplification. In: Proceedings of TCC'09, LNCS, vol. 5444, pp. 109–127. Springer (2009)
20. Erway, C., Küpçü, A., Papamanthou, C., Tamassia, R.: Dynamic provable data possession. In: ACM CCS'09. pp. 213–222. ACM (2009)
21. 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. pp. 40–49. FOCS '13 (2013)
22. Goldreich, O.: A Sample of Samplers - A Computational Perspective on Sampling (survey). Electronic Colloquium on Computational Complexity (ECCC) 4(20) (1997)
23. Goldreich, O.: Foundations of Cryptography: Volume 1, Basic Tools. Cambridge University Press, New York, NY, USA (2006)
24. Hao, Z., Zhong, S., Yu, N.: A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability. TKDE'11 23(9), 1432–1437 (2011)
25. Hohenberger, S., Lysyanskaya, A.: How to Securely Outsource Cryptographic Computations. In: Proceedings of the Second International Conference on Theory of Cryptography. pp. 264–282. TCC'05 (2005)
26. Juels, A., Burton S. Kaliski, J.: PORs: Proofs of retrievability for large files. In: ACM CCS'07. pp. 584–597. ACM (2007)
27. Naor, M., Rothblum, G.N.: The complexity of online memory checking. Journal of the ACM 56(1) (2009)
28. Reed, I.S., Solomon, G.: Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics 8(2), 300–304 (1960)
29. Ren, Y., Shen, J., Wang, J., Fang, L.: Outsourced data tagging via authority and delegable auditing for cloud storage. In: 49*th* Annual IEEE International Carnahan Conference on Security Technology. pp. 131–134. ICCST'15, IEEE (2015)
30. Ren, Y., Shen, J., Wang, J., Han, J., Lee, S.: Mutual verifiable provable data auditing in public cloud storage. Journal of Internet Technology 16(2), 317–324 (2015)
31. Ren, Y., Xu, J., Wang, J., Kim, J.U.: Designated-verifier provable data possession in public cloud storage. International Journal of Security and its Applications 7(6), 11–20 (2013)

32. Schwarz, T.J.E., Miller, E.L.: Store, forget, and check: Using algebraic signatures to check remotely administered storage. In: ICDCS'06. IEEE (2006)
33. Sebé, F., Domingo-Ferrer, J., Martínez-Ballesté, A., Deswarte, Y., Quisquater, J.J.: Efficient remote data possession checking in critical information infrastructures. TKDE'08 20(8), 1034–1038 (2008)
34. Shacham, H., Waters, B.: Compact proofs of retrievability. In: ASIACRYPT'08. LNCS, vol. 5350, pp. 90–107. Springer (2008)
35. Shacham, H., Waters, B.: Compact proofs of retrievability. Journal of Cryptology 26(3), 442–483 (2013)
36. Shah, M.A., Baker, M., Mogul, J.C., Swaminathan, R.: Auditing to keep online storage services honest. In: HotOS'07. USENIX Association (2007)
37. 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
38. Shen, S.T., Tzeng, W.G.: Delegable provable data possession for remote data in the clouds. In: Proceedings of the 13th International Conference on Information and Communications Security. pp. 93–111. ICICS'11, Springer-Verlag, Berlin, Heidelberg (2011)
39. Shi, E., Stefanov, E., Papamanthou, C.: Practical dynamic proofs of retrievability. In: ACM CCS'13. pp. 325–336. ACM (2013)
40. 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)
41. Wang, B., Li, B., Li, H.: Public auditing for shared data with efficient user revocation in the cloud. In: INFOCOM'13. pp. 2904–2912. IEEE (2013)
42. Wang, C., Chow, S.S., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for secure cloud storage. IEEE Tran. on Computers 62(2), 362–375 (2013)
43. Wang, C., Ren, K., Lou, W., Li, J.: Toward publicly auditable secure cloud data storage services. IEEE Network Magazine 24(4), 19–24 (2010)
44. Wang, C., Wang, Q., Ren, K., Cao, N., Lou, W.: Towards secure and dependable storate services in cloud computing. IEEE Transactions on Services Computing 5(2), 220–232 (2012)
45. Wang, C., Wang, Q., Ren, K., Lou, W.: Ensuring data storage security in cloud computing. In: Proceedings of IWQoS'09. pp. 1–9. IEEE (2009)
46. Wang, C., Wang, Q., Ren, K., Lou, W.: Privacy-preserving public auditing for data storage security in cloud computing. In: INFOCOM'10. pp. 525–533. IEEE (2010)
47. Wang, Q., Wang, C., Li, J., Ren, K., Lou, W.: Enabling public verifiability and data dynamics for storage security in cloud computing. In: ESORICS'09. LNCS, vol. 5789, pp. 355–370. Springer (2009)
48. Wang, Q., Wang, C., Ren, K., Lou, W., Li, J.: Enabling public auditability and data dynamics for storage security in cloud computing. TPDS'11 22(5), 847–859 (2011)
49. Xiaofeng Chen, Jin Li, J.M.Q.T.W.L.: New Algorithms for Secure Outsourcing of Modular Exponentiations. In: Proceedings of 17th European Symposium on Research in Computer Security. pp. 541–556. ESORICS 2012
50. Xu, J., Chang, E.C.: Towards Efficient Proofs of Retrievability. In: AsiaCCS '12: ACM Symposium on Information, Computer and Communications Security (2012)
51. 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
52. Yang, K., Jia, X.: Data storage auditing service in cloud computing: challenges, methods and opportunities. World Wide Web 15(4), 409–428 (2012)
53. Yang, K., Jia, X.: An efficient and secure dynamic auditing protocol for data storage in cloud computing. TPDS'13 24(9), 1717–1726 (2013)
54. 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)
55. Zeng, K.: Publicly verifiable remote data integrity. In: ICICS'08. LNCS, vol. 5308, pp. 419–434. Springer (2008)
56. Zhu, Y., Hu, H., Ahn, G.J., Yu, M.: Cooperative provable data possession for integrity verification in multi-cloud storage. TPDS'12 23(12), 2231–2244 (2012)
57. 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'11. pp. 1550–1557. ACM (2011)

A Proof for Completeness

If all parties are honest and data are intact, we have

$$e(g,g)^{\alpha \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(\psi_\alpha, g^\alpha), \tag{28}$$

$$e(g,g)^{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(\phi_\alpha, g^\alpha / g^\xi) \cdot e(g,g)^z. \tag{29}$$

$$\forall x, \mathsf{Poly}_{\vec{\boldsymbol{v}}}(x) \equiv \frac{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(x) - \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi)}{x - \xi} \mod p \tag{30}$$

$$z = \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\xi) \mod p \tag{31}$$

$$e(g,g)^{\alpha \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(\psi_\alpha, g^\alpha), \tag{32}$$

$$e(g,g)^{\rho\beta \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\beta)} = e(\psi_\beta, g^\beta), \tag{33}$$

$$e(g,g)^{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(\phi_\alpha, g^\alpha / g^\xi) \cdot e(g,g)^z, \tag{34}$$

$$\left(g^{\bar{\sigma}} / Y_\sigma\right)^{r_\sigma^{-1}} = g^{\sum_{i \in \mathbf{C}} w_i \sigma_i}, \tag{35}$$

$$\left(g^{\bar{t}} / Y_t\right)^{r_t^{-1}} = g^{\sum_{i \in \mathbf{C}} w_i t_i}. \tag{36}$$

A.0.3 Correctness of Equation (9) By combining Eq (8)(30)(31)(6), we can compute the right hand side of Equation (9) as below:

$$\begin{aligned} \mathbf{RHS} &= e(g^{\mathsf{Poly}_{\vec{\boldsymbol{v}}}(\alpha)}, \quad g^{\alpha - \xi}) \cdot e(g,g)^z = e(g,g)^{\mathsf{Poly}_{\vec{\boldsymbol{v}}}(\alpha) \cdot (\alpha - \xi) + z} \\ &= e(g,g)^{\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(\psi_\alpha, g) = \mathbf{LHS}. \end{aligned} \tag{37}$$

A.0.4 Correctness of Equation (10) Substituting Eq (32) into Eq (10), we have left hand side of Eq (10):

$$\mathbf{LHS}^{\gamma^{-1}} \tag{38}$$

$$= \frac{e(g,g)^{\alpha \mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)}}{e(g, \ g^{\sum_{i \in \mathbf{C}} w_i \sigma_i})} \tag{39}$$

$$= \frac{e(g,g)^{\langle \vec{\boldsymbol{\alpha}}, \ \vec{\boldsymbol{\mathcal{F}}} \rangle}}{e\left(g, \quad g^{\sum_{i \in \mathbf{C}} w_i \left(\langle \vec{\boldsymbol{\alpha}}, \ \vec{\boldsymbol{F}}_i \rangle + \mathcal{R}_{s_0}(i)\right)}\right)} \tag{40}$$

$$= \frac{e(g,g)^{\langle \vec{\boldsymbol{\alpha}}, \ \vec{\boldsymbol{\mathcal{F}}} \rangle}}{e\left(g,g\right)^{\left(\left\langle \vec{\boldsymbol{\alpha}}, \ \sum_{i \in \mathbf{C}} w_i \vec{\boldsymbol{F}}_i \right\rangle + \sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)\right)}} \tag{41}$$

$$= \frac{e(g,g)^{\langle \vec{\boldsymbol{\alpha}}, \ \vec{\boldsymbol{\mathcal{F}}} \rangle}}{e\left(g,g\right)^{\left(\langle \vec{\boldsymbol{\alpha}}, \ \vec{\boldsymbol{\mathcal{F}}} \rangle + \sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)\right)}} \tag{42}$$

$$= \frac{1}{e(g,g)^{\sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)}}. \tag{43}$$

We can compute the right hand side of Eq (10):

$$\textbf{RHS} \tag{44}$$

$$= \frac{e(g,g)^{\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}}{e(g,\ \ g^{\sum\limits_{i\in\mathbf{C}} w_i t_i - \sum\limits_{i\in\mathbf{C}} w_i \mathcal{R}_{s_1}(i)})} \tag{45}$$

$$= \frac{e(g,g)^{\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}}{e(g,\ \ g^{\sum\limits_{i\in\mathbf{C}} w_i (t_i - \mathcal{R}_{s_1}(i))})} \tag{46}$$

$$= \frac{e(g,g)^{\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}}{e\left(g,\ \ g^{\sum\limits_{i\in\mathbf{C}} w_i \left(\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{F}}_i\rangle + \gamma\mathcal{R}_{s_0}(i)\right)}\right)} \tag{47}$$

$$= \frac{e(g,g)^{\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}}{e\left(g,g\right)^{\left(\rho\left\langle\vec{\boldsymbol{\beta}},\ \sum\limits_{i\in\mathbf{C}} w_i \vec{\boldsymbol{F}}_i\right\rangle + \sum\limits_{i\in\mathbf{C}} w_i\cdot\gamma\mathcal{R}_{s_0}(i)\right)}} \tag{48}$$

$$= \frac{e(g,g)^{\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}}{e\left(g,g\right)^{\left(\rho\langle\vec{\boldsymbol{\beta}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle + \sum\limits_{i\in\mathbf{C}} w_i\cdot\gamma\mathcal{R}_{s_0}(i)\right)}} \tag{49}$$

$$= \frac{1}{e(g,g)^{\sum\limits_{i\in\mathbf{C}} w_i\cdot\gamma\mathcal{R}_{s_0}(i)}} = \textbf{LHS}. \tag{50}$$

A.0.5 Correctness of Equation (11)

$$\textbf{LHS} = \left(e(g,g)^{\mathsf{Poly}_{\vec{v}}(\alpha)\cdot(\alpha-\xi)+z}\right)^{\alpha} = e(g,g)^{\alpha\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}}(\alpha)} = e(g,g)^{\langle\vec{\boldsymbol{\alpha}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle}. \tag{51}$$

$$\textbf{RHS} = e(g,\ g^{\sum\limits_{i\in\mathbf{C}} w_i\sigma_i}) \cdot e(g,\ g)^{\left(-\sum\limits_{i\in\mathbf{C}} w_i\mathcal{R}_{s_0}(i)\right)} \tag{52}$$

$$= e(g,g)^{\left(\sum\limits_{i\in\mathbf{C}} w_i(\sigma_i - \mathcal{R}_{s_0}(i))\right)} \tag{53}$$

$$= e(g,g)^{\left(\sum\limits_{i\in\mathbf{C}} w_i\langle\vec{\boldsymbol{\alpha}},\ \vec{\boldsymbol{F}}_i\rangle\right)} \tag{54}$$

$$= e(g,g)^{\left\langle\vec{\boldsymbol{\alpha}},\ \sum\limits_{i\in\mathbf{C}} w_i\vec{\boldsymbol{F}}_i\right\rangle} \tag{55}$$

$$= e(g,g)^{\langle\vec{\boldsymbol{\alpha}},\ \vec{\boldsymbol{\mathcal{F}}}\rangle} \tag{56}$$

$$= \textbf{LHS} \tag{57}$$

B Proof for Soundness w.r.t. Auditor

B.1 Simulate Game 2 using the input of m-BSDH Problem

Let bilinear map $e : \mathbb{G}\times\mathbb{G}\to\mathbb{G}_T$ and tuple $\mathbf{T} = (g, g^{\varsigma}, g^{\varsigma^2}, \ldots, g^{\varsigma^m}) \in \mathbb{G}^m$ be as stated in the m-BSDH Assumption. With information $\mathbf{T}$, we can simulate **Game 2** as below. Recall that the adversary $\mathcal{A}$ in this game is the dishonest prover (i.e. the cloud storage server).

Setup. Let $\alpha := \varsigma$, which is unknown. Define $\alpha_j := \alpha^j \mod p$ for each $j \in [1,m]$. Choose at random two group elements $\gamma, \rho \in_R \mathbb{Z}_p^*$, a vector $\vec{\boldsymbol{u}} = (u_1,\ldots,u_m) \in_R \mathbb{Z}_p^m$, and pseudorandom function seed s_1. For each $j \in [1,m]$, we can find values of g_j: $g_j := g^{\alpha_j} = g^{\alpha^j} = g^{\varsigma^j} \in \mathbb{G}$. Let $g_0 = g$ and $h_0 = g^{\rho}$. Implicitly define the unknown vector $\vec{\boldsymbol{\beta}}$ as $\rho\vec{\boldsymbol{\beta}} - \gamma\vec{\boldsymbol{\alpha}} = \vec{\boldsymbol{u}}$. For each j, compute $h_j := g_j^{\gamma}\cdot g^{u_j} = g^{\gamma\alpha_j + u_j} = g^{\rho\beta_j} = h_0^{\beta_j} \in \mathbb{G}$. The secret key is $sk = (\alpha, \vec{\boldsymbol{\beta}}, s_0)$, 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 $vpk = (h_0, h_1, \ldots, h_m)$, where the random seed s_0 for pseudorandom function $\mathcal{PRF}$ will be determined later. Send (pk, vpk) to the adversary.

Notice that, the simulator knows values of vsk, vpk and pk, but does not know values of $(\alpha, \vec{\boldsymbol{\beta}}, s_0)$.

Learning.

- STORE-QUERY(F): Given a data file F chosen by the adversary $\mathcal{A}$. The simulator simulates the algorithm Tag as below: For each $i \in [0, n-1]$, choose the authentication tag $\sigma_i \in_R \mathbb{Z}_p$ at random, which will implicitly define the values of the file-specific randomness $\mathcal{R}_{s_0}(i)$. The simulator is able to compute: $g^{\langle \vec{\alpha},\ \vec{F}_i \rangle} = \prod_{j=1}^{m} g_j^{\vec{F}_i[j-1]}$; $g^{\rho \langle \vec{\beta},\ \vec{F}_i \rangle} = \prod_{j=1}^{m} h_j^{\vec{F}_i[j-1]}$; $g^{\mathcal{R}_{s_0}(i)} = \frac{g^{\sigma_i}}{g^{\langle \vec{\alpha},\ \vec{F}_i \rangle}}$. The simulator can also compute $t_i := \gamma\sigma_i + \left\langle \vec{u},\ \vec{F}_i \right\rangle + \mathcal{R}_{s_1}(i) \mod p$, which is equal to $\gamma\left(\left\langle \vec{\alpha},\ \vec{F}_i \right\rangle + \mathcal{R}_{s_0}(i)\right) + \left\langle \rho\vec{\beta} - \gamma\vec{\alpha},\ \vec{F}_i \right\rangle + \mathcal{R}_{s_1}(i) = \rho\left\langle \vec{\beta},\ \vec{F}_i \right\rangle + \gamma\mathcal{R}_{s_0}(i) + \mathcal{R}_{s_1}(i)$ as desired. Send file F and authentication tags $\{(i, \sigma_i, t_i)\}$ to the adversary $\mathcal{A}$.
- VERIFY-QUERY: The simulator has full information of pk, vpk, and γ. Although the simulator does not know s_1, it knows values $g^{\mathcal{R}_{s_1}(i)}$ for each $i \in [0, n-1]$. The simulator can execute the verifier algorithm V in the proposed DPOS scheme exactly. So the simulator can find the exactly same decision bit $b \in \{\mathtt{1}, \mathtt{0}\}$ as in a real game.
  Next, the simulator tries to find the decision bit b_0 of the owner. Notice that Eq 11 in algorithm $\mathsf{OwnerVerify}$ can be written equivalently as below

$$e(\phi_\alpha, g^{\alpha^2}/g^{\alpha\xi}) \cdot e(g^\alpha, g)^z \cdot e(g,g)^{\sum_{i\in\mathbf{C}} w_i \mathcal{R}_{s_0}(i)} = e(g,g)^{\bar{\sigma}} \tag{58}$$

  Although s_0 is unknown, the simulator is still able to compute both sides of the above equation: $g^{\alpha^2} = g_2, g^{\alpha\xi} = g_1^\xi$, and the term $g^{\sum_{i\in\mathbf{C}} w_i \mathcal{R}_{s_0}(i)}$ is computed as $\prod_{i\in\mathbf{C}} \left(g^{\mathcal{R}_{s_0}(i)}\right)^{w_i}$. Recall that challenge $(\mathbf{C}, \{w_i\}, \xi)$ is chosen by simulator, $(z, \phi_\alpha, \bar{\sigma})$ is (part of) response provided by adversary, and (g, g_1, g_2) are part of public key pk. So the simulator can find the exactly same decision bit $b_0 \in \{\mathtt{1}, \mathtt{0}\}$ as in a real game. Simulator sends (b, b_0) to the adversary as result to this query.
- REVOKEVK-QUERY: The simulator has knowledge of vpk and vsk, and runs algorithm $\mathsf{UpdVK}(vpk, vsk, \{t_i\})$ to obtain new verification key pair (vpk', vsk') and new authentication tags $\{t_i'\}$.

B.2 Proof for Lemma 3

Proof. Our hypothesis is: Adversary $\mathcal{A}$ can output such (T_0, T_1) as stated in Lemma 3. Let bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and tuple $T = (g, g^\varsigma, g^{\varsigma^2}, \ldots, g^{\varsigma^m}) \in \mathbb{G}^m$ be as stated in the m-BSDH Assumption. We will construct a PPT algorithm $\mathcal{B}$, which will simulate **Game 2** to interact with adversary $\mathcal{A}$ as in Appendix B.1, and then compute (c, d) from $\mathcal{A}$'s output (T_0, T_1), such that $d = e(g,g)^{1/(\varsigma+c)}$.

The algorithm $\mathcal{B}$ simulates the **Game 2** to interact with the adversary $\mathcal{A}$, from information T, as in Appendix B.1. By the hypothesis, both T_0 and T_1 are valid, i.e. satisfy Eq (9) and Eq (10):

$$e(\psi_\alpha, g) = e(\phi_\alpha,\ g^\alpha/g^\xi) \cdot e(g,g)^z \tag{59}$$

$$e(\psi_\alpha, g) = e(\phi'_\alpha,\ g^\alpha/g^\xi) \cdot e(g,g)^{z'} \tag{60}$$

Combining Eq (72) and Eq (73), we have

$$e(\phi_\alpha/\phi'_\alpha,\ g^\alpha/g^\xi) = e(g,g)^{z'-z}. \tag{61}$$

Since $z' \neq z \mod p$, the inverse $(z'-z)^{-1} \mod p$ exists. Let $c = -\xi \mod p$ and compute d as

$$d = e(\phi_\alpha/\phi'_\alpha,\ g)^{(z'-z)^{-1} \mod p} \in \mathbb{G}_T. \tag{62}$$

Output (c, d) as the solution to the m-BSDH problem. One can verify that

$$d^{\alpha+c} = \left(e(\phi_\alpha/\phi'_\alpha,\ g)^{(z'-z)^{-1} \mod p}\right)^{\alpha-\xi} = e(\phi_\alpha/\phi'_\alpha,\ g^{\alpha-\xi})^{(z'-z)^{-1}} \tag{63}$$

$$= \left(e(g,g)^{z'-z}\right)^{(z'-z)^{-1} \mod p} = e(g,g). \tag{64}$$

B.3 Proof for Lemma 4

Proof. Our hypothesis is: Adversary $\mathcal{A}$ can output such (T_0, T_1) as stated in Lemma 4. Let bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ and tuple $\mathbf{T} = (g, g^\varsigma, g^{\varsigma^2}, \ldots, g^{\varsigma^m}) \in \mathbb{G}^m$ be as stated in the m-BSDH Assumption. We will construct a PPT algorithm $\mathcal{D}$, which will simulate **Game 2** to interact with adversary $\mathcal{A}$, and then compute (c, d) from $\mathcal{A}$'s output (T_0, T_1), such that $d = e(g,g)^{1/(\varsigma+c)}$.

The algorithm $\mathcal{D}$ simulates the **Game 2** to interact with the adversary $\mathcal{A}$, from information $\mathbf{T}$, as in Appendix B.1. By the hypothesis, both T_0 and T_1 are valid, i.e. satisfy Eq (9) and Eq (10):

$$\left(\frac{e(\psi_\alpha,\ g^\alpha)}{e\left(g,\ g^{\bar{\sigma}}\right)}\right)^\gamma = \frac{e(\psi_\beta, g)}{e\left(g,\ g^{\bar{t}} \cdot g^{-\sum\limits_{i\in\mathbf{C}} w_i \mathcal{R}_{s_1}(i)}\right)} \tag{65}$$

$$\left(\frac{e(\psi'_\alpha,\ g^\alpha)}{e\left(g,\ g^{\bar{\sigma}'}\right)}\right)^\gamma = \frac{e(\psi'_\beta, g)}{e\left(g,\ g^{\bar{t}'} \cdot g^{-\sum\limits_{i\in\mathbf{C}} w_i \mathcal{R}_{s_1}(i)}\right)} \tag{66}$$

Divide Eq (65) with Eq (66), we have

$$\left(\frac{e(\frac{\psi_\alpha}{\psi'_\alpha},\ g^\alpha)}{e(g,g)^{\Delta\sigma}}\right)^\gamma = \left(\frac{e(\frac{\psi_\beta}{\psi'_\beta},\ g)}{e(g,g)^{\Delta t}}\right) \quad \in \mathbb{G}_T \tag{67}$$

where $\Delta\sigma := \bar{\sigma}' - \bar{\sigma}$ and $\Delta t := \bar{t}' - \bar{t}$.

For ease of exposition, let us represent the above Eq (67) as $A^\gamma = B$, where the meaning of variable A and B can be explained straightforwardly by looking at Eq (67). The adversary $\mathcal{A}$ (i.e. the dishonest prover/cloud storage server) has sufficient information to compute values of A and B by itself. So the adversary $\mathcal{A}$ is able to compute values A and B, such that $A^\gamma = B$.

Notice that, in our scheme, among all data that the adversary (dishonest cloud storage server) owns (i.e data blocks, authentication tags $\{\sigma_i, t_i\}$, and public keys (pk, vpk)), the secret value γ only appears in the computation of t_i, where γ is perfectly protected by $\mathcal{R}_{s_1}(i)$ (which is *true* randomness in **Game 2**) from the adversary $\mathcal{A}$ (dishonest prover). Once a verification public/private key pair is revoked, γ will be re-randomized as $\gamma'\gamma$, where γ' is a newly chosen uniform random variable hidden from $\mathcal{A}$. Therefore, γ is semantically secure against $\mathcal{A}$, and $\mathcal{A}$ is unable do brute-force search attack to find values of γ. Therefore, the adversary $\mathcal{A}$ is unable to compute a pair (A, B) such that $A^\gamma = B$ and $A \neq 1$. As a result, it has to be the case that $A = B = \mathbf{1} = e(g,g)^0 \in \mathbb{G}_T$.

We have

$$A = \left(\frac{e(\frac{\psi_\alpha}{\psi'_\alpha},\ g^\alpha)}{e(g,g)^{r_\sigma^{-1} \cdot r \cdot \Delta\sigma}}\right) = \left(\frac{e(\frac{\psi_\beta}{\psi'_\beta},\ g^\beta)}{e(g,g)^{r_t^{-1} \cdot r \cdot \Delta t}}\right) = B = \mathbf{1} \quad \in \mathbb{G}_T \tag{68}$$

Recall that $(\psi_\alpha, \psi_\beta, \bar{\sigma}, \bar{t}) \neq (\psi'_\alpha, \psi'_\beta, \bar{\sigma}', \bar{t}')$ are distinct, by our hypothesis.

Case 1: $\Delta\sigma \neq 0 \mod p$**.** Let $c = 0 \in \mathbb{Z}_p$ and compute $d \in \mathbb{G}_T$ as below

$$d = e(\frac{\psi_\alpha}{\psi'_\alpha},\ g)^{(\Delta\sigma)^{-1} \mod p} \in \mathbb{G}_T. \tag{69}$$

One can verify that (c, d) is a solution to the m-BSDH problem, by computing

$$e(g,\ g) = \left(e(\frac{\psi_\alpha}{\psi'_\alpha}, g^\alpha)\right)^{(\Delta\sigma)^{-1} \mod p} \tag{70}$$

$$e(g,g)^{1/(\alpha+c)} = \left(e(\frac{\psi_\alpha}{\psi'_\alpha}, g^{\alpha/(\alpha+c)})\right)^{(\Delta\sigma)^{-1} \mod p} = d \ \in \mathbb{G}_T. \tag{71}$$

Case 2: $\Delta\sigma = 0 \mod p$**.** Since $\Delta\sigma = 0$, $A = \mathbf{1} \in \mathbb{G}_T$ and $\alpha \neq 0$, we can conclude $\psi_\alpha = \psi'_\alpha$. Both tuples T_0 and T_1 should also satisfy Eq (9), we have

$$e(\psi_\alpha, g) = e(\phi_\alpha,\ g^\alpha / g^\xi) \cdot e(g,g)^z \tag{72}$$

$$e(\psi'_\alpha, g) = e(\phi'_\alpha,\ g^\alpha / g^\xi) \cdot e(g,g)^{z'} \tag{73}$$

Notice $\psi_\alpha = \psi'_\alpha$, and $z \neq z'$. The same as proof of Lemma 3, we let $c = -\xi \mod p$ and compute d as $d = e(\phi_\alpha/\phi'_\alpha,\ g)^{(z'-z)^{-1} \mod p} \in \mathbb{G}_T$, where (c, d) will be a solution to the m-BSD problem.

Therefore, $\mathsf{Pr}\left[(c,d) \text{ solves } m\text{-BSDH problem}\right] = \mathsf{Pr}\left[\mathcal{A} \text{ wins Lem 4 }\right]$. □

B.4 Proof of Lemma 5

B.4.1 Proof of Claim 1

Proof. **Failure1** Loot at the "if" statement at Line 9 of SOLVER Algorithm 2 (on page 15). Since our hypothesis of Claim 1 is $|\mathbf{S}_{\mathcal{W}}| \geq c$, we have $\mathsf{Pr}[\text{SOLVER outputs } \texttt{Failure1}] = 0$.
Failure2 Loot at codes around the "if" statement at Line 15 of SOLVER Algorithm 2. By the definition of $\mathbf{S}_{\mathcal{W}}$, for each $W \in \mathbf{S}_{\mathcal{W}}$, there exists an witness set $\mathbb{W}_{[W]}$ with size at least m. Thus, $\mathsf{Pr}[\text{SOLVER outputs } \texttt{Failure2}] = 0$.
Failure3 Loot at codes around the "if" statement at Line 21 of SOLVER Algorithm 2. By definition of set $\mathbf{S}_{\mathcal{W}}$, since $\xi^*_{i,j} \in \mathbb{W}_{[W^*_i]}$, where $\mathbb{W}_{[W^*_i]}$ is the witness set for $W^*_i \in \mathbf{S}_{\mathcal{W}}$, we have $(W^*_i, \xi^*_{i,j}) \in \mathbf{S}_{\boldsymbol{\xi}}$. By definition of set $\mathbf{S}_{\boldsymbol{\xi}}$, the witness set $\mathbb{W}_{[W^*_i, \xi^*_{i,j}]}$ has size at least 1. Therefore, $\mathsf{Pr}[\text{SOLVER outputs } \texttt{Failure3}] = 0$.
Failure4 Look at codes from Line 27 to Line 29 of SOLVER Algorithm 2. According to Lagrange polynomial interpolation method, the polynomial of degree $(m-1)$ that passes through all of m *distinct* points do exit and is unique. Thus, $\mathsf{Pr}[\text{SOLVER outputs } \texttt{Failure4}] = 0$.
Failure5 Look at codes from Line 32 to Line 34 of SOLVER Algorithm 2. In the linear equation system at Line 32, the coefficient matrix is $({W_1^*}^\top, \ldots, {W_c^*}^\top)$, where row vector $W^*_i = (w_i, w_i^2, \ldots, w_i^c)$ for $i \in [1, c]$ and all w_i's are distinct (See codes around Line 12 of Algorithm 2). It is straightforward to find that this coefficient matrix has non-zero determinant, due to the property of Vandermonde matrix. Therefore, the linear equation system has unique solution, and $\mathsf{Pr}[\text{SOLVER outputs } \texttt{Failure5}] = 0$.

For the above reason, if set $\mathbf{S}_{\mathcal{W}}$ has size at least c, then with overwhelming high probability, the procedure SOLVER will output a solution (i.e. return a value at Line 36), instead of abort with failure. □

B.4.2 Proof of Claim 2

Proof. By definition of sets $\mathbf{S}_{\boldsymbol{\xi}}$ and $\mathbf{S}_{\mathcal{W}}$, these sets only concern tuple $(W, \xi, z, b) \in \mathbf{S}$ with $b = 1$. So at Line 27, the value $z^*_{i,j}$ is genuine for any $i \in [1, c]$, $j \in [1, m]$, by the assumption of Claim 2.

Since polynomial with degree $(m-1)$ and passing m distinct points is unique (Lagrange polynomial), the resulting polynomial $\mathsf{Poly}_{\vec{\boldsymbol{\mathcal{F}}}_i}$ at Line 27 is genuine. Recall Eq (3) at Step P1 of prover algorithm P_1,

$$\vec{\boldsymbol{\mathcal{F}}}_i = \sum_{\iota \in \mathbf{C}^*} w_\iota \vec{\boldsymbol{F}}_\iota \mod p. \tag{74}$$

Therefore, $\{\vec{\boldsymbol{F}}_\iota\}_{\iota \in \mathbf{C}^*}$ is a solution to the linear equation system at Line 32. Since this linear equation system has unique solution A^*_i's, we have $\{\vec{\boldsymbol{F}}_\iota\}_{\iota \in \mathbf{C}^*} = \{A^*_i : i \in [1, c]\}$. More precisely, letting $\mathbf{C}^* = \{\iota_1, \iota_2, \ldots, \iota_c\}$, where $\iota_1 < \iota_2 < \iota_3 < \ldots < \iota_c$, we have $\vec{\boldsymbol{F}}_{\iota_i} = A^*_i$ for each $i \in [1, c]$.

□

B.5 Proof of Lemma 6

Claim 5 *Suppose* $\Pr_{\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}) = 1] \geq \epsilon$. *We have*

1. $\mathsf{Pr}[\mathcal{W}$ *is* $(\epsilon/2)$*-good*$] > \epsilon/2$;
2. *If* W *is* $(\epsilon/2)$*-good, then* $\mathsf{Pr}[\boldsymbol{\xi}$ *is* $(\epsilon/4)$*-good w.r.t* $W] > \epsilon/4$.

Proof (Proof of Claim 5). Treating W as the first parameter of function $f_{\mathbf{C}^*}$, and the 2-tuple $(\xi, \mathfrak{C}_{\mathcal{A}})$ as the second parameter, we apply Lemma 8 and have $\mathsf{Pr}[\mathcal{W}$ is $(\epsilon/2)$-good$] > \epsilon/2$. Define a new function

$$f_{W, \mathbf{C}^*}(\xi, \mathfrak{C}_{\mathcal{A}}) \stackrel{\text{def}}{=} f_{\mathbf{C}^*}(W, \xi, \mathfrak{C}_{\mathcal{A}}).$$

Given that W is $(\epsilon/2)$-good, we apply Lemma 8 again on function $f_{W, \mathbf{C}^*}(\cdot, \cdot)$, where the first parameter of $f_{W, \mathbf{C}^*}$ is ξ, and the second parameter is $\mathfrak{C}_{\mathcal{A}}$. We get $\mathsf{Pr}[\boldsymbol{\xi}$ is $(\epsilon/4)$-good w.r.t. $W] > \epsilon/4$.

□

B.5.1 Proof of Claim 3

Proof. Looking at Line 3 of SIMULATOR Algorithm 1, N_1 instances of value $\mathcal{W}$ will be generated and denoted as $W_1, \ldots, W_{N_1}$. Define $r_i = 1$ if W_i is $(\epsilon/2)$-good, otherwise $r_i = 0$. Then $(r_1, \ldots, r_{N_1})$ can be considered as the result of N_1 independent Bernoulli experiment with probability $\mathsf{Pr}[r_i = 1] = \mathsf{Pr}[\mathcal{W}$ is $\epsilon/2$-good$]$. We have probability $\mathsf{Pr}[\mathcal{W}$ is $\epsilon/2$-good$] > \epsilon/2$, under our hypothesis that $\Pr_{\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(\mathcal{W}, \boldsymbol{\xi}, \mathfrak{C}_{\mathcal{A}}) = 1] \geq \epsilon$, according to Claim 5. Therefore, $\mathsf{Pr}[r_i = 1] > \epsilon/2$ for each i.

Notice that $N_1 = \lceil \lambda(\epsilon/2)^{-1} \rceil$. We apply Lemma 9 on $r_1, \ldots, r_{N_1}$ which are results of N_1 independent Bernoulli experiment with probability $\delta = \mathsf{Pr}[r_i = 1] > \epsilon/2$. With overwhelming high probability in λ, there exist $d := c <$

$\lambda^{0.9}$ distinct indices $i_1, \ldots, i_c \in [1, N_1]$, such that $r_{i_j} = 1$ for each $j \in [1, c]$. As a result, for each $j \in [1, c]$, W_{i_j} is $\epsilon/2$-good.

Since all W_i's are chosen independently and uniformly randomly from a domain of size $(p-1)$, the collision probability is

$$\mathsf{Pr}\left[\exists a \neq b \in \{i_1, \ldots, i_c\}, \quad W_a = W_b\right] \tag{75}$$

$$\leq \binom{c}{2} \mathsf{Pr}\left[W_{i_1} = W_{i_2}\right] < \frac{1}{2}c^2 \cdot \frac{1}{p-1} \tag{76}$$

which is negligible in $\lambda \approx \log p$. Consequently, all $W_{i_1}, \ldots, W_{i_c}$ are distinct with o.h.p. $(1 - \frac{1}{2}c^2 \cdot \frac{1}{p-1})$. Claim 3 is proved.

B.5.2 Proof of Claim 4

Proof (Proof of Claim 4). By applying Claim 5, if W_i is $(\epsilon/2)$-good, then $\mathsf{Pr}_{\boldsymbol{\xi}}[\boldsymbol{\xi}$ is $(\epsilon/4)$-good w.r.t $W_i] > \epsilon/4$.

Let $r_j = 1$ if $\xi_{i,j}$ is $\epsilon/4$-good w.r.t. W_i, and $r_j = 0$ otherwise. Then $(r_1, \ldots, r_{N_2})$ can be considered as the result of N_1 independent Bernoulli experiment with probability $\mathsf{Pr}[r_j = 1] = \mathsf{Pr}[\boldsymbol{\xi}$ is $(\epsilon/4)$-good w.r.t $W_i] > \epsilon/4$.

Notice that $N_2 = \lceil \lambda(\epsilon/4)^{-1} \rceil$. Applying Lemma 9 over r_j's, with overwhelming high probability, there exist $d := m < \lambda^{0.9}$ indices $j_\iota, \iota \in [1, m]$, such that $r_{j_\iota} = 1$ and thus ξ_{i,j_ι} is $\epsilon/4$-good w.r.t. W_i.

For each $\iota \in [1, m]$, we can show that $(W_i, \xi_{i,j_\iota}) \in \mathbf{S}_{\boldsymbol{\xi}}$ as below: Let $r_k = 1$ if $b_{i,j_\iota,k} = 1$, otherwise $r_k = 0$. $\mathsf{Pr}[r_k = 1] = \mathsf{Pr}[b_{i,j_\iota,k} = 1] = \Pr_{\mathfrak{C}_{\mathcal{A}}}[f_{\mathbf{C}^*}(W_i, \xi_{i,j_\iota}, \ \mathfrak{C}_{\mathcal{A}}) = 1] \geq \epsilon/4$. Notice that $N_3 = \lceil \lambda(\epsilon/4)^{-1} \rceil$. Applying Lemma 9 over r_k's, with overwhelming high probability, there exists at least one index $k^* \in [1, N_3]$, such that $r_{k^*} = 1$, which implies that $b_{i,j_\iota,k^*} = 1$. Therefore, $(W_i, \xi_{i,j_\iota}, z_{i,j_\iota,k^*}, b_{i,j_\iota,k^*} = 1) \in \mathbf{S}$. By definition of $\mathbf{S}_{\boldsymbol{\xi}}$, $(W_i, \xi_{i,j_\iota}) \in \mathbf{S}_{\boldsymbol{\xi}}$ with $\{z_{i,j_\iota,k^*}\}$ as witness set.

Consequently, $W_i \in \mathbf{S}_{\mathcal{W}}$ with $\{\xi_{i,j_\iota}\}_{\iota \in [1,m]}$ as witness set. Claim 4 is proved.

C Proof of Soundness w.r.t Owner

C.1 Proof of Lemma 7

Proof. Simulate **Game 2'** using the input of m-BSDH problem, as in Appendix B.1, with the difference that vsk is given to the adversary in setup phase. Since both $T_0 = (z, \phi_\alpha, \bar{\sigma})$ and $T_1 = (z', \phi'_\alpha, \bar{\sigma}')$ are valid, i.e. satisfying Eq (11) in algorithm OwnerVerify, we have

$$\left(e(\phi_\alpha, g^\alpha / g^\xi) e(g,g)^z\right)^\alpha \ \stackrel{?}{=} \ e(g, \ g^{\bar{\sigma}}) \cdot e(g,g)^{\left(-\sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)\right)} \tag{77}$$

$$\left(e(\phi'_\alpha, g^\alpha / g^\xi) e(g,g)^{z'}\right)^\alpha \ \stackrel{?}{=} \ e(g, \ g^{\bar{\sigma}'}) \cdot e(g,g)^{\left(-\sum_{i \in \mathbf{C}} w_i \mathcal{R}_{s_0}(i)\right)} \tag{78}$$

Combining Eq (77) and Eq (78), we have

$$\left(e(\phi'_\alpha \phi_\alpha^{-1}, \ g^\alpha / g^\xi) e(g,g)^{z'-z}\right)^\alpha \ \stackrel{?}{=} \ e(g, \ g^{\bar{\sigma}'-\bar{\sigma}}) \tag{79}$$

Case 1: $\bar{\sigma}' = \bar{\sigma}$. If $z' = z$, then we have $e(\phi'_\alpha \phi_\alpha^{-1}, \ g^\alpha / g^\xi) = e(g,g)^0$, which implies that $\xi = \alpha$ or $\phi'_\alpha = \phi_\alpha$.

Otherwise, $z' \neq z$. Thus the inverse $(z'-z)^{-1} \mod p$ exits. Let $c = 0$ and compute d as below

$$d = e(\phi'_\alpha \phi_\alpha^{-1}, \ g^\alpha / g^\xi)^{(z'-z)^{-1}} \in \mathbb{G}_T \tag{80}$$

It is easy to verify that $d^{\alpha+c} = d^\alpha = e(g,g)$.

Case 2: $\bar{\sigma}' \neq \bar{\sigma}$. Thus the inverse $(\sigma' - \sigma)^{-1} \mod p$ exits. Let $c = 0$ and compute d as

$$d = \left(e(\phi'_\alpha \phi_\alpha^{-1}, \ g^\alpha / g^\xi) e(g,g)^{z'-z}\right)^{(\bar{\sigma}'-\bar{\sigma})^{-1} \mod p} \in \mathbb{G}_T \tag{81}$$

It is easy to verify that $d^{\alpha+c} = d^\alpha = e(g,g)$. □

D Two Lemmas on Random Sampling

In this section, we provide two lemmas on random sampling which will simplify our proof of main theorem. These lemmas may have appeared (implicitly) in the cryptography literature. Here we do not declare any contribution on them.

Lemma 8 *Let $\mathbb{X}$ and $\mathbb{Y}$ be two finite sets. Let $\mathcal{X}$ denote a random variable over the domain $\mathbb{X}$ and $\mathcal{Y}$ denote a (dependent or independent) random variable over the domain $\mathbb{Y}$. Consider any function $f : \mathbb{X} \times \mathbb{Y} \to \{0, 1\}$, such that $\mathsf{Pr}[f(\mathcal{X}, \mathcal{Y}) = 1] \geq \epsilon > 0$ for some real value ϵ. For any constant $a \in (0, \frac{1}{2})$, define a set $\mathbf{S}_a = \{x \in \mathbb{X} : \mathsf{Pr}[f(x, \mathcal{Y}) = 1] \geq a\epsilon\}$. We have*

$$\mathsf{Pr}[\mathcal{X} \in \mathbf{S}_a] \geq \left(\frac{1-a}{\frac{1}{\epsilon} - a}\right) > \frac{1}{2}\epsilon = \mathcal{O}(\epsilon).$$

Proof (Proof of Lemma 8). We have

$$\begin{aligned}
& \epsilon && (82)\\
& \leq \mathsf{Pr}[f(\mathcal{X}, \mathcal{Y}) = 1] && (83)\\
& = \sum_{x \in \mathbf{S}_a} \mathsf{Pr}[f(x, \mathcal{Y}) = 1]\mathsf{Pr}[\mathcal{X} = x] \ + \\
& \quad \sum_{x \in \mathbb{X} \setminus \mathbf{S}_a} \mathsf{Pr}[f(x, \mathcal{Y}) = 1]\mathsf{Pr}[\mathcal{X} = x] && (84)\\
& \leq \sum_{x \in \mathbf{S}_a} 1 \cdot \mathsf{Pr}[\mathcal{X} = x] \ + \ \sum_{x \in \mathbb{X} \setminus \mathbf{S}_a} a\epsilon \cdot \mathsf{Pr}[\mathcal{X} = x] && (85)\\
& = a\epsilon + (1 - a\epsilon) \sum_{x \in \mathbf{S}_a} \mathsf{Pr}[\mathcal{X} = x] && (86)\\
\Rightarrow \quad & \mathsf{Pr}[\mathcal{X} \in \mathbf{S}_a] = \sum_{x \in \mathbf{S}_a} \mathsf{Pr}[\mathcal{X} = x] \geq \left(\frac{1-a}{\frac{1}{\epsilon} - a}\right) \in (\frac{1}{2}\epsilon, \ \epsilon] = \mathcal{O}(\epsilon).
\end{aligned}$$

□

Lemma 9 *Let κ be an integer. Let $\delta, \epsilon \in (0, 1]$ be two real values and $\delta \geq \epsilon$. Let $t = \lceil \kappa\epsilon^{-1} \rceil$. Independently sample t number of values $r_1, \ldots, r_t$ from $\{0, 1\}$ under the Bernoulli distribution with probability δ. Let d be a positive integer and $d \leq \kappa^c$ for some real valued constant $c \in (0, 1)$. Then with overwhelming high probability (w.r.t. κ), there exist d distinct indices $i_1, i_2, \ldots, i_d \in [1, t]$ such that $\forall j \in [1, d], r_{i_j} = 1$.*

We remark that the original form of Chernoff bound [15, 23] does not serve our purpose.

Proof (Proof of Lemma 9). Let us consider the set $\mathbf{S} = \{i \in [1, t] : r_i = 1\}$. We will show that the size of set $\mathbf{S}$ is at least d with overwhelming high probability.

For each index $i \in [1, t]$, we have $\mathsf{Pr}[r_i = 1] = \delta \geq \epsilon$.

$$\begin{aligned}
\mathsf{Pr}[|\mathbf{S}| < d] = \sum_{j=0}^{d-1} \mathsf{Pr}[|\mathbf{S}| = j] &= \sum_{j=0}^{d-1} \binom{t}{j} \delta^j (1-\delta)^{t-j} && (87)\\
&\leq \sum_{j=0}^{d-1} \binom{t}{j} \epsilon^j (1-\epsilon)^{t-j} && (88)\\
&\leq \sum_{j=0}^{d-1} \binom{t}{d-1} \epsilon^{d-1} (1-\epsilon)^{t-d+1} && (89)\\
&\leq d(\kappa+1)^{d-1} e^{-\kappa + \mathcal{O}(\kappa^c)} && (90)\\
&\leq \frac{\kappa^c}{e^{\mathcal{O}(\kappa)}} && (91)
\end{aligned}$$

where e is the base of natural logarithm.

Now we explain the above inference. Equation (88) can be derived from Equation (87), since for each $j \in \{0, 1, 2, \ldots, d-1\}$, function $f(\delta) = \delta^j (1-\delta)^{t-j}, \delta \in (0, 1)$, has a negative first order derivative since $\kappa > \kappa^c \geq d$, and is thus a monotone decreasing function of $\delta \in (0, 1)$.

Equation (89) can be derived from Equation (88), since the probability mass function of a binomial distribution is a bell shape (like normal distribution) and reach its maximum at $j = t \cdot \epsilon \geq \kappa > d$. That is, for $0 \leq j < d < t \cdot \epsilon$, the function $g(j) = \binom{t}{j} (\epsilon)^j (1-\epsilon)^{t-j}$ is a monotone increasing function of j.

Equation (90) can be derived from Equation (89), due to the following two Equations (92) and (93).

$$\begin{aligned}(1-\epsilon)^{t-d+1} &= (1-\epsilon)^{\frac{1}{\epsilon} \times (\epsilon t + \epsilon(-d+1))} \leq (1-\epsilon)^{\frac{1}{\epsilon} \times (\kappa - \epsilon\kappa^c)} \\ &< \left(\frac{1}{e}\right)^{\kappa - \mathcal{O}(\kappa^c)} = e^{-\kappa + \mathcal{O}(\kappa^c)}. \end{aligned} \tag{92}$$

$$\binom{t}{d-1} \epsilon^{d-1} \leq t^{d-1} \epsilon^{d-1} = (t\epsilon)^{d-1} \leq (\kappa+1)^{d-1} \tag{93}$$

Equation (91) can be derived from Equation (90), since

$$\frac{(\kappa+1)^{d-1}}{e^{\kappa - \mathcal{O}(\kappa^c)}} \leq \frac{(\kappa+1)^{(\kappa^c - 1)}}{e^{\kappa - \mathcal{O}(\kappa^c)}} = e^{(\kappa^c - 1)\ln(\kappa+1) - \kappa + \mathcal{O}(\kappa^c)} = e^{-\mathcal{O}(\kappa)}. \tag{94}$$

So $de^{-\mathcal{O}(\kappa)} \leq \kappa^c \cdot e^{-\mathcal{O}(\kappa)}$ is negligible in κ. Recall that $d < \kappa^c$ for constant $c \in (0,1)$.

Therefore, $\mathsf{Pr}[|\mathbf{S}| < d]$ is negligible in κ and the set $\mathbf{S}$ has size at least d with overwhelming high probability $(1 - \mathsf{Pr}[|\mathbf{S}| < d])$ in κ.