# Multi-file proofs of retrievability for cloud storage auditing

Bin Wang[a*] and Xiaojing Hong[b]

[a]No.196 West HuaYang Road, Information Engineering College of Yangzhou University, Yangzhou City, Jiangsu Province, 225127 P.R.China

[b]No.5 South YangZiJiang Road, Yangzhou City, Jiangsu Province, 225101 P.R. China

**E-mail:** jxbin76@yeah.net [a*]

**Abstract:** Cloud storage allows clients to store a large amount of data with the help of storage service providers (SSPs). Proof-of-retrievability(POR) protocols allow one server to prove to a verifier the availability of data stored by some client. Shacham et al. presented POR protocols based on homomorphic authenticators and proved security of their schemes under a stronger security model, which requires the existence of an extractor to retrieve the original file by receiving the program of a successful prover. When using their POR protocol with public verifiability to verify the availability of multiple files separately, the number of pairing operations computed by a verifier is linear with the number of files. To improve the heavy burden on the verifier, we introduce a notion called multi-proof-of-retrievability(MPOR), allowing one verifier to verify the availability of multiple files stored by a server in one pass. We also design a MPOR protocol with public verifiability by extending the work of Shacham et al. The advantage of our MPOR scheme is that computational overhead of a verifier in our scheme is constant, independent of the number of files. Nevertheless, the soundness of our MPOR protocol is proved under a relatively weak security notion. In particular, analysis of our MPOR protocol shows that each file can be extracted in expected polynomial time under certain restriction on the size of processed files.

**Keywords:** Cloud storage; Storage service provider; Proof-of-retrievability; Homomorphic authenticator; Public verifiability

## 1. Introduction

Cloud computing [13] has a deep impact on IT industry by allowing clients to access high-quality cloud based service in pay-as-you-go manner. Cloud storage systems which involve delivery of data storage as a service over Internet are becoming more attractive. Amazon S3 [1], a well-known storage service, allows clients to store large amounts of data and access their stored data from any place at low costs. One of the major challenges on cloud storage systems is how to efficiently verify whether some remote storage service provider(SSP) is faithfully storing clients' data since the SSP may not be trusted. Clients' data may be at the danger of loss either by data loss incidents or by malicious deletion from a SSP who wants to save storage by deleting rarely accessed portion of the stored data.

There has been a lot of research focused on proof-of-storage mechanisms without the need to download the whole data for verification. This kind of solutions is also called storage auditing [9,15,17], allowing clients to verify through cryptographic means that their data on an untrusted SSP are kept intact and ready for retrieval if needed. Some crucial system criteria for designing proof-of-storage systems can be summarized as follows [16]:

(1) *Computation and communication overhead* of proof-of-storage protocols and *storage overhead* on a server should be as efficient as possible.

(2) Verifiers should be *stateless* without the need to maintain states for storage auditing.

(3) The most important security criterion for proof-of-storage systems requires that a verifier should be convinced that the file is actually stored by a server when the server can pass the check of storage auditing. Early research works on proof-of-storage systems lacks provable security guarantee without providing formal security models and proofs.

To formalize the security criterion for proof-of-storage systems, formal models and constructions were first studied by Naor et al. [14] and Juels et al. [12] by using homomorphic authenticators based on message authentication code. A file is first encoded redundantly and divided into message blocks. Each block is authenticated by a MAC tag computed by the client. The client then erases the file and sends the encoded file and the MAC tags to the server. The idea behind their protocols is that a verifier only need check a random subset of the message blocks stored by the server to check whether the original file is correctly stored by the server in order to guarantee an efficient storage audit protocol. The length of the server's response is improved with the help of homomorphic authenticators by aggregating the authentication tags of different message blocks.

Security for proof-of-storage systems is captured by requiring the existence of an extractor to retrieve the specified file by interacting with a server that can pass the verification of storage auditing. This security notion is also called "proof-of-retrievability" (POR). This concept is similar to zero knowledge proofs of knowledge [5]. A weaker notion called "Proof-of-data-possession"(PDP) [3] can only guarantee that a certain percentage (e.g., 99%) of message blocks can be recovered with high probability (e.g., 0.99). There are also some papers [4,7,18] considering proof-of-storage systems supporting dynamic operations on the stored data. In addition, POR protocols that support *public verifiability* means everyone can check whether a client's data are correctly stored by a server. Digital signatures can be used to replace Mac tags to achieve public verifiability.

The NR model in [14] restricts a checker to access specific memory locations from the prover. Shacham et al. [16] strengthened the NR model by allowing an extractor black-box access to a prover program. Another distinct feature formalized by Shacham et al. in their security model [16] is that the specified file can be extracted so long as the prover program can correctly answer any small (but non-negligible) fraction of verification queries. In addition, Shacham et al. [16] designed a homomorphic authenticator based on BLS short signature [6] to present a POR protocol with public verifiability secure in the random oracle model. Homomorphic authenticators can be used to improve the response length from the server by compressing authentication values $\{\sigma_i\}$ of blocks $\{m_i\}$ into one authenticator $\sigma$ for their linear combination $\{\sum v_i m_i\}$.

We notice that the POR protocol with public verifiability [16] requires the verifier to compute two pairing operations to verify the response from a server during one instance of their POR protocol. This means if one wants to verify the availability of $n$ files stored by the server separately, $2n$ pairing operations should be computed by the verifier, which is a heavy overhead when a client has a large number of files stored by that server.

Motivated by the above discussion, we aim to design a multi-proof-of-retrievability (MPOR) protocol with public verifiability. In other words, our MPOR protocol allows one verifier to verify the availability of $n$ files stored by the server in one pass, while the computational overhead is independent of the parameter $n$. The idea behind our construction is to further compress the homomorphic authenticators from different files to reduce the computation and communication overhead.

The key point of our work is to choose a proper security model for MPOR protocol and prove the security of our construction under it. Soundness of MPOR protocols is defined in a relatively weak manner compared with that of POR protocols. In the multi-file setting, one invocation of the extractor algorithm can only guarantee that at least one file among those $n$ files can be recovered. The reason is that the extracted knowledge is now distributed over these $n$ files randomly. Hence we cannot guarantee that all the files can be extracted by the extractor all at once in the multi-file setting. On the other hand, our analysis shows that each file can be extracted in expected polynomial time under the assumption that each file's size is of the same magnitude (e.g., digital images, office documents). So we define soundness for MPOR protocols by requiring the existence of an extractor to retrieve each file in expected polynomial time.

The rest of this paper is organized as follows. At first, we describe the notations used in this paper and bilinear mappings in section 2. Syntax of MPOR schemes is introduced in section 3. We also define correctness, soundness for MPOR protocols. The soundness for MPOR protocols is relatively weak in the sense that it only requires each file to be extracted in expected polynomial time under the assumption that each file's size is of the same magnitude. In the following, we design a MPOR protocol with public verifiability by extending the work in [16] and prove our MPOR scheme meet the requirement of soundness defined in this paper. Finally, we evaluate performance of our MPOR protocol to show that the cost of a verifier is greatly reduced compared with that of the original POR protocol [16] when verifying multiple files stored by a server.

## 2. Preliminaries

### 2. 1 Notation

We use the notation $x \leftarrow_R S$ to mean "the element $x$ is chosen with uniform probability from the set $S$". If $A$ is a algorithm, then $y \leftarrow A^{O(\cdot)}(x_1, \cdots)$ means that $A$ has input $x_1, \cdots$, access to a oracle $O$, and the output of $A$ is assigned to $y$. Let $[1..n] = \{1, \cdots, n\}$.

### 2.2 Bilinear pairing

Given a security parameter $k$, an efficient algorithm $PG(1^k)$ outputs $(e, G, G_T, g, p)$, where $G$ is a cyclic group of a prime order $p$ generated by

$g, 2^{k-1} < p < 2^k$. $G_T$ is a cyclic group of the same order, and let $e: G \times G \rightarrow G_T$ be an efficiently computable bilinear function with the following properties:

1. Bilinear: $e(g^a, g^b) = e(g,g)^{ab}$, for all $a, b \in Z_p$.

2. Non-degenerate: $e(g,g) \neq 1_{G_T}$

**3. Definitions**

**3.1 Syntax of a multi-proof-of-retrievability scheme**

A multi-proof-of-retrievability(MPOR) scheme consists of the following algorithms:

$Kg(1^k)$: Given a security parameter $k$, this randomized key-generation algorithm outputs a secret/public key pair $(sk, pk)$.

$St(sk, M)$: Given a secret key $sk$ and a file $M$ chosen by a client, this file-storing algorithm first encodes $M$ by applying a rate-$\rho$ erasure code [2] to obtain $\widehat{M}$ such that any $\rho$ fraction of the encoded file $\widehat{M}$ is sufficient to recover the original file $M$. Finally it outputs a authentication tag $\tau$ and some auxiliary information $aug$, $(\widehat{M}, (\tau, aug)) \leftarrow St(M, sk)$, where $sk$ is the secret key of the client. $\widehat{M}$ and $aug$ will be stored on the server side. $\tau$ and $aug$ will be used in the following MPOR protocol.

The rest part of our MPOR scheme consist of the algorithms $P$ (prover) and $MV$ (verifier). The following notation is used to denote one interactive execution of our MPOR protocol between the algorithms $P$ and $MV$:

$$\langle MV(pk, \{\tau_i\}_{i=1}^n) \rightleftarrows P(pk, \{\widehat{M_1}, \cdots \widehat{M_n}\}, \{aug_i\}_{i=1}^n) \rangle \rightarrow (b, \perp)$$

where $pk$ is the public key of the client who stores the encoded files $\{\widehat{M_1}, \cdots \widehat{M_n}\}$ on the server side. Parameter $n$ determines the number of files that can be verified simultaneously. $\{(\tau_i, aug_i)\}_{i=1}^n$ are generated by the file-storing algorithm $St(\cdot)$ for $\{\widehat{M_1}, \cdots \widehat{M_n}\}$ respectively. The output of the verification algorithm $MV$ is bit $b$. $b = 1$ denotes that the verifier is convinced that the original files $\{M_i\}_{i=1}^n$ are stored correctly by the prover.

Correctness of our MPOR scheme requires that an honest server can always convince a verifier of validity of the stored files. That is:

for any $(sk, pk) \leftarrow Kg(1^k)$, $\{(\widehat{M_i}, (\tau_i, aug_i)) \leftarrow St(M_i, sk)\}_{i=1}^n$, we have

$$\langle MV(pk, \{\tau_i\}_{i=1}^n) \rightleftarrows P(pk, \{\widehat{M_1}, \cdots \widehat{M_n}\}, \{aug_i\}_{i=1}^n)\rangle \to (b = 1, \perp)$$

**3.2 Soundness of multi-proof-of-retrievability protocols**

Soundness of MPOR protocols informally requires that if a prover can convince a verifier by passing the verification, then the original files $\{M_i\}_{i=1}^n$ can ready for retrieval if needed. To formalize this point, an extraction algorithm is required to recover the original files $\{M_i\}_{i=1}^n$ by interacting with the successful prover via the MPOR protocol. The following game $Exp_{MPOR}^{sound,n}(A, 1^k)$ will give the formal definition of soundness of MPOR protocols.

$Exp_{MPOR}^{sound,n}(A, 1^k)$

Phase 1: The challenger $C$ generates a keypair $(pk, sk) \leftarrow Kg(1^k)$ and provides $pk$ to $A$.

Phase 2: The adversary $A$ interacts with the challenger by making queries to a store oracle. The store query is handled by the oracle as follows.

Store$(M_i)$ // $M_i$ is a file chosen by $A$.

Compute $(\widehat{M_i}, (\tau_i, aug_i)) \leftarrow St(sk, M_i)$;

Return $(\widehat{M_i}, (\tau_i, aug_i))$;

For some files $M_1, \cdots M_n$ queried to the store oracle who responds $\{(\tau_i, aug_i)\}_{i=1}^n$ respectively, $A$ makes a query to a MPOR oracle, which is handled as follows.

MPOR$(\tau_1, \cdots \tau_n)$

The challenger $C$ runs an instance of the MPOR protocol with $A$ as follows:

$$< C(pk, \{\tau_1, \cdots \tau_n\}) \rightleftarrows A > \to (b, \perp)$$

The challenger plays the role of a verifier during the above execution and outputs a bit $b$ to denote whether it is convinced that the original files are being stored by the adversary correctly.

Return $b$;

At the end of phase 2, $A$ outputs the challenge set $T=\{\tau_1^*,\cdots,\tau_n^*\}$ consisting of the tags returned by the store oracle for some queries $M_1^*,\cdots,M_n^*$ respectively. Description of a prover program $P^*$ is also provided by $A$.

The prover program $P^*$ is $\varepsilon$-admissible if it can convincingly answer $\varepsilon$ fraction of MPOR queries, i.e., $\Pr[<MV((pk,\{\tau_1,\cdots\tau_n\})\rightleftarrows P^*>\rightarrow(b=1,\perp)]\geq\varepsilon$. Here the probability is over the coins of the verifier and the prover.

Phase 3: We say a MPOR scheme is $\varepsilon$-sound if there exists an efficient extraction algorithm $Extr$ such that for every adversary $A$ who plays the above game and outputs the challenge tags $\{\tau_1^*,\cdots,\tau_n^*\}$ returned by the store oracle for the queries $M_1^*,\cdots,M_n^*$ respectively and a $\varepsilon$-admissible prover program $P^*$ at the end of phase 2, then $\forall i,1\leq i\leq n$, there exists an extraction algorithm $Extr$ who recovers the file $M_i^*$ in expected polynomial time by interacting with $P^*$ via the MPOR protocol. That is:

$\forall i,1\leq i\leq n, Extr^{P^*(\cdot)}(pk,\{\tau_1^*,\cdots,\tau_n^*\})\rightarrow M_i^*$ occurs in expected polynomial time except with negligible probability.

**4. Our multi-proof-of-retrievability scheme**

Let $\Sigma=(\text{SKg, SSig, SVer})$ be a digital signature scheme.

$Kg(1^k)$: Generate a signing keypair $(ssk,spk)\leftarrow\text{SKg}(1^k)$ by the key generation algorithm of $\Sigma$. Choose $\alpha\leftarrow_R Z_p$ and compute $v=g^\alpha$. The secret key is $sk=(\alpha,ssk)$. The public key is $pk=(v,spk)$.

$St(sk,M_l)$: Given a file $M_l$ chosen by a client, this file-storing algorithm first chooses a random file-name $fn_l$ from a large domain. The collision probability over file names is negligible when the domain is large enough. In the following, apply a erasure code to $M_l$ to obtain $\widehat{M_l}$ and split $\widehat{M_l}$ into $n_l$ message blocks $\{m_{li}\}_{1\leq i\leq n_l}$. Parse the secret key of the client as $sk\leftarrow(\alpha,ssk)$ and pick a random $u_l\leftarrow Z_p$. Let $\tau_{0l}=fn_l\,||\,n_l\,||\,u_l$ and compute a signature $\sigma_{0l} = \text{SSig}(ssk, \tau_{0l})$ by the signature-producing algorithm of $\Sigma$ and an authentication tag $\tau_l = \tau_{0l} \| \sigma_{0l}$. For $1 \le i \le n_l$, it computes $aug_{li} = (H(fn_l \| i) \cdot u_l^{m_{li}})^{\alpha}$. The encoded file $\widehat{M_l}$ and $\{aug_{li}\}_{i=1}^{n_l}$ will be stored on the server side.

One interactive execution of our MPOR protocol between the algorithms $P$ (prover) and $MV$ (verifier) can be described as follows:

(1) First, the verifier $MV$ makes a MPOR-query $Q$ to the prover $P$: $Q = pk \cup \{P_1 \cup Q_1, \cdots, P_n \cup Q_n\}$, $P_l = (\lambda_l \leftarrow_R Z_p, fn_l)$, $Q_l = \{(j, v_j)\}$ with distinct $j \in [1..n_l]$ and each coefficient $v_j \leftarrow_R Z_p^*$, where $pk$ is the public key of the client, $fn_l$ is the file-name for a file $M_l$ and $n_l$ is the number of message blocks of the file $M_l$. The size of each set $Q_l$ is a fixed system parameter $s$.

(2) Having received the query $Q$, the prover algorithm $P(Q, \{\widetilde{M_l}\}_{l=1}^{n}, \{aug_{li}\}_{1 \le l \le n, 1 \le i \le n_l})$ responds as follows:

First it parses $\{\widetilde{M_l}\}_{l=1}^{n}$ as blocks $\{m_{li}\}_{1 \le l \le n, 1 \le i \le n_l}$ respectively. In the following, compute:

$$\mu_l = \sum_{(j, v_j) \subset Q_l} v_j m_{lj}, \quad \sigma_l = \prod_{(j, v_j) \subset Q_l} aug_{lj}^{v_j}, \quad 1 \le l \le n$$

$$\vec{\mu} = (\mu_1, \cdots, \mu_l, \cdots, \mu_n), \quad \sigma = \prod_{l=1}^{n} \sigma_l^{\lambda_l}$$

The prover $P$ sends the response $(\vec{\mu}, \sigma)$ to the verifier.

(3) Having received the response $(\vec{\mu}, \sigma)$, the verification algorithm $MV(pk, \{\tau_1, \cdots \tau_n\}, (\vec{\mu}, \sigma))$ proceeds as follows:

First it parses $pk = (v, spk)$ and the authentication tags $\tau_l = \tau_{0l} \| \sigma_{0l}$, $1 \le l \le n$.

If $\exists l : 1 \le l \le n$, the signature verification algorithm $\text{SVer}(spk, \tau_{0l}, \sigma_{0l}) = 0$, then it outputs $b = 0$. Otherwise, parse $\tau_{0l}$ as $fn_l \| n_l \| u_l$, $1 \le l \le n$ and the vector $\vec{\mu}$ as $(\mu_1, \cdots, \mu_l, \cdots, \mu_n)$. If the following equation holds, output $b = 1$. Otherwise output $b = 0$.

$$e(\sigma, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l}), v)$$

It is easy to verify the correctness of our MPOR scheme as follows:

$$\sigma = \prod_{l=1}^{n} \sigma_l^{\lambda_l} = \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} aug_{lj}^{\ v_j})^{\lambda_l} = \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l}(H(fn_l \| j)\cdot u_l^{m_{lj}})^{v_j})^{\lambda_l \cdot \alpha}$$

$$= \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l \cdot \alpha} \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} u_l^{v_j \cdot m_{lj}})^{\lambda_l \cdot \alpha}$$

$$= \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l \cdot \alpha} \prod_{l=1}^{n}(u_l^{\sum_{(j,v_j)\subset Q_l} v_j \cdot m_{lj}})^{\lambda_l \cdot \alpha}$$

$$= \prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l \cdot \alpha} \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l})^{\alpha}$$

The above result means

$$e(\sigma, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l \cdot \alpha} \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l})^{\alpha}, g)$$

$$= e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l}), v)$$

**Remark:** The verifier chooses a MPOR query as:

$Q = pk \cup \{P_1 \cup Q_1, \cdots, P_n \cup Q_n\}$ , $P_l = (\lambda_l \leftarrow_R Z_p, fn_l)$ , $Q_l = \{(j, v_j)\}$ with distinct ***indices*** $j \in [1..n_l]$ and each ***coefficient*** $v_j \leftarrow_R Z_p^*$. Let $N = \sum_{1\le i\le n} n_i$ . The size of each set $Q_l$ is a fixed system parameter $s$ such that $ns < N$ .

A vector notation for the $s$-element set $Q_l = \{(j, v_j)\}$ over indices $I_l = \{j\} \subseteq [1..n_l]$ is represented by a $N$-element vector $\overrightarrow{q_l} \in (Z_p)^N$, such that $q_{l,j} = v_j$ if $j \in I_l + N_{l-1}$ and $q_{l,j} = 0$ for all $j \notin I_l + N_{l-1}, 1 \le j \le N$ , where $N_{l-1} = \sum_{1\le i\le l-1} n_i$ , $I_l + N_{l-1} = \{j + N_{l-1} : j \in I_l\}$ .

Given a $ns$-element set $I = \bigcup_{1\le l\le n}(I_l + N_{l-1}) \subseteq [1..N]$, the query $Q$ can be regarded as chosen over the indices $I \subseteq [1..N]$. The joint coefficient vector notation for the query $Q$ is a $N$- element vector $\vec{q} = \sum_{1 \le l \le n} \vec{q_l}$ . Let $\vec{c_1}, \cdots, \vec{c_N}$ be that canonical basis for $(Z_p)^N$ , $\vec{q} = \sum_{\substack{(j,v_j) \in Q_l \\ 1 \le l \le n}} v_j \cdot \overrightarrow{c_{j+N_{l-1}}}$ .

Recall that the honest response $(\vec{\mu}, \sigma)$ for the query $Q$ satisfies:

$$\mu_l = \sum_{(j,v_j) \subset Q_l} v_j m_{lj} \quad \sigma_l = \prod_{(j,v_j) \subset Q_l} aug_{lj}^{v_j} \ , \ 1 \le l \le n$$

$$\vec{\mu} = (\mu_1, \cdots, \mu_l, \cdots, \mu_n), \ \sigma = \prod_{l=1}^{n} \sigma_l^{\lambda_l}$$

We can view the message blocks of these $n$ files as a $N \times n$ matrix $H$ :

$H = [\vec{h_1}, \cdots, \vec{h_n}]$, $\vec{h_l} = (0^{\overrightarrow{\sum_{1 \le i \le l-1} n_i}}, (m_{l1}, \cdots, m_{ln_l}), 0^{\overrightarrow{\sum_{l+1 \le i \le n} n_i}})^T$ , where each column vector $\vec{h_l}$ corresponds to the message blocks of $\widehat{M_l}$ respectively. We can also equivalently denote $\vec{\mu}$ in the response by a vector $\vec{q} \cdot H$ , where $\vec{q}$ is the joint coefficient vector notation for the query $Q$.

## 5. Security proofs

We prove the security of our MPOR scheme by a series of games. Game 0 is exactly the same as $Exp_{MPOR}^{sound,n}(A, 1^k)$ with the following modification.

The challenger initially sets a flag $d = 0$ and keeps a table of the store queries made by the adversary and its responses for these queries. Based on the table and one MPOR query $Q$, the challenger is able to determine the deterministic verification response $(\vec{\mu}, \sigma)$ returned by the honest prover algorithm.

Having received the response $(\hat{\vec{\mu}}, \hat{\sigma})$ returned by the adversary behaving as a prover in one execution of our MPOR protocol $< C(pk, \{\tau_1, \cdots \tau_n\}) \rightleftarrows A > \rightarrow (b, \perp)$, the challenger sets $d = 1$ if for all executions, the verification algorithm $MV(pk, \{\tau_1, \cdots \tau_n\}, \hat{\vec{\mu}}, \hat{\sigma})$ outputs $b = 1$. Otherwise the challenger sets $d = 0$. Let $\varepsilon_i$ denote $\Pr[d = 1]$ in Game $i, i \ge 0$.

The above modification in Game 0 will not change the view of the adversary.

Game 1 is almost the same as Game 0. The challenger keeps a list of authentication tags generated by itself when handling queries to the store oracle. If any tag submitted by the adversary can be verified as valid but is not on the list of tags generated by the challenger, the challenger aborts. The major modifications are represented by the following boxed statements.

Phase 1: The challenger $C$ picks an empty list sList and generates a keypair $(pk,sk) \leftarrow Kg(1^k)$, where $sk=(\alpha,ssk)$, $pk=(v,spk)$. $C$ provides $pk$ to $A$.

The adversary $A$ interacts with the challenger by making queries. The store and MPOR queries are handled as follows.

Store$(M_l)$ // $M_l$ is a file chosen by $A$.

Compute $(\widehat{M_l},\tau_l,\{aug_{li}\}_{i=1}^{n_l}) \leftarrow St(sk,M_l)$;

Parse $\tau_l$ as $\tau_{0l} \| \sigma_{0l}$ and $\boxed{\text{sList} \leftarrow \text{sList} \cup \{\tau_{0l}\}}$

Return $(\widehat{M_l},\tau_l,\{aug_{li}\}_{i=1}^{n_l})$;

MPOR$(\tau_1,\cdots\tau_n)$

If $\boxed{\exists \tau_l \in \{\tau_1,\cdots\tau_n\}}$ such that $\tau_l=\tau_{0l} \| \sigma_{0l}$ can be verified as a valid signature and $\tau_{0l} \notin \text{sList}$, the challenger aborts. Otherwise $C$ runs an instance of MPOR protocol with $A$: $< C(pk,\{\tau_1,\cdots\tau_n\}) \rightleftarrows A > \rightarrow (b,\perp)$.

Return $b$;

Finally, $A$ outputs the challenge set $T=\{\tau_1^*,\cdots,\tau_n^*\}$ consisting of tags returned by the store oracle for some queries $M_1^*,\cdots,M_n^*$ respectively. The description of a prover program $P^*$ is also provided by $A$.

If $\boxed{\exists \tau_l^* \in \{\tau_1^*,\cdots,\tau_n^*\}}$ such that $\tau_l^*=\tau_{0l}^* \| \sigma_{0l}^*$ can be verified as a valid signature and $\tau_{0l}^* \notin \text{sList}$, the challenger aborts. The other part of Game 1 is kept unchanged.

**Claim 1:** $|\varepsilon_0-\varepsilon_1|$ is negligible under the assumption that the signature scheme $\sum$ is *existential unforgeable* [11] under the chosen message attack.

**Proof:** It is obvious that Game 0 and Game 1 proceed identically unless the event $E_1$ "any tag submitted by the adversary can be verified as a valid signature but is not on the list of authentication tags generated by the challenger" occurs. Hence $|\varepsilon_0 - \varepsilon_1| \le \Pr[E_1]$. But if this event happens with non-negligible probability, we can construct an adversary $B$ against the unforgeability of the signature scheme $\sum$.

$B$ takes a signing public key $spk \leftarrow \mathrm{SKg}(1^k)$ as input and is given access to a signing oracle $SSig_{ssk}(\cdot)$. $B$ picks a random $\alpha$ and sets the secret key $sk = (\alpha, \perp)$, the public key $pk = (v = g^{\alpha}, spk)$. It is clear that $B$ is able to simulates the view of $A$ in Game 1 perfectly with the help the signing oracle.

If the event $E_1$ happens, $B$ simply outputs the tag submitted by $A$ as its forgery against the signature scheme $\sum$.

Game 2 is almost the same as Game 1. The challenger in Game 2 verifies the response from the adversary during each execution of our MPOR protocol in a way different from the standard verification algorithm $MV(\cdot)$.

Recall that the challenger is able to determine the deterministic verification response $(\vec{\mu}, \sigma)$ returned by the honest prover algorithm $P(\cdot)$ by the modification in Game 1, given the corresponding MPOR query. Let $(\hat{\vec{\mu}}, \hat{\sigma})$ be the response returned by the adversary in one execution of our MPOR protocol : $< MV(pk, \{\tau_1, \cdots \tau_n\}) \rightleftarrows A > \rightarrow (b, \perp)$ . The challenger sets $d = 1$ and aborts if there is at least one execution $MV(pk, \{\tau_1, \cdots \tau_n\}, \hat{\vec{\mu}}, \hat{\sigma})$ outputs $b = 1$ and $\hat{\sigma} \ne \sigma$.

**Claim 2:** $|\varepsilon_2 - \varepsilon_1|$ is negligible under the CDH assumption.

**Proof:** Game 2 is distinct from Game 1 only when the response from the adversary in one execution of our MPOR protocol can pass the verification but is not equal to the correct response from the honest prover algorithm.

Let $E_2$ denote the event "The adversary in one execution of our MPOR protocol can pass the verification but $\hat{\sigma} \ne \sigma$". Game 2 is distinct from Game 1 only when $E_2$ happens. Hence $|\varepsilon_2 - \varepsilon_1| \le \Pr[E_2]$. If $\Pr[E_2]$ is non-negligible, we can construct a simulator that solves the CDH problem in the random oracle model.

The simulator $S$ takes an instance $(g, g^{\alpha}, h = g^{\beta})$ of the CDH problem as input and simulates the environment of Game 2 for the adversary $A$ as follows.

$S$ generates a signing key pair by running $(ssk, spk) \leftarrow \mathrm{SKg}(1^k)$ and sets $v \leftarrow g^{\alpha}$, which implicitly defines the secret key $sk = (\alpha, ssk)$ and the public key $pk = (v, spk)$. $S$ provides $pk$ to $A$.

To respond each query issued to the random oracle $H(\cdot)$, $S$ first parses it as $fn_l \| i$ and programs the response of $H(\cdot)$ as we will describe later.

To respond each query $M_l$ issued to the store oracle, $S$ first chooses a random file-name $fn_l$, encodes $M_l$ to obtain $\widehat{M_l}$ and splits it into $n_l$ blocks $\{m_{li}\}_{1\le i\le n_l}$. $S$ sets $u_l \leftarrow g^{\beta_l} h^{\gamma_l}, \gamma_l, \beta_l \leftarrow_R Z_p$. $S$ picks $r_{li} \leftarrow_R Z_p, 1 \le i \le n_l$ and programs the response of $H(fn_l \| i)$ as $H(fn_l \| i) = g^{r_{li}} \big/ (g^{\beta_l \cdot m_{li}} h^{\gamma_l \cdot m_{li}})$. At this point, $S$ computes $aug_{li}, 1 \le i \le n_l,$ as follows:

$$\begin{aligned} aug_{li} &= (H(fn_l \| i) \cdot u_l^{m_{li}})^{\alpha} = ((g^{r_{li}} \big/ (g^{\beta_l \cdot m_{li}} h^{\gamma_l \cdot m_{li}})) \cdot (g^{\beta_l} h^{\gamma_l})^{m_{li}})^{\alpha} \\ &= (g^{r_{li}})^{\alpha} = (g^{\alpha})^{r_{li}} \end{aligned}$$

$S$ computes $\tau_l$ according to the specification of the file-storing algorithm $St(\cdot)$ via the signing key and returns $(\widehat{M_l}, \tau_l, \{aug_{li}\}_{i=1}^{n_l})$ to $A$. $S$ interacts with $A$ until the event $E_2$ happens.

Assume that $Q = pk \cup \{P_1 \cup Q_1, \cdots, P_n \cup Q_n\}$, $P_l = (\lambda_l, fn_l)$, $Q_l = \{(l_i, v_{l_i})\}$ is the MPOR query issued by the verifier in one execution of our MPOR protocol for the encoded files $\widehat{M}_1, \cdots \widehat{M}_n$. Let the response returned by the adversary as a prover to this query be $\hat{\vec{\mu}}, \hat{\sigma}$. Let $(\vec{\mu}, \sigma)$ be the deterministic response returned by the honest prover algorithm for the query $Q$, which satisfies the following:

$$\vec{\mu} = (\mu_1, \cdots, \mu_l, \cdots, \mu_n), \quad \sigma = \prod_{l=1}^{n} \sigma_l^{\lambda_l}$$

$$\mu_l = \sum_{(j,v_j)\subset Q_l} v_j m_{lj} \quad \sigma_l = \prod_{(j,v_j)\subset Q_l} aug_{lj}^{\ v_j} \ , \ 1 \le l \le n$$

$$e(\sigma, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \,\|\, j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l}), v)$$

When $E_2$ happens, $\widehat{\vec{\mu}}, \widehat{\sigma}$ can also pass the verification and the following hold:

$$\widehat{\vec{\mu}} = (\widehat{\mu_1}, \cdots, \widehat{\mu_l}, \cdots, \widehat{\mu_n})$$

$$e(\widehat{\sigma}, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \,\|\, j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\widehat{\mu_l} \cdot \lambda_l}), v)$$

Let $\Delta\sigma = \widehat{\sigma} - \sigma$ , $\Delta\mu_l = \widehat{\mu_l} - \mu_l, 1 \le l \le n$ . If $\forall 1 \le l \le n, \widehat{\mu_l} = \mu_l$ , it follows that $\widehat{\sigma} = \sigma$ according to the verification equation. Consequently, $\Delta\mu_l \ne 0$ holds for at least one position $l$ by the assumption $\widehat{\sigma} \ne \sigma$. We derive the following by division:

$$e(\widehat{\sigma}/\sigma, g) = e((\prod_{l=1}^{n}(u_l^{\triangle\mu_l \cdot \lambda_l}), v) = e((\prod_{l=1}^{n}((g^{\beta_l} h^{\gamma_l})^{\triangle\mu_l \cdot \lambda_l}), v)$$

Rearranging terms yields $e((\widehat{\sigma}/\sigma) \cdot v^{-\sum_{1 \le l \le n} \beta_l \cdot \lambda_l \cdot \triangle\mu_l}, g) = e(h, v)^{\sum_{1 \le l \le n} \gamma_l \lambda_l \triangle\mu_l}$ .

As $v = g^{\alpha}, h = g^{\beta}$, we see that the solution $g^{\alpha\beta}$ to the CDH problem can be written as $((\widehat{\sigma}/\sigma) \cdot v^{-\sum_{1 \le l \le n} \beta_l \cdot \lambda_l \cdot \triangle\mu_l})^{\frac{1}{\sum_{1 \le l \le n} \gamma_l \lambda_l \triangle\mu_l}}$ unless $\sum_{1 \le l \le n} \gamma_l \lambda_l \triangle\mu_l$ is equal to zero.

For any fixed sequence $\{\triangle\mu_l\}_{l=1}^{n}$ that is not all zero, the probability that $\sum_{1 \le l \le n} \gamma_l \lambda_l \triangle\mu_l = 0$ is $1/p$ since each $\gamma_l$ chosen by the simulator is uniformly distributed over $Z_p$ and hidden from the adversary's view since $u_l = g^{\beta_l} h^{\gamma_l}$ reveals no information of $\gamma_l$ . Hence the success probability of $S$ solving the CDH problem is at least $\Pr[E_2] - 1/p$.

The challenger in Game 3 verifies the response from the adversary during each execution of our MPOR protocol in a way different from Game 2. Given a MPOR query $Q$, let $(\vec{\mu}, \sigma)$ be the deterministic response returned by the honest prover algorithm for the query

$Q$ and $(\overset{\rightharpoonup}{\widehat{\mu}}, \widehat{\sigma})$ be the response returned by the adversary in one execution of our MPOR protocol. The challenger parses $\overset{\rightharpoonup}{\widehat{\mu}}$ as $(\widehat{\mu_1}, \cdots, \widehat{\mu_l}, \cdots, \widehat{\mu_n})$ and aborts if $\exists l, \widehat{\mu_l} \neq \mu_l$, where $\mu_l = \sum_{(j,v_j) \subset Q_l} v_j m_{lj}, 1 \leq l \leq n$.

**Claim 3:** $|\varepsilon_3 - \varepsilon_2|$ is negligible under the discrete logarithm assumption.

**Proof:** Game 3 is distinct from Game 2 only when the response from the adversary in one of the MPOR protocol executions may cause the challenger to abort as specified. Let $E_3$ denote the event "The adversary behaving as a prover in one execution of our MPOR protocol can pass the verification as specified but $\exists l, \widehat{\mu_l} \neq \sum_{(j,v_j) \subset Q_l} v_j m_{lj}, 1 \leq l \leq n$".

Game 3 is distinct from Game 2 only when $E_3$ happens. Hence $|\varepsilon_3 - \varepsilon_2| \leq \Pr[E_3]$. If $\Pr[E_3]$ is non-negligible, we can construct a simulator that solves the discrete logarithm problem.

The simulator $S$ takes an instance $(g, h = g^{\beta})$ of the discrete logarithm problem as input and simulates the environment of Game 3 for the adversary $A$ as follows.

$S$ generates a signing keypair $(ssk, spk) \leftarrow \mathrm{SKg}(1^k)$ and picks $\alpha \leftarrow_R Z_p$. Let $v \leftarrow g^{\alpha}$, which defines the secret key $sk = (\alpha, ssk)$ and the public key $pk = (v, spk)$. $S$ provides $pk$ to $A$.

To respond each query $M_l$ issued to the store oracle, $S$ first chooses a random file-name $fn_l$ from a large domain, encodes $M_l$ to obtain $\widehat{M_l}$ and splits it into $n_l$ blocks $\{m_{li}\}_{1 \leq i \leq n_l}$. $S$ sets $u_l \leftarrow g^{\beta_l} h^{\gamma_l}, \gamma_l, \beta_l \leftarrow_R Z_p$.

$S$ interacts with the adversary until the event $E_3$ happens.

Assume $Q = pk \cup \{P_1 \cup Q_1, \cdots, P_n \cup Q_n\}$, $P_l = (\lambda_l, fn_l)$, $Q_l = \{(l_i, v_{l_i})\}$ is the MPOR query issued by the verifier in one execution of our MPOR protocol for the files $\widehat{M_1}, \cdots \widehat{M_n}$. The response returned by the adversary to this query is $(\overset{\rightharpoonup}{\widehat{\mu}}, \widehat{\sigma})$. Let $(\vec{\mu}, \sigma)$ be the deterministic response returned by the honest prover algorithm for the query $Q$.

According to the proof in Game 2, we know that $\widehat{\sigma} = \sigma$ except with negligible probability $negl(\lambda)$. Under this assumption, we derive the following according to the verification equation:

$$e(\sigma, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l}), v)$$

$$e(\widehat{\sigma}, g) = e(\prod_{l=1}^{n}(\prod_{(j,v_j)\subset Q_l} H(fn_l \| j)^{v_j})^{\lambda_l} \cdot \prod_{l=1}^{n}(u_l^{\widehat{\mu_l} \cdot \lambda_l}), v)$$

We conclude $\prod_{l=1}^{n}(u_l^{\mu_l \cdot \lambda_l}) = \prod_{l=1}^{n}(u_l^{\widehat{\mu_l} \cdot \lambda_l})$ by taking $\widehat{\sigma} = \sigma$, which means

$$1 = \prod_{l=1}^{n}(u_l^{\vartriangle\mu_l \cdot \lambda_l}) = \prod_{l=1}^{n}((g^{\beta_l} h^{\gamma_l})^{\vartriangle\mu_l \cdot \lambda_l})$$

Let $\Delta\mu_l = \widehat{\mu_l} - \mu_l, 1 \le l \le n$. $\Delta\mu_l \ne 0$ holds for at least one position $l$ by the assumption $\exists l, \widehat{\mu_l} \ne \sum_{(j,v_j)\subset Q_l} v_j m_{lj}, 1 \le l \le n$.

If $\sum_{1\le l\le n} \gamma_l \lambda_l \vartriangle\mu_l \ne 0 \bmod p$, the discrete logarithm $\beta = -(\frac{\sum_{1\le l\le n} \beta_l \lambda_l \vartriangle\mu_l}{\sum_{1\le l\le n} \gamma_l \lambda_l \vartriangle\mu_l}) \bmod p$ because we have $h = g^{-(\frac{\sum_{1\le l\le n} \beta_l \lambda_l \vartriangle\mu_l}{\sum_{1\le l\le n} \gamma_l \lambda_l \vartriangle\mu_l})}$.

Similarly, for any fixed sequence $\{\vartriangle\mu_l\}_{l=1}^{n}$ that is not all zero we can argue that the probability that $\sum_{1\le l\le n} \gamma_l \lambda_l \vartriangle\mu_l$ is equal to zero is $1/p$. Hence the success probability of $S$ solving the discrete logarithm problem is at least $\Pr[E_3] - 1/p - negl(\lambda)$.

Response of the adversary in Game 3 is forced to be the same as that output by the honest prover algorithm of our MPOR protocol. A ***well-behaved*** prover program $P^*$ causes the verification algorithm $MV(\cdot)$ to accept in each execution of MPOR protocol by responding with $(\vec{\mu}, \sigma)$ computed by the honest prover algorithm. **Claims** 1-3 show that any adversary that wins in the game $Exp_{MPOR}^{sound,n}(A, 1^k)$ is ***well-behaved*** except with negligible probability. In the following, we show that extraction will succeed by interacting with a ***well-behaved*** prover program.

**Definition 1:** Given a MPOR query $Q$ as input, a ***polite*** prover program $\widehat{P}$ outputs either the correct response computed by the honest prover algorithm or a special symbol $\perp$. If $\widehat{P}$ outputs the correct response with probability at least $\varepsilon$, then we call $\widehat{P}$ a $\varepsilon$-***polite*** prover program.

A $\varepsilon$-***well-behaved*** prover program $P^*$ can be transformed into $\varepsilon$-***polite*** prover program $\widehat{P}^{P^*}(\cdot)$ with black box access to $P^*$.

Having received a MPOR query $Q$ from a verifier, $\widehat{P}$ plays the role of verifier to interact with $P^*$ by forwarding the query $Q$ to $P^*$. Having received the response $(\vec{\mu},\sigma)$ from $P^*$, $\widehat{P}$ outputs $(\vec{\mu},\sigma)$ if and only if the verification algorithm $MV(pk,\{\tau_1,\cdots\tau_n\},(\vec{\mu},\sigma))$ outputs 1; otherwise $\widehat{P}$ outputs $\perp$. $\widehat{P}$ provides $P^*$ with fresh randomness and rewinds it for each interaction. As $P^*$ is $\varepsilon$-***well-behaved*** prover program, $\widehat{P}$ is $\varepsilon$-***polite***. Note that the tags $\{\tau_1,\cdots\tau_n\}$ that are responses to the store queries can help $\widehat{P}$ to verify the correctness of $(\vec{\mu},\sigma)$ returned by $P^*$.

Let $N=\sum_{1\le i\le n} n_i$. For a subspace $\Psi$ of $(Z_p)^N$, denote the dimension of $\Psi$ by $\dim\Psi$. Let $\mathrm{Free}\Psi$ be the indices of the canonical basis vectors $\{\vec{c_i}\}\in(Z_p)^N$ included in $\Psi$. In other words, $\mathrm{Free}\Psi=\{i\in[1..N]:\vec{c_i}\in\Psi\}$.

**Lemma 1 [ 16, Claim 4.6]:** For a subspace $\Psi$ of $(Z_p)^N$, and let $I$ be an $ns$-element subset of $[1..N]$. If $I\not\subset\mathrm{Free}\Psi$, then a random MPOR query $Q$ over indices in $I$ with its coefficient vector $\vec{q}\in\Psi$ occurs with probability at most $1/(p-1)$.

**Lemma 2 [16, Claim 4.7]:** Let $\#(\mathrm{Free}\Psi)=m$. For a random $ns$-element subset $I$ of $[1..N]$, the probability that $I\subseteq\mathrm{Free}\Psi$ is at most $m^{ns}/(N-ns+1)^{ns}$.

**Theorem 1:** Suppose that $\widehat{P}$ is a $\varepsilon$- ***polite*** prover program and let

$\omega = \dfrac{1}{p-1} + \dfrac{(\rho N)^{ns}}{(N-ns+1)^{ns}}$. If $\varepsilon > \omega$, for each file $\widehat{M_l}$, $1 \le l \le n$, the expected time to recover at least $\rho$ fraction of it is $O((1+\varepsilon \cdot N^2)\dfrac{\rho nN}{\varepsilon - \omega} + t^2 n^2)$, where $t$ is the number of MPOR queries issued to the prover program by the extractor during one round of interaction.

**Proof:** The $t$ MPOR query-response pairs during one round of interaction with the prover program $\widehat{P}$ contribute the following to the extractor's knowledge of the encoded files $\widehat{M_1^*}, \cdots, \widehat{M_n^*}$:

$$\overrightarrow{q^{(1)}} \cdot H = \overrightarrow{\mu^{(1)}}, \cdots, \overrightarrow{q^{(t)}} \cdot H = \overrightarrow{\mu^{(t)}}$$

$H$ is the matrix constructed from the encoded files $\widehat{M_1^*}, \cdots, \widehat{M_n^*}$ , which is described at the end of Section 4. We rewrite the above as $V \cdot H = W$ where $V$ is the $t \times N$ matrix whose row vectors are the $t$ coefficient vectors $\{\overrightarrow{q^{(i)}}\}_{i=1}^t$ of the MPOR queries and $W$ is the $t \times n$ matrix whose row vectors are $t$ corresponding MPOR responses $\{\overrightarrow{\mu^{(i)}}\}_{i=1}^t$. The matrix $V$ can be reduced to a matrix $G = U \cdot V$ in the row-reduced echelon form, where $U$ is a $t \times t$ matrix with nonzero determinant computed by applying Gaussian elimination to $V$.

The extractor's knowledge during the above interaction can be represented by the matrixs $G, U, V, W$, where the matrix $G$ in the row-reduced echelon form. The subspace generated by the matrix $G$ is denoted by $\Psi$. The extractor's knowledge space is initially empty. The extractor repeats the following behavior until $|\text{Free}\Psi| \ge \rho N$.

The extractor chooses a random MPOR query $Q$ over the indices $I \subseteq [1..N]$ with coefficient vector $\vec{q}$ by its random coins and runs the $\varepsilon$-*polite* $\widehat{P}$ on $Q$. $\widehat{P}$ answers the correct response $\vec{\mu} = \vec{q} \cdot H$ with probability $\varepsilon$. We consider the following three types:

1. $\vec{q} \notin \Psi$:

For queries of this type, the extractor extends its knowledge as follows:

It adds the row vector $\vec{q}$ to the current matrix $V$, obtaining $V'$ and adds the response vector $\vec{\mu}$ to the existing matrix $W$, obtaining $W'$. It also computes $G' = U' \cdot V'$ in the row-reduced echelon form. $G', U', V', W'$ represent the update of the extractor's knowledge.

2. $\vec{q} \in \Psi$ but $I \not\subset \mathrm{Free}\Psi$ :

3. $I \subseteq \mathrm{Free}\Psi$ :

For queries of type 2 or 3, the extractor does not update its knowledge and continue its current interaction with the prover program $\widehat{P}$.

A query of type 1 increases $\dim\Psi$ by 1. The extractor's interaction with $\widehat{P}$ is guaranteed to terminate when the number of queries of type 1 is above $\rho N$.

By Lemma 1, queries of type 2 make up at most $1/(p-1)$ since

$$\Pr_Q[Q \text{ is type } 2] = \Pr_Q[\vec{q} \in \Psi \wedge I \not\subset \mathrm{Free}\Psi]$$

$$= \Pr_Q[\vec{q} \in \Psi \mid I \not\subset \mathrm{Free}\Psi] \cdot \Pr_Q[I \not\subset \mathrm{Free}\Psi] \le \Pr_Q[\vec{q} \in \Psi \mid I \not\subset \mathrm{Free}\Psi] \le 1/(p-1)$$

Assume $\#(\mathrm{Free}\Psi) = m$. For a random $ns$- element subset $I$ of $[1..N]$, the probability that $I \subseteq \mathrm{Free}\Psi$ is at most $m^{ns}/(N-ns+1)^{ns}$ by Lemma 2. By the convention set for the extractor, $m \le \rho N$, this quantity is at most $(\rho N)^{ns}/(N-ns+1)^{ns}$.

Therefore the fraction of queries of type 2 or 3 is at most $\omega = \dfrac{1}{p-1} + \dfrac{(\rho N)^{ns}}{(N-ns+1)^{ns}}$.

As $\widehat{P}$ is a $\boldsymbol{\varepsilon}$***-polite*** prover program, $\widehat{P}$ is able to answer at least $\varepsilon$ fraction of the queries. Therefore, a random MPOR query chosen by the extractor will be of type 1 with probability at least $\varepsilon - \omega$. To yield $\rho N$ queries of type 1, the extractor will carry out $O(\dfrac{\rho N}{\varepsilon - \omega})$ interactions with $\widehat{P}$ in this round.

As the matrix $G$ in the row-reduced echelon form, it is possible to determine whether a query $Q$ is of type 1, to which $\widehat{P}$ has responded. The extractor adds the coefficient vector $\vec{q}$ of $Q$ to the current matrix $V$ and applies Gaussian elimination to $V$ to yield $G$, which takes $O(N^2)$ time [8]. If the newly added row is not all zeros, then $\vec{q}$ is of type 1. As Gaussian elimination need only be applied to at most $\varepsilon$ fraction of the queries responded correctly by $\widehat{P}$, the running time of the extractor in this phase is $O((1+\varepsilon \cdot N^2)\dfrac{\rho N}{\varepsilon - \omega})$.

On the other hand, when this phase has finished, the knowledge of the extractor consists of the matrixes $G,U,V,W$ such that $V \cdot H = W$, $G = U \cdot V$. $G$ is in the row-reduced echelon form. The free dimension of the subspace $\Psi$ spanned by $G$ is $\rho N$ by the conventions set for the extractor. For each $i \in \text{Free}\Psi$, there must be a row in $G$, say row $t$, that equals some canonical basis vector $\overrightarrow{c_i} \in (Z_p)^N$ since $G$ is in the row-reduced echelon form. Multiplying both sides of $V \cdot H = W$ by $U$, we obtain $G \cdot H = U \cdot W$.

We define a set $G_l = \{i : i \in \text{Free}\Psi, \sum_{1 \le i \le l-1} n_i < i \le \sum_{1 \le i \le l} n_i\}$ for each index $l$, $1 \le l \le n$. As the number of indices $i \in \text{Free}\Psi$ is $\rho N$, there exists at least one index $l$ such that $|G_l|$ is at least $\rho n_l$ by the pigeonhole principle. Recall that $H = [\overrightarrow{h_1}, \cdots, \overrightarrow{h_n}]$, $\overrightarrow{h_l} = (0^{\overrightarrow{\sum_{1 \le i \le l-1} n_i}}, (m_{l1}, \cdots, m_{ln_l}), 0^{\overrightarrow{\sum_{l+1 \le i \le n} n_i}})^T$. The inner products between these canonical vectors $\overrightarrow{c_i}, i \in G_l$ and the corresponding column vector $\overrightarrow{h_l}$ will recover at least $\rho$ fraction of the encoded file $\widehat{M_l^*}$, which can extracted from the product $U \cdot W$. The computation will takes $O(t^2 n)$ time [8].

In the following, we analyze the probability $p_l$ of an event $F_l$ " $|G_l| \ge \rho n_l$ for the given index $l, 1 \le l \le n$".

At first, we will analyze the probability $p_l^/$ of an event " $|G_l| = \rho n_l$ for the given index $l, 1 \le l \le n$".

As mentioned above, we assume that each file's size is of the same magnitude. For ease of analysis, we further assume $b = n_1 = \cdots = n_l$. For instance, these files of the same magnitude can be redundantly padded before encoding.

The probability $p_l^/ = \binom{n_l}{\rho n_l}\binom{N - n_l}{\rho(N - n_l)} \Big/ \binom{N}{\rho N} = \binom{b}{\rho b}\binom{nb - b}{\rho(nb - b)} \Big/ \binom{nb}{\rho nb}$

The probability $p_l = \sum_{\rho b \le j \le \rho nb} \binom{b}{j}\binom{nb - j}{\rho nb - j} \Big/ \binom{nb}{\rho nb}$, only depends on the values of $\rho, n, b$ and is independent of the choice of $l$.

As $\Pr[F_1 \vee, \cdots, \vee F_n] = 1, 1 = \Pr[F_1 \vee \cdots, \vee F_n] \le \sum_{l=1}^{n} F_l = n \cdot p_l$ by the union bound. We know that $\Pr[F_l] \geq \frac{1}{n}$. Hence the extractor can recover at least $\rho$ fraction of the encoded file $\widehat{M_l^*}$ with probability at least $1/n$ during this phase. This means at least $\rho$ fraction of the file $\widehat{M_l^*}$ can be recovered by running the extractor algorithm $n$ rounds on average, who rewinds the $\varepsilon$-***polite*** prover $\widehat{P}$ with fresh coins before each interaction.

In conclusion, for each file $\widehat{M_l^*}$, $1 \leq l \leq n$, the expected time to recover at least $\rho$ fraction of it is $O((1+\varepsilon \cdot N^2)\frac{\rho n N}{\varepsilon - \omega} + t^2 n^2)$. As $\widehat{M_l^*}$ is redundantly encoded by erasure codes [2] such that any $\rho$ fraction of the $\widehat{M_l^*}$ is sufficient to recover the original file $M_l^*$. The original file $M_l^*$ is guaranteed to be recovered in this case.

## 6. Performance analysis

We evaluate the performance of the proposed MPOR protocol and that of the POR protocol [16] in terms of the required communication and computational cost to verify $n$ files stored on the server side. The result is stated in Table 1, 2. Pair denotes one pairing operation. $MExp^n(G)$ denotes one general multi-exponentiation $g_1^{e_1} \cdots g_n^{e_n}$ over a group $G$. The size of each set $Q_l$ is assumed to be a fixed $s$ in both schemes. For ease of comparison, we assume that there is only one sector per each encoded message block in the POR protocol [16].

The advantage of our MPOR protocols lies in the fact that the number of pairing operations computed by the verifier is independent of the parameter $n$. In addition, length of the response from a prover is further reduced by aggregating the responses from the prover.

Nevertheless, the soundness of our MPOR protocol only satisfies a relatively weak security notion under the assumption that each file's size is of the same magnitude. One invocation of the extractor algorithm can only guarantee that at least one file among those $n$ files can be recovered. Our analysis shows that each file can be extracted in expected polynomial time under our assumption on the size of processed files.

## 7. Conclusion

Proof-of-retrievability protocols can help a client to be assured of the availability of files stored by a server. H. Shacham et al. [16] strengthened the security model for POR protocols by allowing an extractor black-box access to a prover program. In addition, they presented a POR protocol with public verifiability based on a homomorphic authenticator derived from BLS short signature. When using their POR protocol with public verifiability to verify the availability of multiple files separately, the number of pairing operations computed by a verifier is linear with the number of files. To handle this issue, we extend the work in [16] by introducing a new notion called multi-proof-of-retrievability. Our MPOR protocol with public verifiability allows one verifier to verify the availability of $n$ files stored by a server in one pass, while the computational overhead of a verifier in our MPOR scheme is constant, independent of the parameter $n$. Analysis of our MPOR protocol shows that each file can be extracted in expected polynomial time under certain restriction on the size of processed files.

**Acknowledgement**

This work is supported by Natural Science Foundation of Higher Education Institutions, in Jiangsu Province office of education, P.R. China (Grant No. 10KJD520005)

## References

[1] Amazon simple storage service (Amazon S3), http://aws.amazon.com/s3/

[2] A.Alon and M.Luby, A linear time erasure-resilient code with near optimal recovery, IEEE.Tran.Inf.Theory,42(6) (1996) 1732-1736.

[3] G. Ateniese, R.Burns, R.Curtmola, J.Herring, O.Khan, L.Kissner, Z.Peterson and D. Song, Remote data checking using provable data possession, ACM Trans.Inf.Syst.Security, 14(1), (2011) Article No.12,

[4] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, Scalable and efficient provable data possession, Cryptology ePrint Archive, Report 2008/114, 2008. http://eprint.iacr.org/2008/114

[5] M. Bellare and O. Goldreich, On defining proofs of knowledge, CRYPTO'92, LNCS 740 (1993) 390-420.

[6] D. Boneh, B.Lynn and H. Shacham, Short signatures from weil pairing, Journal of Cryptology, 17(4) (2004) 297-319.

[7] D. Cash, A. Kupcu, and D. Winchs, Dynamic Proofs of Retrievability via Oblivious RAM, Cryptology ePrint Archive, http://eprint.iacr.org/2012/550

[8] H. Cohen, A course in computational algebraic number theory, Berlin, Springer, 1993

[9] Y. Deswarte, J.-J.Quisquarter and A. Saidaneand H. Shacham, Remote integrity checking, IICIS 2003, IFIP Vol.140 (2004) 1-11.

[10] C. C. Erway, A. Kuppcu, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. ACM CCS 09, (2009) 213-222.

[11] S. Goldwasser, S.Micali and R. Rivest, A digital signature scheme secure against adaptive chosen-message attacks, SIAM Journal of Computing, 17(2) (1988) 281-308.

[12] A. Juels and B.Kaliski, PORs: proofs of retrievability for large files, CCS 2007, ACM, (2007) 584-597.

[13] P. Mell and T. Grance, NIST SD 800-145, The NIST definition of cloud computing, NIST special publication, 2011

[14] M. Naor and G.Rothblum, The complexity of online memory checking, J.ACM, 56(1), (2009) Article No.2

[15] T. Schwarz and E. Miller, Store, forget and check: Using algebraic signatures to check remotely administered storage, ICDCS 2006, IEEE, (2004) 1-11.

[16] H. Shacham and B.Waters, Compact proofs of retrievability, Journal of Cryptology, published online

[17] M.Shah, M.Baker, J.Mogul and R.Swaminathan, Auditing to keep online storage service honest, Proceedings of HotOS 2007, ACM, (2007) Article No.11.

[18] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, Enabling public verifiability and data dynamics for storage security in cloud computing, ESORICS 2009, Springer, LNCS 5789, (2009) 355-370.

Table 1. Computational overhead of a verifier

| Scheme | Cost for verifying $n$ files | |
|---|---|---|
| | Pairing | Exponentiation |
| POR protocol [16] | $2n$ | $2n \cdot MExp^{s-1}(G)$ |
| Our MPOR protocol | 2 | $n \cdot MExp^{s-1}(G) + MExp^{n-1}(G)$ |

Table 2. Length of the response from a prover

| Scheme | Total bit length of the $n$ responses from the prover |
|---|---|
| POR protocol [16] | $n \cdot (\vert Z_p \vert + \vert G \vert)$ |
| Our MPOR protocol | $n \cdot (\vert Z_p \vert) + \vert G \vert$ |