Towards Efficient Proofs of Retrievability

Jia Xu [*]
Institute for Infocomm Research, Singapore
xuj@i2r.a-star.edu.sg

Ee-Chien Chang [†]
National University of Singapore
changec@comp.nus.edu.sg

Abstract

Proofs of Retrievability ($\mathcal{POR}$) is a cryptographic formulation for remotely auditing the integrity of files stored in the cloud, without keeping a copy of the original files in local storage. In a $\mathcal{POR}$ scheme, a user Alice backups her data file together with some authentication data to a potentially dishonest cloud storage server Bob. Later, Alice can periodically and remotely verify the integrity of her data file using the authentication data, without retrieving back the data file. Besides security, performances in communication, storage overhead and computation are major considerations. Shacham and Waters (Asiacrypt '08) gave a fast scheme with $\mathcal{O}(s\lambda)$ bits communication cost and a factor of $1/s$ file size expansion where λ is the security parameter. In this paper, we incorporate a recent construction of constant size polynomial commitment scheme (Kate, Zaverucha and Goldberg, Asiacrypt '10) into Shacham and Waters scheme. The resulting scheme requires $\mathcal{O}(\lambda)$ communication bits (particularly, 920 bits if a 160 bits elliptic curve group is used or 3512 bits if a 1024 bits modulo group is used) per verification and a factor of $1/s$ file size expansion. Experiment results show that our proposed scheme is indeed efficient and practical. Our security proof is based on Strong Diffie-Hellman Assumption.

Categories and Subject Descriptors H.3.2 [*Information Storage and Retrieval*]: Information Storage; D.4.6 [*Security and Protection*]: Cryptographic controls

General Terms Storage, Integrity, Security, Algorithm

Keywords Cloud Storage, Proofs of Retrievability, Remote Data Integrity Check, Homomorphic Authentication Tag, Polynomial Commitment

[*] Xu is partially supported by Grant SecDC-112172014.

[†] Chang is partially supported by Grant TDSI/09-003/1A.

ASIACCS '12, May 2–4, 2012, Seoul, Korea.

1. Introduction

Storing data in a cloud storage, for example Amazon Cloud Drive, Microsoft Skydrive, or Dropbox, is gaining popularity recently. We are considering scenarios where users may have concerns of the integrity of their data stored in the cloud storage. Such prudent users may have doubts about the cloud storage server's promise on maintaining the data integrity. Instead, they desire a technical way to be assured that the the cloud storage server is keeping his promise and following the service level agreement (SLA).

Threats to integrity of data stored in cloud is indeed realistic. Several events about data loss in Amazon cloud storage have been reported [16, 27]. There are also plenty of data loss cases that are claimed by individuals but not confirmed officially by the cloud server, e.g. data loss cases in Dropbox [14].

Proofs of Retrievability ($\mathcal{POR}$) model proposed by Juels and Kaliski [21] is among the first few attempts to formulize the notion of "remotely and reliably verifying the data integrity without retrieving the data file". A $\mathcal{POR}$ scheme consists of setup phase and a sequence of verification phases. In the setup phase, a data owner Alice preprocesses her data file using her private key to generate some authentication information. Then Alice sends the data file together with authentication information to the potentially dishonest cloud storage server Bob, and removes them from her local storage. Consequently, in the end of setup phase, Alice only has her private key in her local storage, and Bob has both the data file and the corresponding authentication information. In each subsequent verification phase, Alice generates a random challenge query and Bob is supposed to produce a short response upon the received challenge query, based on Alice's data file and the corresponding authentication information. In the end of a verification phase, Alice will verify Bob's response using her private key and decide to *accept* or *reject* this response.

The performance of a $\mathcal{POR}$ scheme is determined by a few factors: the number of communication bits (i.e. the bit lengths of a challenge and a response) exchanged between Alice and Bob per verification, storage overhead on Alice/Bob's side, computation time on Alice/Bob's side in a verification, and computation time for data preprocessing in setup on Alice side.

Shacham and Waters [26] proposed two $\mathcal{POR}$ schemes, where one supports private verifiability and the other supports public verifiability. In this paper, we are interested in the $\mathcal{POR}$ scheme [26] with private verifiability and refer to it as SW scheme. In this SW scheme, the size of a response (or proof) is dominated by s group elements where each is λ bits long. We manage to aggregate these s group elements into two group elements, leading to a reduction in proof size from $\mathcal{O}(s\lambda)$ bits to $\mathcal{O}(\lambda)$ bits, by exploiting an intriguing property of polynomial, which is recently used by Kate, Zaverucha and Goldberg [22] to construct a polynomial commitment. Combining with the result of Dodis, Vadhan and Wichs [13], which reduced the challenge size of SW scheme from $\mathcal{O}(\lambda^2)$ to $\mathcal{O}(\lambda)$, we reduce the communication cost per verification of SW scheme from $\mathcal{O}(\lambda^2 + s\lambda)$ to $\mathcal{O}(\lambda)$.

1.1 Our Contributions

Our contributions can be summarized as below: We propose a new efficient $\mathcal{POR}$ scheme in Section 5. In this scheme, a data block consists of s group elements and a subset of ℓ data blocks are accessed in each verification. The storage overhead is $1/s$ of the data file size, and communication cost is $\mathcal{O}(\lambda)$ bits per verification where λ is the security parameter, and the computation cost is $\mathcal{O}(s)$ group exponentiations on the server side (prover) and $\mathcal{O}(\ell)$ group addition/multiplication/PRF on the client (verifier) side. We prove that the proposed $\mathcal{POR}$ scheme is secure (Theorem 1 in Section 5.2) under Strong Diffie-Hellman Assumption.

The empirical study in Section 6 shows that our scheme is practical in computation under reasonable setting. In a typical setting where an elliptic curve group of prime order p (the bit-length of p is $\lambda = 160$) is used and the system parameter $s = 100$, during each verification, 920 communication bits are exchanged between the data owner and the cloud storage server where 440 bits for challenge and 480 bits for response, and 100 elliptic curve exponentiations are required to generate a response on the cloud storage server side. The small number of communication bits is also desirable in situations where the challenge and response could be piggybacked into other communication packets between the data owner and the cloud storage server.

1.2 Organization

The rest of this paper is organized as below: We brief the background on SW scheme [26] and review the related works in Section 2. Particularly, in Section 2.1.3, we overview our proposed scheme in the context of background works [13, 26]. Section 3 reviews the security formulation of Proofs of Retrievability. We describe background knowledge on the polynomial commitment scheme [22] in Section 4 and present our main scheme in Section 5. We analyze the performance of the proposed scheme, conduct experiments and report the empirical results in Section 6. After that, Section 7 concludes this paper.

Section 5 is self-contained and readers may jump to Section 5 directly to read our main construction.

2. Background and Related work

In this section, we brief the background on Shacham and Waters [26] $\mathcal{POR}$ scheme and improvement on this scheme by Dodis, Vadhan and Wichs [13], and review related works on proofs of storage. We also overview our result in this paper, based on works by Shacham and Waters [26] and Dodis *et al.* [13].

2.1 SW scheme and Our Improvement

2.1.1 Original SW scheme

Shacham and Waters [26] proposed two $\mathcal{POR}$ schemes, where one supports private verifiability and the other supports public verifiability. Here we adopt the interpretation of [13] to review the SW $\mathcal{POR}$ [26] scheme with private verifiability.

In the setup, the data owner Alice encodes her data file using some error erasure encoding scheme (e.g. Reed-Solomon code [25]), such that the encoded file consists of many blocks and each block is a vector of s group elements from $\mathbb{Z}_p$, where p is a λ bits long prime. Next, Alice generates an authentication tag for each block in the encoded file, using a homomorphic linear authenticator. Alice then sends the encoded file together with all authentication tags to the potentially dishonest cloud storage server Bob. Thus, at the end of setup, Alice has only a private key in her local storage, and Bob has Alice's encoded data file and authentication tags. Since an authentication tag is a group element in $\mathbb{Z}_p$, the size of all authentication tags is $1/s$ of the size of encoded file. The error erasure encoding in the setup is called as *primary encoding* (or *initial encoding*) by Dodis *et al.* [13].

In a verification, the data owner Alice, who is taking the role of verifier, interacts with cloud storage server Bob, who is taking the role of prover. Alice chooses a random challenge, which consists of two parts: The first part C is a subset of block indices that specifies a random subset of ℓ blocks in the encoded file. Let us denote the selected blocks as ℓ vectors $\{\vec{\boldsymbol{x}}_i \in \mathbb{Z}_p^s : 0 \leq i \leq \ell - 1\}$. The second part is an ℓ-dimensional vector $\vec{\boldsymbol{\nu}} = (\nu_0, \ldots, \nu_{\ell-1}) \in \mathbb{Z}_p^\ell$. Bob computes linear combination $\vec{\boldsymbol{\mu}} := \sum_{i=0}^{\ell-1} \nu_i \vec{\boldsymbol{x}}_i \in \mathbb{Z}_p^s$, and an authentication tag $\sigma_{\vec{\boldsymbol{\mu}}}$ of $\vec{\boldsymbol{\mu}}$ using the linear homomorphism of the homomorphic linear authenticator. Bob sends $(\vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}})$ to Alice as response and Alice checks the validity of the received message-tag pair. Dodis *et al.* [13] pointed out that, for each $j \in [0, s-1]$, the j-th component of $\vec{\boldsymbol{\mu}}$ is just a Hadamard codeword with parameter $\vec{\boldsymbol{\nu}}$ of j-th components of selected blocks $\vec{\boldsymbol{x}}_i$:

$$\vec{\boldsymbol{\mu}}[j] = \langle \vec{\boldsymbol{\nu}},\ (\vec{\boldsymbol{x}}_0[j], \ldots, \vec{\boldsymbol{x}}_{\ell-1}[j]) \rangle = \sum_{i=0}^{\ell-1} \nu_i \vec{\boldsymbol{x}}_i[j].$$

They [13] called this Hadamard coding as *secondary encoding*. In contrast, in the MAC based $\mathcal{POR}$ scheme implied in [23], the prover just returns the long message $\{\vec{\boldsymbol{x}}_i \in \mathbb{Z}_p^s : 0 \leq i \leq \ell - 1\}$ and corresponding authentication information, without secondary encoding.

In each verification, the challenge size is in $\mathcal{O}(\lambda^2)$: the first part C of challenge is of size $\mathcal{O}(\ell \log n)$ and the second part $\vec{\boldsymbol{\nu}}$ is of size $\mathcal{O}(\ell\lambda)$, where $\ell = \mathcal{O}(\lambda)$ is the number of blocks selected in each verification, and $n = poly(\lambda)$ is the number of blocks in the encoded file. The response size is in $\mathcal{O}(s\lambda)$: the message $\vec{\boldsymbol{\mu}} \in \mathbb{Z}_p^s$ and the tag $\sigma_{\vec{\boldsymbol{\mu}}} \in \mathbb{Z}_p$, where $\lambda \approx \log p$.

2.1.2 Improved SW scheme

Dodis, Vadhan and Wichs [13] reduces the size of challenge in SW scheme from $\mathcal{O}(\lambda^2)$ to $\mathcal{O}(\lambda)$. They made two modifications: Firstly, instead of choosing the subset C in a challenge uniformly randomly, they [13] samples C using a (δ, γ)-*hitter* given by Goldreich [19]. Consequently, the size of the first part of challenge is reduced to $\mathcal{O}(\lambda)$. Secondly, the Hadamard code is replaced with Reed-Solomon code. That is, the vector $\vec{\boldsymbol{\nu}} = (\nu_0, \ldots, \nu_{\ell-1})$ is replaced by vector $(\nu^0, \nu^1, \ldots, \nu^{\ell-1})$ for some $\nu \in \mathbb{Z}_p$, and the second part of a challenge is changed from vector $(\nu_0, \ldots, \nu_{\ell-1})$ to a single group element ν.

However, the size of response in a verification is still unchanged.

2.1.3 Overview of Our result

Based on works of [13, 26], in this paper, we apply Reed-Solomon code *again* on the response $(\vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}})$ of SW scheme [13, 26] as *tertiary encoding*: the new challenge will contain a group element $r \in \mathbb{Z}_p$ in addition, and the new response will be $(\langle \vec{\boldsymbol{\mu}},\ \vec{\boldsymbol{r}} \rangle, \sigma')$, where $\vec{\boldsymbol{r}} = (r^0, \ldots, r^{s-1})$ and the authentication tag σ' of the inner product $\langle \vec{\boldsymbol{\mu}},\ \vec{\boldsymbol{r}} \rangle$ is computed from $(\vec{\boldsymbol{\mu}}, \sigma_{\vec{\boldsymbol{\mu}}})$ using the idea of [22] (We will review [22] later in Section 4).

As a result, this paper further reduces the response size in a verification of SW scheme from $\mathcal{O}(s\lambda)$ to $\mathcal{O}(\lambda)$, where the size of all authentication tags is unchanged—still $1/s$ of the size of encoded file, the same as the original SW scheme [26].

The asymptotic complexities in communication and storage of the resulting $\mathcal{POR}$ scheme will match the Provable Data Possession ($\mathcal{PDP}$) scheme proposed by Ateniese *et al.* [1, 3], where it is well known that $\mathcal{PDP}$ is a weaker security model than $\mathcal{POR}$. Compared to the $\mathcal{PDP}$ scheme given by Ateniese *et al.* [1, 3], on the dark side, our proposed scheme has a longer public key ($\mathcal{O}(s\lambda)$ bits versus $\mathcal{O}(\lambda)$ bits); on the bright side, our proposed scheme is much more practical in computation in the setup. Essentially, Ateniese *et al.* [1, 3] requires one group multiplication per each bit of the data file in the setup, while the proposed scheme requires one group multiplication per each chunk of data bits (160 bits per chunk if elliptic curve is used or 1024 bits per chunk if modulo group is used).

The costs of our modifications are: (1) the computation of the authentication tag σ' of the inner product $\langle \vec{\boldsymbol{\mu}},\ \vec{\boldsymbol{r}} \rangle$ requires s group exponentiations in $\mathbb{Z}_p$; (2) the per-user public key[1] consists of s group elements from $\mathbb{Z}_p$. We will measure the computation time of our proposed scheme through experiments; the size of a public key is about 100 kilobits (Kbit) when $s = 100$ and $\lambda = 1024 \approx \log p$.

We remark that our proposed scheme can be alternatively instantiated over an elliptic curve group (bilinear map is not required) with small size (e.g. 2^{160}).

2.2 Related work

Recently, there are extensive studies [1–3, 7–10, 13, 15, 20, 21, 26, 28, 29] on remote data integrity check. Juels and Kaliski [21] presented a strong security model called "Proofs of Retrievability" ($\mathcal{POR}$) and a $\mathcal{POR}$ scheme with bounded-use, and Ateniese *et al.* [1, 3] gave an efficient scheme which is secure under a weaker "Provable Data Possession" ($\mathcal{PDP}$) model. Efficient methods using some sorts of homomorphic authentication tag are proposed in [2, 9, 26]. Dynamic-$\mathcal{PDP}$ [15] extends to dynamic setting, public verifiability is exploited in Shacham and Waters [26] and Wang *et al.* [29], and the privacy issue in public verification is studied in Wang *et al.* [28]. Readers can refer to Yang and Jia [30] for a survey of secure cloud storage.

The most efficient variant scheme E-PDP in Ateniese *et al.* [1] suffers from the attack by Shacham and Waters [26]. In the main scheme of Ateniese *et al.* [1], the prover is required to compute the product $\prod_{(i,a_i)\in \mathtt{Chal}} \mathtt{T}_i^{a_i}$ for all tags $\mathtt{T}_i$ selected by the challenge `Chal`. The authors proposed an efficient variant scheme, named E-PDP, by setting all coefficients a_i in the challenge `Chal` as 1, so that only multiplication is involved and expensive exponentiation is avoided. Shacham and Waters [26] presented an attack on E-PDP, such that the adversary can answer correctly a non-negligible fraction of queries, but there exists no extractor that can recover any data block.

Dodis *et al.* [13] reduces the communication cost of SW scheme from $\mathcal{O}(\lambda^2)$ to $\mathcal{O}(\lambda)$ in the special case where $s = 1$ and the size of all authentication tags is equal to the size of error erasure encoded file.

Kate *et al.* [22] proposed an efficient commitment scheme for polynomial and Benabbas *et al.* [4] proposed a verifiable delegation scheme for polynomial. Both schemes alone can be extended to support $\mathcal{POR}$ easily but with limitations: The $\mathcal{POR}$ scheme in Benabbas *et al.* [4] requires only $\mathcal{O}(\lambda)$ bits response per verification due to their newly constructed verifiable delegation scheme for polynomial. However, in this scheme, the size of all authentication tags is also equal to the size of error erasure encoded file. In a $\mathcal{POR}$ scheme implied by Kate *et al.* [22], either every bit in the data

[1] In contrast, the public key of Ateniese *et al.* [1] is only $\mathcal{O}(\lambda)$ bits long.

Notation	Semantics
$x \xleftarrow{\$} S$	Uniformly randomly choose x from the finite set S.
λ	Bit-length of group size, i.e. group size is 2^λ.
s	The number of group elements in a data block. Typically, an authentication tag consists of one group element. So s is also the ratio of the size of a data block to the size of an authentication tag.
ℓ	The number of data blocks accessed during a verification.
n	The number of data blocks in an encoded data file. Thus the size of an encoded data file is $ns\lambda$ bits.
$\vec{\boldsymbol{m}}$	A vector of form $(m_0, m_1, m_2, \ldots, m_{d-1})$, where d is the dimension of vector $\vec{\boldsymbol{m}}$.
$f_{\vec{\boldsymbol{m}}}(x)$	A polynomial $m_0 + m_1 x + m_2 x^2 + \ldots + m_{d-1} x^{d-1}$ of degree $d-1$ with vector $\vec{\boldsymbol{m}}$ as coefficient, where d is the dimension of vector $\vec{\boldsymbol{m}}$.
PRF	Pseudorandom function [18].
$\mathcal{POR}$	Proofs of Retrievability [21].
$\mathcal{PDP}$	Provable Data Possession [1].
EPOR	Efficient Proofs of Retrievability; it is the name of the scheme proposed in this paper.

Table 1. Summary of Key Notations in this paper.

file has to be accessed during each verification, or there is linear storage overhead w.r.t. the number of file blocks on data owner side. We will elaborate more on the polynomial commitment scheme [22] in Section 4.

3. Formulation

The key notations used in this paper are summarized in Table 1.

3.1 System Model

We restate the $\mathcal{POR}$ [21, 26] model as below, with slight modifications on notations. We adopt the 1-round prove-verify version in Juels and Kaliski [21] for simplicity.

DEFINITION 1 ($\mathcal{POR}$ [21, 26]). *A* Proofs Of Retrievability *($\mathcal{POR}$) scheme consists of four algorithms* (KeyGen, DEncode, Prove, Verify)*:*

- $\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$*: Given security parameter λ, the randomized key generating algorithm outputs a public-private key pair (pk, sk).*
- $\mathsf{DEncode}(sk, \mathbf{M}) \rightarrow (\mathtt{id}_\mathbf{M}, \hat{\mathbf{M}})$*: Given the private key sk and a data file $\mathbf{M}$, the encoding algorithm* DEncode *produces a unique identifier $\mathtt{id}_\mathbf{M}$ and the encoded file $\hat{\mathbf{M}}$.*
- $\mathsf{Prove}(pk, \mathtt{id}_\mathbf{M}, \hat{\mathbf{M}}, C) \rightarrow \psi$*: Given the public key pk, an identifier $\mathtt{id}_\mathbf{M}$, an encoded file $\hat{\mathbf{M}}$, and a challenge query C, the prover algorithm* Prove *produces a proof ψ.*
- $\mathsf{Verify}(sk, \mathtt{id}_\mathbf{M}, C, \psi) \rightarrow$ `accept` *or* `reject`*: Given the private key sk, an identifier $\mathtt{id}_\mathbf{M}$, a challenge query C, and a proof ψ, the deterministic verifying algorithm* Verify *will output either* accept *or* reject.

Completeness. A $\mathcal{POR}$ scheme (KeyGen, DEncode, Prove, Verify) is *complete*, if an honest prover (who ensures the integrity of his storage and executes the procedure Prove to compute a proof) will always be accepted by the verifier. More precisely, for any key pair (pk, sk) generated by KeyGen, and any data file $\mathbf{M}$, any challenge query C, if $\psi \leftarrow \mathsf{Prove}(pk, \mathtt{id}_\mathbf{M}, \hat{\mathbf{M}}, C)$, then $\mathsf{Verify}(sk, \mathtt{id}_\mathbf{M}, C, \psi)$ outputs accept with probability 1, where $(\mathtt{id}_\mathbf{M}, \hat{\mathbf{M}}) \leftarrow \mathsf{DEncode}(sk, \mathbf{M})$.

3.2 Security Model

3.2.1 Trust Model and Scope of Topic

In a $\mathcal{POR}$ system, only the data owner is trusted and the cloud storage server is treated as untrusted and potentially malicious.

The following topics are out of the scope of this paper, since most existing techniques in these topics can be applied with our work: (1) Support of dynamic operations (e.g. insertion or deletion of a data block after setup); (2) Denial of Service Attack; (3) Frame attack where dishonest data owner claims that an honest cloud storage server was cheating.

3.2.2 $\mathcal{POR}$ Security Game

We rewrite the $\mathcal{POR}$ security game in [21, 26] in a standard way. The security game between a *probabilistic polynomial time* (PPT) adversary $\mathcal{A}$ and a PPT challenger $\mathcal{C}$ w.r.t. a $\mathcal{POR}$ scheme $\mathcal{E} = (\mathsf{KeyGen}, \mathsf{DEncode}, \mathsf{Prove}, \mathsf{Verify})$ is as below.

Setup: The challenger $\mathcal{C}$ runs the key generating algorithm KeyGen to obtain public-private key pair (pk, sk), and gives pk to the adversary $\mathcal{A}$.

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

- Store query $(\mathbf{M})$: Given a data file $\mathbf{M}$ chosen by $\mathcal{A}$, the challenger $\mathcal{C}$ responds by running data encoding algorithm $(\mathtt{id}, \hat{\mathbf{M}}) \leftarrow \mathsf{DEncode}(sk, \mathbf{M})$ and sending the encoded data file $\hat{\mathbf{M}}$ together with its identifier $\mathtt{id}$ to $\mathcal{A}$.
- Verification query $(\mathtt{id})$: Given a file identifier $\mathtt{id}$ chosen by $\mathcal{A}$, if $\mathtt{id}$ is the first part of output of some previous store query that $\mathcal{A}$ has made, then the challenger $\mathcal{C}$ initiates a $\mathcal{POR}$ verification with $\mathcal{A}$ w.r.t. the data file $\mathbf{M}$ associated to the identifier $\mathtt{id}$ in this way:
  - $\mathcal{C}$ chooses a random challenge `Chal`;
  - $\mathcal{A}$ produces a proof ψ w.r.t. the challenge `Chal`; *Note: adversary $\mathcal{A}$ may generate the proof in an arbitrary method rather than applying the algorithm* Prove*.*
  - $\mathcal{C}$ verifies the proof ψ by running algorithm $\mathsf{Verify}(sk, \mathtt{id}, \mathtt{Chal}, \psi)$. Denote the output as b.

  At the end $\mathcal{C}$ sends the decision bit $b \in \{\mathsf{accept}, \mathsf{reject}\}$ to $\mathcal{A}$ as feedback. Otherwise, if $\mathtt{id}$ is not in the output of any previous store query that $\mathcal{A}$ has made, $\mathcal{C}$ does nothing.

Commit: Adversary $\mathcal{A}$ chooses a file identifier $\mathtt{id}^*$ among all file identifiers she obtains from $\mathcal{C}$ by making store queries in **Learning** phase, and commit $\mathtt{id}^*$ to $\mathcal{C}$. Let $\mathbf{M}^*$ denote the data file associated to identifier $\mathtt{id}^*$.
Retrieve: The challenger $\mathcal{C}$ initiates ζ number of $\mathcal{POR}$ verifications with $\mathcal{A}$ w.r.t. the data file $\mathbf{M}^*$, where $\mathcal{C}$ plays the role of verifier and $\mathcal{A}$ plays the role of prover, as in the **Learning** phase. From messages collected in these ζ interactions with $\mathcal{A}$, $\mathcal{C}$ extracts a data file $\mathbf{M}'$ using some PPT extractor algorithm. The adversary $\mathcal{A}$ wins this game, if and only if $\mathbf{M}' \neq \mathbf{M}^*$.

The adversary $\mathcal{A}$ is ϵ-*admissible* [26], if the probability that $\mathcal{A}$ convinces $\mathcal{C}$ to accept in a verification in the **Retrieve** phase, is at least ϵ. We denote the above game as $\mathsf{Game}^{\mathcal{E}}_{\mathcal{A}}(\zeta)$.

DEFINITION 2 ([21, 26]). *A $\mathcal{POR}$ scheme $\mathcal{E}$ is sound, if for any PPT ϵ-admissible adversary $\mathcal{A}$ with non-negligible ϵ, there exists a polynomial ζ, such that the advantage $\mathsf{Adv}^{\mathcal{E}}_{\mathcal{A}}(\zeta)$ defined as below is negligible.*

$$\mathsf{Adv}^{\mathcal{E}}_{\mathcal{A}}(\zeta) \stackrel{\text{def}}{=} \mathsf{Pr}\left[\mathcal{A} \textit{ wins } \mathsf{Game}^{\mathcal{E}}_{\mathcal{A}}(\zeta)\right] \tag{1}$$

3.3 Assumption

DEFINITION 3 (s-SDH Assumption [5, 6]). *Let p and $q = 2p + 1$ be prime, and $\mathcal{QR}$ be the subgroup with order p of quadratic residues in $\mathbb{Z}^*_q$. Let g be a random generator of $\mathcal{QR}$. Let $\alpha \stackrel{\$}{\leftarrow} \mathbb{Z}_p$ be chosen at random. Given as input a tuple $(p, q, T = (g, g^{\alpha}, g^{\alpha^2}, \ldots, g^{\alpha^{s-1}}))$, for any PPT adversary $\mathcal{A}$, the probability*

$$\mathsf{Pr}\left[w = g^{1/(\alpha+c)} \textit{ where } (c, w) = \mathcal{A}(p, q, T)\right]$$

is negligible in $\log p$.

We remark that when our scheme is alternatively instantiated over an elliptic curve group, the elliptic curve version of Strong Diffie-Hellman Assumption is required.

4. Background on Polynomial Commitment Scheme

Kate, Zaverucha and Goldberg [22] proposed a constant[2] size commitment scheme for polynomial. This scheme has a property which is desirable for our construction of $\mathcal{POR}$ scheme, that is, their commitment scheme allows the committer of a polynomial $f(x)$ to generate a constant size proof for the correctness of the polynomial evaluation $f(r)$ at *any* particular point $x = r$. Their scheme exploits a simple and elegant algebraic property of polynomials: *For any polynomial $f(x) \in \mathbb{Z}_p[x]$ and for any scalar input $r \in \mathbb{Z}_p$, the polynomial $x - r$ divides the polynomial $f(x) - f(r)$.*

[2] Here "constant size" means $\mathcal{O}(\lambda)$ bits where λ is the security parameter, in other words, constant number of group elements where each group element is λ bits long.

Let us denote with $f_{\vec{\boldsymbol{m}}}(x) \in \mathbb{Z}_p[x]$ the polynomial with coefficient vector $\vec{\boldsymbol{m}} = (m_0, \ldots, m_{s-1}) \in (\mathbb{Z}_p)^s$, that is, $f_{\vec{\boldsymbol{m}}}(x) \equiv \sum_{j=0}^{s-1} m_j x^j$. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative group of prime order p and $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map.

The commitment scheme [22] can be summarized as below: In the setup, a trust party chooses a public key $pk := (g, g^{\alpha}, \ldots, g^{\alpha^{s-1}}) \in \mathbb{G}^s$, where g is a generator of group $\mathbb{G}$ and $\alpha \in \mathbb{Z}_p$ is chosen at random and kept secret. In order to commit a polynomial $f_{\vec{\boldsymbol{m}}}(x) := \sum_{j=0}^{s-1} m_j x^j$ with coefficient vector $\vec{\boldsymbol{m}} = (m_0, \ldots, m_{s-1}) \in (\mathbb{Z}_p)^s$, a committer can compute a commitment $\mathcal{C}$ using the public key pk, that is, $\mathcal{C} := \prod_{j=0}^{s-1} \left(g^{\alpha^j}\right)^{m_j} = g^{f_{\vec{\boldsymbol{m}}}(\alpha)} \in \mathbb{G}$, and then publish $\mathcal{C}$. Later, for any scalar $r \in \mathbb{Z}_p$, the committer can compute $y := f_{\vec{\boldsymbol{m}}}(r) \in \mathbb{Z}_p$ and generate a *short* proof ψ (or *witness* as in [22]) to convince a verifier that y is indeed the correct evaluation of $f_{\vec{\boldsymbol{m}}}(r)$, without revealing the polynomial $f_{\vec{\boldsymbol{m}}}(x)$. The proof (or witness) ψ is generated in the following steps:

1. Divide the polynomial $f_{\vec{\boldsymbol{m}}}(x) - f_{\vec{\boldsymbol{m}}}(r)$ with $(x - r)$ using polynomial long division, and denote the coefficient vector of the resulting quotient polynomial as $\vec{\boldsymbol{w}} = (w_0, w_1, \ldots, w_{s-2})$, that is, $f_{\vec{\boldsymbol{w}}}(x) \equiv \frac{f_{\vec{\boldsymbol{m}}}(x) - f_{\vec{\boldsymbol{m}}}(r)}{x - r}$.
2. Then compute $\psi := g^{f_{\vec{\boldsymbol{w}}}(\alpha)}$ using the public key pk in the same way as computing $g^{f_{\vec{\boldsymbol{m}}}(\alpha)}$, i.e. $\psi := \prod_{j=0}^{s-2} \left(g^{\alpha^j}\right)^{w_j} = g^{f_{\vec{\boldsymbol{w}}}(\alpha)} \in \mathbb{G}$.

After receiving (r, y, ψ) from the committer, a verifier can verify whether $y \stackrel{?}{=} f_{\vec{\boldsymbol{m}}}(r)$ with the proof ψ and public key $pk = (g, g^{\alpha}, \ldots, g^{\alpha^{s-1}})$ and the public commitment $\mathcal{C}$ of the unknown polynomial $f_{\vec{\boldsymbol{m}}}(x)$, using a bilinear map [3] $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$:

$$e(g, \mathcal{C}) / e(g, g)^y \stackrel{?}{=} e(\psi, g^{\alpha} / g^r). \tag{2}$$

Note that the left hand side of above equation (2) is $e(g, \mathcal{C})/e(g, g)^y = e(g, g)^{f_{\vec{\boldsymbol{m}}}(\alpha) - y}$, and the right hand side is $e(\psi, g^{\alpha}/g^r) = e(g^{f_{\vec{\boldsymbol{w}}}(\alpha)}, g^{\alpha - r}) = e(g, g)^{(\alpha - r) f_{\vec{\boldsymbol{w}}}(\alpha)}$.

5. EPOR: Efficient Proofs of Retrievability

In this section, we construct an efficient $\mathcal{POR}$ scheme with private verifiability and call it EPOR. Our construction integrates the idea of Kate *et al.* [22] and Shacham and Waters' $\mathcal{POR}$ scheme [26] (the private verifiability version) in a seamless way. In the following description, EPOR is constructed over a modulo group. In addition, EPOR can also be alternatively instantiated over an elliptic curve group (bilinear map is not required since our scheme does not support public verifiability).

Recall that: (1) the notation $f_{\vec{\boldsymbol{m}}}(x)$ denotes the polynomial with coefficient vector $\vec{\boldsymbol{m}} = (m_0, \ldots, m_{s-1})$, that is,

[3] We remark that in the scheme proposed in this paper, bilinear map is not required, since our proposed scheme does not support public verifiability.

$f_{\vec{\boldsymbol{m}}}(x) \equiv \sum_{j=0}^{s-1} m_j x^j$; (2) our scheme described below exploits an algebraic property of polynomials: for any polynomial $f(x)$ and for any scalar input r, the polynomial $x - r$ divides the polynomial $f(x) - f(r)$.

5.1 Construction

$\underline{\mathsf{KeyGen}(1^\lambda) \rightarrow (pk, sk)}$

Choose at random a $(\lambda + 1)$ bits safe prime q such that $p = (q-1)/2$ is also prime. Let $\mathcal{QR}$ be the cyclic subgroup of quadratic residue modulo q in $\mathbb{Z}_q^*$. Choose at random a generator g of group $\mathcal{QR}$. Choose at random two elements τ, α from $\mathbb{Z}_p^*$: $\tau, \alpha \xleftarrow{\$} \mathbb{Z}_p^*$. Choose at random a PRF key, denoted as seed, from the key space of a pseudorandom function family $\{\mathsf{PRF}_{\mathsf{seed}} : \{0,1\}^{2\lambda} \rightarrow \mathbb{Z}_p\}$. The public key is $pk := (p, \{g^{\alpha^j} \mod q\}_{j=0}^{s-1})$ and the private key is $sk := (p, \mathsf{seed}, \alpha, \tau)$.

Note: (1) Both the size of the group $\mathcal{QR}$ and the multiplicative order of g modulo q are equal to p. (2) Zero is not in $\mathbb{Z}_q^*$ or $\mathcal{QR}$.

$\underline{\mathsf{DEncode}(sk, \mathbf{M}) \rightarrow (\mathtt{id}, \hat{\mathbf{M}})}$

Let $\rho \in (0,1)$ be a system parameter. Apply a rate-ρ error erasure code (e.g. Reed-Solomon code [25]) on data file $\mathbf{M}$ to generate file blocks $(\vec{\boldsymbol{m}}_0, \ldots, \vec{\boldsymbol{m}}_{n-1})$, such that each block $\vec{\boldsymbol{m}}_i = (m_{i,0}, \ldots, m_{i,s-1}) \in \mathbb{Z}_p^s$ is a vector of group elements $m_{i,j} \in \mathbb{Z}_p$, and any ρn number of blocks $\vec{\boldsymbol{m}}_i$'s can recover the original file $\mathbf{M}$ using the error erasure decoding algorithm. Choose a unique identifier $\mathtt{id}$ from domain $\{0,1\}^\lambda$. Parse the private key ssk as $(p, \mathsf{seed}, \alpha, \tau)$. For each $\vec{\boldsymbol{m}}_i$, $0 \le i \le n-1$, compute an authentication tag t_i as below

$$\begin{aligned} t_i &:= \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i) \ + \ \tau f_{\vec{\boldsymbol{m}}_i}(\alpha) \\ &= \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i) \ + \ \tau \sum_{j=0}^{s-1} m_{i,j} \alpha^j \mod p, \end{aligned} \quad (3)$$

where $\mathtt{id} \| i$ denotes an unambiguous string combination of the λ bits string $\mathtt{id}$ and the λ bits string representation of index $i \in [0, n-1]$. The final encoded file $\hat{\mathbf{M}}$ is $\{(i, \vec{\boldsymbol{m}}_i, t_i) : 0 \le i \le n-1\}$.

$\underline{\mathsf{Prove}(pk, \mathtt{id}, \hat{\mathbf{M}}, \mathtt{Chall}) \rightarrow (y, \psi, \sigma)}$

Parse $\mathtt{Chall}$ as $(\{(i, \nu_i) : i \in C\}, \ r)$, where $C \subset [0, n-1]$, $\nu_i \in \mathbb{Z}_p^*$ for each $i \in C$, and $r \in \mathbb{Z}_p^*$. Find the encoded file $\{(i, \vec{\boldsymbol{m}}_i, t_i) : i \in [0, n-1]\}$ associated to the identifier $\mathtt{id}$ and find all data blocks $\vec{\boldsymbol{m}}_i = (m_{i,0}, \ldots, m_{i,s-1})$ and tags t_i with index $i \in C$. Compute

$$\mu_j := \sum_{i \in C} \nu_i m_{i,j} \mod p, \ \text{for } 0 \le j \le s-1, \quad (4)$$

$$\sigma := \sum_{i \in C} \nu_i t_i \mod p. \quad (5)$$

Let vector $\vec{\boldsymbol{\mu}} := (\mu_0, \ldots, \mu_{s-1})$,i.e., $\vec{\boldsymbol{\mu}} = \sum_{i \in C} \nu_i \vec{\boldsymbol{m}}_i$. Evaluate polynomial $f_{\vec{\boldsymbol{\mu}}}(x)$ at point $x = r$ to obtain $y := f_{\vec{\boldsymbol{\mu}}}(r) \mod p$. Divide the polynomial $f_{\vec{\boldsymbol{\mu}}}(x) - f_{\vec{\boldsymbol{\mu}}}(r)$ with $(x - r)$ using polynomial long division, and denote the coefficients vector of the resulting quotient polynomial as $\vec{\boldsymbol{w}} = (w_0, \ldots, w_{s-2})$, that is, $f_{\vec{\boldsymbol{w}}}(x) \equiv \frac{f_{\vec{\boldsymbol{\mu}}}(x) - f_{\vec{\boldsymbol{\mu}}}(r)}{x - r}$. Compute ψ with the public key $pk = (p, \{g^{\alpha^j} \mod q\}_{j=0}^{s-1})$ as below

$$\psi := \prod_{j=0}^{s-2} \left(g^{\alpha^j}\right)^{w_j} = g^{f_{\vec{\boldsymbol{w}}}(\alpha)} \mod q \ \in \mathcal{QR}. \quad (6)$$

Output (y, ψ, σ).

$\underline{\mathsf{Verify}(sk, \mathtt{id}, \mathtt{Chall}, (y, \psi, \sigma)) \rightarrow \mathsf{accept} \ \textbf{or} \ \mathsf{reject}}$

Parse $\mathtt{Chall}$ as $(\{(i, \nu_i) : i \in C\}, \ r)$, where $C \subset [0, n-1]$, $\nu_i \in \mathbb{Z}_p^*$ for each $i \in C$, and $r \in \mathbb{Z}_p^*$. Verify the following equality Eq (7) with the private key $sk = (p, \mathsf{seed}, \alpha, \tau)$. If Eq (7) holds and $\psi \in \mathcal{QR}$, then output accept; otherwise, output reject.

$$\psi^{\alpha - r} \ \stackrel{?}{=} \ g^{\tau^{-1}\left(\sigma - \sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i)\right) - y} \mod q \quad (7)$$

Remark 1.

1. The challenge $\mathtt{Chall}$ consists of three parts: a subset $C \subset [0, n-1]$ of size ℓ, ℓ weights $\{\nu_i \in \mathbb{Z}_p : i \in C\}$, and a group element $r \in \mathbb{Z}_p$. The challenge can be represented compactly by two short PRF seeds and r like Ateniese *et al.* [1], and consequently the resulting scheme can only be proved secure in random oracle model [13, 26]. Alternatively, as mentioned in our overview in Section 2.1.3, we will apply Dodis *et al.* [13]'s result to represent a challenge $\mathtt{Chall}$ compactly as below, so that the proposed scheme can be proved without random oracle:
   (a) Choose the subset C using Goldreich [19]'s (δ, γ)-hitter[4], where the subset C can be represented compactly with only $\log n + 3\log(1/\gamma)$ bits. Assume $n < 2^{40}$ (i.e. file size less than 1024 Terabits) and let $\gamma = 2^{-80}$. Then C can be represented with 280 bits.
   (b) The sequence $(\ldots, \nu_i, \ldots)$ of ℓ weights ν_i, $i \in C$, ordered by increasing i, forms a simple geometric sequence $(\nu^0, \nu^1, \ldots, \nu^{\ell-1})$ for some $\nu \in \mathbb{Z}_p^*$.
2. Compared to Shacham and Waters [26]' scheme, the algorithm Prove in EPOR is able to aggregate the s number of weighted sums $\mu_0, \mu_1, \ldots, \mu_{s-1}$ into two short numbers y and ϕ using the idea in [22], where $y = f_{\vec{\boldsymbol{\mu}}}(r) = \sum_{j=0}^{s-1} \mu_j r^j \in \mathbb{Z}_p$ (r is a random nonce chosen by the data owner) and $\psi = g^{\frac{f_{\vec{\boldsymbol{\mu}}}(\alpha) - f_{\vec{\boldsymbol{\mu}}}(r)}{\alpha - r}} \in \mathbb{Z}_q^*$. In this way, EPOR

[4] Goldreich [19]'s (δ, γ)-hitter guarantees that, for any subset $W \subset [0, n-1]$ with size $|W| \ge (1-\delta)n$, $\mathsf{Pr}[C \cap W \neq \emptyset] \ge 1 - \gamma$. Readers may refer to [13] for more details.

requires only $\mathcal{O}(\lambda)$ communication bits per verification. In comparison, the Shacham and Waters [26] scheme requires $\mathcal{O}(s\lambda)$ communication bits per verification, since $(\mu_0, \mu_1, \ldots, \mu_{s-1})$ are sent back directly as the response.

5.2 Security

THEOREM 1. *The proposed scheme* EPOR *is a complete and sound $\mathcal{POR}$ scheme as defined in Section 3, if the SDH Assumption in Definition 3 holds and* PRF *is cryptographic secure pseudorandom function.*

The proof is given in the Appendix.

6. Performance Analysis

In this section, we analyze the performance of our proposed scheme EPOR in communication, storage, computation and false acceptance rate. We also compare our scheme with existing works by Shacham and Waters [26] and Ateniese *et al.* [1], and measure the computation time of the proposed scheme based on our prototype implementation. We remark that, although our implementation adopts a modulo group of size 2^{1024}, our scheme can alternatively use an elliptic curve group of size 2^{160}.

6.1 Communication

During a verification, the communication cost is the size of a challenge plus the size of its corresponding response (or proof). As discussed in **Remark 1** (in Section 5.1), in our scheme EPOR, a challenge consists of a subset C, and two group elements $\nu, r \in \mathbb{Z}_p$. The subset C can be compactly represented with 280 bits due to results of [19]. The group element r is used to retrieve a polynomial function value $f(r)$ for some polynomial $f(x)$ determined by a linear combination of the data blocks specified in the set C. In the security analysis, the goal of r is to retrieve multiple function values $f(r_i)$'s for different inputs r_i, and then recover the polynomial $f(x)$ by solving a linear equation system. For this reason, we can simply choose r from a smaller range $[1, 2^{80}]$ without any sacrificing in the security. Similarly, we may choose $\nu \in [1, 2^{80}]$. As a result, the challenge size is $280 + 80 \times 2 = 440$ bits.

In our scheme EPOR, a response, i.e. the proof, consists of three group elements y, σ, ψ, which are derived from the challenge, the data blocks and authentication tags. So the size of a response is 3λ bits. Therefore, the communication cost per verification is $3\lambda + 440$ bits.

6.2 Storage

During verification, the data owner only keeps the private key in her local storage. The size of private key is $3\lambda + 80$ bits.

The storage overhead (due to authentication tags) on the cloud storage server side is $1/s$ of the data file size, where the system parameter s is the block size and equals to the ratio of the size of a data block to the size of an authentication tag. The public key is also kept in the cloud storage and its size is $(s + 1)\lambda$ bits. Note that in our scheme, there is only one public key per user, without regarding to the number of files the user stores in the cloud storage server.

6.3 Computation

The proposed scheme EPOR is very efficient in setup. Key generation requires s number of group exponentiations. Suppose an $ns\lambda$ bits encoded data file consists of n data blocks, each block has s group elements and each group element has λ bits. The data preprocess (i.e. the DEncode algorithm) requires only ns number of group multiplications and additions, together with n PRF evaluations. Note that the PRF is simulated with an AES [11] stream cipher, which runs in the counter model [31].

During a verification, the computation complexity on the cloud storage server side is dominated by the computation of ψ in Equation (6) in the algorithm Prove on page 6. This dominant step takes $(s - 1)$ number of group exponentiations, and is the bottleneck of efficiency of our scheme when the block size s becomes large.

6.4 False Positive Rate

Recall that, error erasure code is applied at the beginning of the algorithm DEncode. Suppose a rate-ρ Reed-Solomon code is adopted, that is, any ρ fraction of data blocks in the encoded file can recover the original file, and the ratio of the size of encoded file to the original is $1/\rho$. If an encoded file is corrupted such that it is unable to recover the original using the erasure decoding, then more than $1 - \rho$ fraction of data blocks are corrupted. In this case, a randomly chosen data block is not corrupted with probability smaller than ρ, and the probability[5] that ℓ independently randomly chosen data blocks will not hit any corrupted data block is smaller than ρ^ℓ, independent on the file size. Our scheme guarantees that if a corrupted data block is hit in a verification, then the data owner will accept with only negligible probability. So the false acceptance rate is smaller than ρ^ℓ, if ℓ independently random blocks are accessed in a verification.

We list out the false acceptance rate w.r.t. various challenge size and various erasure code rate in Table 2. The choices of value of challenge size ℓ is 100, 300, 500, or 700; the choices of erasure encode rate ρ is 0.99 or 0.98. Note that the the storage overhead due to erasure encoding is $1/0.99 - 1 \approx 0.0101$ of original file size, if $\rho = 0.99$; $1/0.98 - 1 \approx 0.0204$, if $\rho = 0.98$.

6.5 Recommended System Parameters

We recommend the following system parameter for our proposed scheme EPOR: The error erasure rate is 0.98, block size s is around 160, the challenge size is around 500. In this setting, the false acceptance rate is 4.1024×10^{-5}, the

[5] Note that this argument is based on the case of choosing indices of data blocks at random *with* replacement. The other case where choosing indices at random *without* replacement will have a larger error detection rate.

Challenge Size	False Accept Rate ρ^ℓ with $\rho = 0.99$	False Accept Rate ρ^ℓ with $\rho = 0.98$
$\ell = 100$	0.366032341	0.132619556
$\ell = 300$	0.049040894	0.002332506
$\ell = 500$	0.006570483	0.000041024
$\ell = 700$	0.000880311	0.000000722

Table 2. The False Accept Rate Versus Challenge Size and Erasure code rate. Recall that the challenge size ℓ represents the number of data blocks accessed in a verification.

number of communication bits required in a verification is 920 for elliptic curve group or 3512 for modulo group, the storage overhead is about 2% due to erasure encoding and $1/160 = 0.625\%$ due to authentication tags. Our experiment will confirm that the query latency is within 1 second.

6.6 Comparison

We give a comparison on the performances of our scheme with SW [26] and Ateniese *et al.* [1] in Table 3 with an example. A more detailed and generic comparison is given in Table 4 on page 9. Note that in our proposed scheme EPOR, the practical choice of value s is bounded by the computation on the server side, which is similar to the case of Ateniese [1]. In contrast, in SW [26], the largest practical value of s is limited by the communication requirement.

We also compare the proposed scheme EPOR with Shacham and Waters' scheme [26] in communication and storage overhead. For a 1GB data file, we plot the number of communication bits (i.e. the size of a challenge and a proof) against the storage overhead for both schemes in Figure 1.

6.7 Experiment to measure computation time

The goal of this experiment is to measure the actual running time of the four algorithms KeyGen, DEnc, Prove, Verify in the proposed EPOR scheme, with disk IO time included and networking communication time excluded. Note that the reported query latency includes of running time of Prove and Verify and disk IO time.

6.7.1 Experiment Environment and Setting

We have implemented a prototype of our EPOR scheme in C programming language. The large integer arithmetic is computed using GNU MP [17] library version 5.0.1. The pseudorandom function PRF are simulated with AES symmetric cipher provided in OpenSSL [24] library version 1.0.0d. The disk IO is handled by C library function `mmap`. We observe a low memory consumption for all experiments conducted. Our implementation is not optimized (e.g. it is a single process/thread program) and further performance improvements of our scheme can be expected.

The test machine is a laptop computer, which is equipped with a 2.5GHz Intel Core 2 Duo CPU (model T9300), a 3GB PC2700-800MHZ RAM and a 7200RPM hard disk.

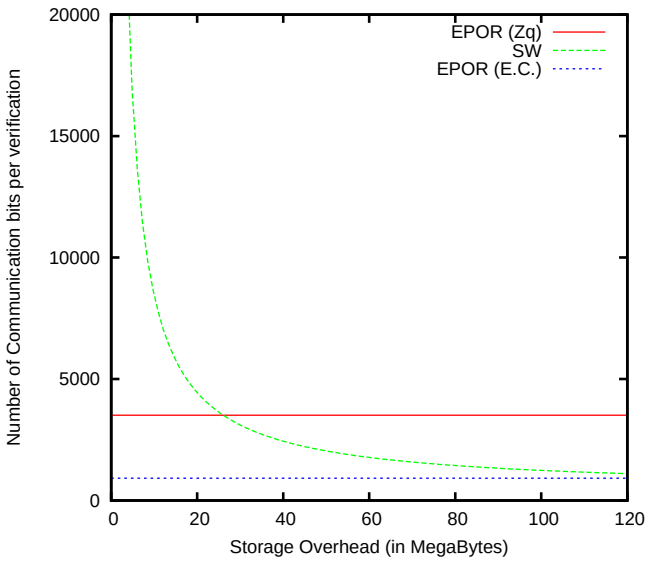


Figure 1. Comparison on communication (in bits) and storage overhead (in megabytes) w.r.t. a 1GB data file. **SW** denotes the $\mathcal{POR}$ scheme with private verification by Shacham and Waters [26]; EPOR (E.C.) denotes our proposed scheme instantiated over elliptic curve group; EPOR ($\mathbb{Z}_q^*$) denotes our proposed scheme instantiated over group $\mathbb{Z}_q^*$. Note: In this comparison, the size of public key is counted as a part of storage overhead for EPOR.

	Semantics	Choices of Values
λ	Bit-length of group size	1024
s	Block size, i.e. the number of group elements in a data block	40, 80, 160, 320, 640, or 960
ℓ	Challenge size, i.e. the number of data blocks accessed in a verification	100, 300, 500 or 700.

Table 5. The choices of values of various system parameters in our experiment

The test machine runs 32 bits version of Gentoo Linux OS with kernel 2.6.36. The file system is EXT4 with 4KB page size.

Our test data files are of size 16MB, 32MB, 64MB, 128MB, 256MB and 512MB, respectively (These are the file sizes after error erasure encoding). The choices of values of various system parameters, i.e. group size 2^λ, block size s and challenge size ℓ, are listed in Table 5.

Our experiments are conducted in this way:

- Key generation: For each choice of block size s, we generate a key pair with size s using the key generating program KeyGen. The generated public key consists of s group elements.
- Data preprocess: For each test file, for each choice of value of block size s, we run the data encoding program DEncode to generate a set of authentication tags.
- Verification: For each test file, for each choice of value of block size s, for each choice of value of challenge

Scheme	Group element size (bits)	Communication bits	Computation (Data Preprocess)	Computation (Prove)
[3][1]	$\lambda = 1024$	$2\lambda + 520 = 2568$	2^{23} exp over $\mathbb{Z}_N^*$	$(100 + \ell)$ exp over $\mathbb{Z}_N^*$
[26]	$\lambda = 80$	$(s+1)\lambda + 360 = 8440$	2^{27} mul over $\mathbb{Z}_p$	100ℓ mul over $\mathbb{Z}_p$
EPOR (E.C.)	$\lambda = 160$	$3\lambda + 440 = 920$	2^{26} mul over $\mathbb{Z}_p$	100 exp over Elliptic Curve
EPOR ($\mathbb{Z}_q^*$)	$\lambda = 1024$	$3\lambda + 440 = 3512$	2^{23} mul over $\mathbb{Z}_p$	100 exp over $\mathbb{Z}_q^*$

Table 3. Comparison with an example among the $\mathcal{PDP}$ scheme by Ateniese *et al.* [1], the $\mathcal{POR}$ scheme by Shacham and Waters [26], and the $\mathcal{POR}$ scheme named EPOR proposed in this paper. After erasure encoding, the file size is 1GB, block size is $s = 100$, and storage overhead due to authentication tags is about 10MB for all schemes. For all schemes listed below, we assume that, during a verification, the (part of) challenge $\{(i, \nu_i) : i \in C\}$ are represented compactly with 280 bits due to results of [13, 19]. System parameter ℓ represents the size of set C. All computation times are represented by the corresponding dominant factor. "exp" and "mul" denote the group exponentiation and group multiplication respectively in the corresponding group. *Note: (1) In Ateniese* et al. *$\mathcal{PDP}$ scheme, exponentiation with a large integer exponent of size $s\lambda$ is required. We represent such exponentiation as a number of s normal group exponentiation* exp*, where the exponent is λ bits long. (2) One 1024 bits modular exponentiation or one 160 bits elliptic curve exponentiation takes roughly 5 millisecond in a standard PC.*

Scheme	Group element size (bits)	Communication (bits)	Storage Overhead	Computation (Prover)	Computation (Verifier)	Computation (Data Preprocess)
[12]	$\lambda = 1024$	2λ	Zero	$\|\mathsf{F}\|/\lambda$ exp	1 exp	1 exp
[22]	$\lambda = 160$	3λ	Zero	$\|\mathsf{F}\|/\lambda$ exp + $2\|\mathsf{F}\|/\lambda$ (mul + add)	2 pairing	$\|\mathsf{F}\|/\lambda$ (mul + add) + 1 exp
[4]	$\lambda = 160$	$2\lambda + 440$	$\|\mathsf{F}\|$	$\|\mathsf{F}\|/\lambda$ (exp + mul + add)	2 exp	$\|\mathsf{F}\|/\lambda$ (exp + mul + add)
[3][1]	$\lambda = 1024$	$2\lambda + 520$	$\|\mathsf{F}\|/s$	$(\ell + s)$ exp + 2ℓ mult. + ℓ add + 1 hash + 1 samp	ℓ (exp + mult.) + 1 hash + 1 samp	$\|\mathsf{F}\|/\lambda$ exp + n hash
[26]	$\lambda = 80$	$(s+1)\lambda + 360$	$\|\mathsf{F}\|/s$	$s\ell$ (add + mult) + 1 samp	$(\ell + s)$ (add + mult) + ℓ PRF + 1 samp	$\|\mathsf{F}\|/\lambda$ (mul + add) + $\|\mathsf{F}\|/(\lambda s)$ PRF
EPOR (E.C.)	$\lambda = 160$	$3\lambda + 440$	$\|\mathsf{F}\|/s$	$(s-1)$ exp + $(s\ell + s + \ell)$ (add + mul) + 1 samp	2 exp + ℓ (add + mult) + ℓ PRF + 1 samp	$\|\mathsf{F}\|/\lambda$ (mul + add) + $\|\mathsf{F}\|/(\lambda s)$ PRF
EPOR ($\mathbb{Z}_q^*$)	$\lambda = 1024$	$3\lambda + 440$	$\|\mathsf{F}\|/s$	$(s-1)$ exp + $(s\ell + s + \ell)$ (add + mul) + 1 samp	2 exp + ℓ (add + mult) + ℓ PRF + 1 samp	$\|\mathsf{F}\|/\lambda$ (mul + add) + $\|\mathsf{F}\|/(\lambda s)$ PRF

Table 4. Performance Comparison. All schemes support private verification only. In each scheme (except the first two in the table), a challenge set $C \subset [0, n-1]$ contains ℓ block indices and can be compactly represented with 280 bits due to results of [13, 19]. In the table, "exp", "mul" and "add" represent exponentiation, multiplication and addition in the corresponding groups/fields; "samp" represents the sample method given by Goldreich [19]; notation $|\mathsf{F}|$ denotes the file size in bits. Recall that the notations λ, s, ℓ, n are as described in Table 1 in Section 3. *Note: In Ateniese* et al. *$\mathcal{PDP}$ scheme, exponentiation with a large integer exponent of size $s\lambda$ is required. We represent such exponentiation as a number of s normal group exponentiation* exp*, where the exponent is λ bits long. Similar for the RSA based scheme.*

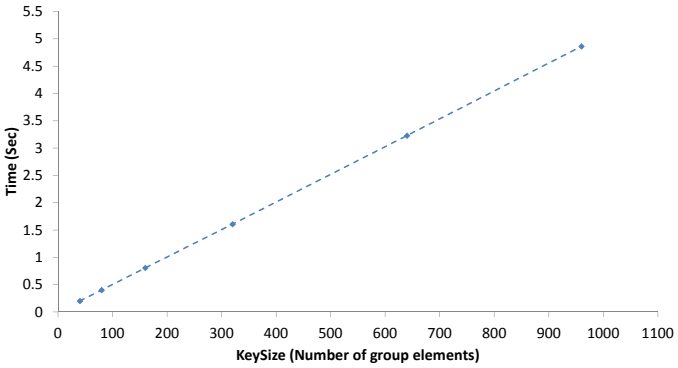
size ℓ, we run the Prove and Verify programs to simulate the interaction between the data owner and cloud storage server.

Every single experiment case is repeated for 10 times and the reported timing data are the averages. We remark that experiment trials are run in sequence without parallelism.

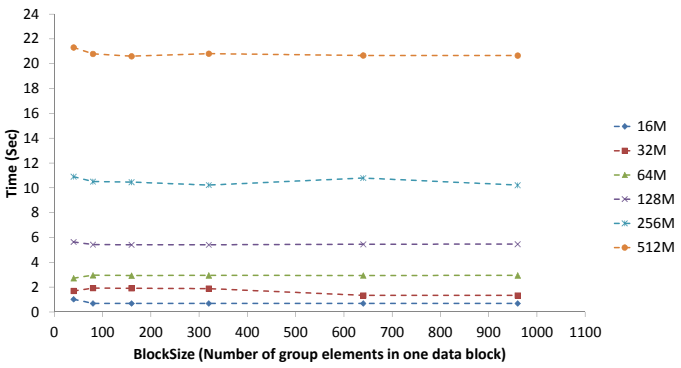
6.7.2 Experiment Results

The experiment results are showed in Figure 2. All experiment results are averaged over 10 trials. Since all experiment results vary little across different trials, we do not report the variances or confidence intervals.

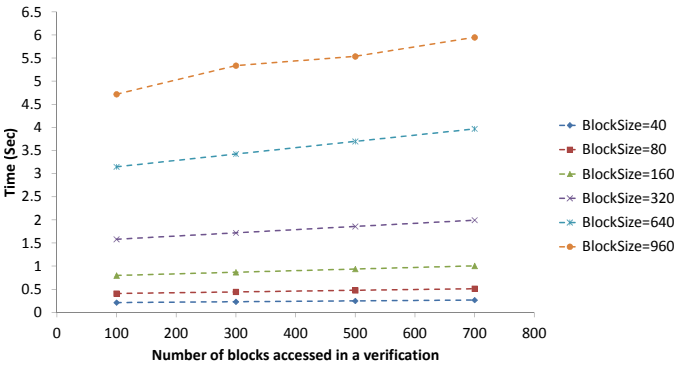
Our experiment result in Figure 2(a) indicates that the key generating time is proportional to the key size, i.e. the number of group elements in a key. The experiment result in Figure 2(b) indicates that the data preprocess time (particularly, DEncode) is proportional to the data file size and almost independent on the block size s. The experiment also shows that the query latency is proportional to the block size s, almost independent on the file size, and grows very slowly with the challenge size ℓ, suggesting that the computation of exponentiations becomes the bottleneck when s is so large. All of these results agree with our analysis.



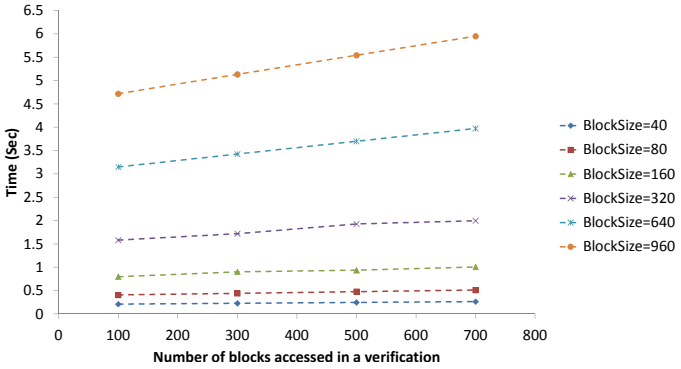
(a) Time to generate a key VS the key size



(b) Data preprocess time VS the block size. Each line is labeled with the size (in megabytes) of corresponding data file.



(c) Query latency VS the challenge size for a 128MB data file. Each line is labeled with the corresponding block size.



(d) Query latency for a 512MB data file.

Figure 2. The subfigure (c) and (d) represent the results of the same experiment w.r.t. different data files, where (c) for a 128MB data file and (d) for a 512MB data file. The key size is the number of group elements in a key; the block size is the number of group elements in one data block; the challenge size is the number of data blocks accessed during one verification. All time measurements include disk IO time, but do not include network communication time.

7. Conclusion

We proposed an efficient and secure $\mathcal{POR}$ scheme which supports private verifiability. The proposed scheme requires only linear communication bits w.r.t. the security parameter (particularly 920 bits when elliptic curve group is used) per verification and $1/s$ storage overhead, where s can be as large as hundreds. The small number of communication bits in a verification makes it possible to piggyback the challenge and/or response of our scheme into other communication packets between the data owner and the cloud storage server if any.

How to apply the idea of Kate, Zaverucha and Goldberg [22] to reduce the response size of the other $\mathcal{POR}$ scheme with public verifiability in Shacham and Waters [26], remains an open problem.

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *CCS '07: ACM conference on Computer and communications security*, pages 598–609, 2007.

[2] G. Ateniese, S. Kamara, and J. Katz. Proofs of Storage from Homomorphic Identification Protocols. In *ASIACRYPT '09: International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, pages 319–333, 2009.

[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 Transactions on Information and System Security*, 14:12:1–12:34, 2011.

[4] S. Benabbas, R. Gennaro, and Y. Vahlis. Verifiable Delegation of Computation over Large Datasets. In *CRYPTO*, page 110, 2011.

[5] D. Boneh and X. Boyen. Short Signatures Without Random Oracles. In *Advances in Cryptology—EUROCRYPT 2004*, pages 56–73, 2004.

[6] D. Boneh and X. Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *J. Cryptol.*, 21:149–177, 2008. ISSN 0933-2790.

[7] K. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: theory and implementation. In *CCSW '09: ACM workshop on Cloud computing security*, pages 43–54, 2009.

[8] K. D. Bowers, A. Juels, and A. Oprea. HAIL: a high-availability and integrity layer for cloud storage. In *CCS '09: ACM conference on Computer and communications security*, pages 187–198, 2009.

[9] E.-C. Chang and J. Xu. Remote Integrity Check with Dishonest Storage Server. In *ESORICS '08: European Symposium on Research in Computer Security: Computer Security*, pages 223–237, 2008.

[10] R. Curtmola, O. Khan, R. Burns, and G. Ateniese. MR-PDP: Multiple-Replica Provable Data Possession. In *ICDCS '08: International Conference on Distributed Computing Systems*, pages 411–420, 2008.

[11] J. Daemen and V. Rijmen. *The Design of Rijndael: AES - The Advanced Encryption Standard*. 2002.

[12] Y. Deswarte, J.-J. Quisquater, and A. Saïdane. Remote Integrity Checking: How to Trust Files Stored on Untrusted Servers . In *Proceeding of the Conference on Integrity and Internal Control in Information Systems*, pages 1–11, 2003.

[13] Y. Dodis, S. Vadhan, and D. Wichs. Proofs of Retrievability via Hardness Amplification. In *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, TCC '09, pages 109–127, 2009.

[14] Dropbox. Dropbox Forums on Data Loss Topic. `http://forums.dropbox.com/tags.php?tag=data-loss`.

[15] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. In *CCS '09: ACM conference on Computer and communications security*, pages 213–222, 2009.

[16] Amazon Forum. Major Outage for Amazon S3 and EC2. `https://forums.aws.amazon.com/thread.jspa?threadID=19714&start=15&tstart=0`.

[17] GMP. The GNU Multiple Precision Arithmetic Library. `http://www.gmplib.org/`.

[18] O. Goldreich. *Foundations of Cryptography: Volume 1, Basic Tools*. Cambridge University Press, New York, NY, USA, 2006. ISBN 0521035368.

[19] O. Goldreich. A Sample of Samplers - A Computational Perspective on Sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[20] Z. Hao, S. Zhong, and N. Yu. A Privacy-Preserving Remote Data Integrity Checking Protocol with Data Dynamics and Public Verifiability. *IEEE Transactions on Knowledge and Data Engineering (Concise Paper)*, 23(9):1432 – 1437, 2011.

[21] A. Juels and B. Kaliski. Pors: proofs of retrievability for large files. In *CCS '07: ACM conference on Computer and communications security*, pages 584–597, 2007. ISBN 978-1-59593-703-2.

[22] A. Kate, G. Zaverucha, and I. Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In *ASIACRYPT*, pages 177–194, 2010.

[23] M. Naor and G. Rothblum. The complexity of online memory checking. *J. ACM*, 56:2:1–2:46, 2009.

[24] OpenSSL. OpenSSL Project. `http://www.openssl.org/`.

[25] I. Reed and G. Solomon. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics (SIAM)*, 8(2):300–304, 1960.

[26] H. Shacham and B. Waters. Compact Proofs of Retrievability. In *ASIACRYPT*, pages 90–107, 2008.

[27] Business Insider. Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data. `http://www.businessinsider.com/amazon-lost-data-2011-4`.

[28] C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-Preserving Public Auditing for Data Storage Security in Cloud Computing. In *IEEE INFOCOM '10*, pages 525–533, 2010.

[29] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *ESORICS'09: European conference on Research in computer security*, pages 355–370, 2009.

[30] K. Yang and X. Jia. Data storage auditing service in cloud computing: challenges, methods and opportunities. *(will appear in) Journal of World Wide Web*, 2011.

[31] A. Young and M. Yung *Malicious Cryptography: Exposing Cryptovirology* ISBN 0764549758

A. Security Proof

Here we provide the proof of Theorem 1.

A.1 The underlying authenticator is unforgeable

LEMMA 2. *Suppose the pseudorandom function* PRF *is secure and Strong Diffie-Hellman Assumption holds. In the proposed scheme* EPOR*, the prover's response* (y, ψ, σ) *is unforgeable.*

The pseudorandom function can be replaced with a true random generator function, with negligible difference, due the the standard hybrid argument [18]. In the below proof, we just treat the output of PRF as true randomness.

Proof : Suppose a PPT adversary $\mathcal{A}$ can forge a response (y', ψ', σ') such that $\mathsf{Verify}(sk, \mathtt{id}, \mathtt{Chall}, (y', \psi', \sigma')) = \mathtt{accept}$. Since both the forgery response $(y', \psi', \sigma',)$ and the corresponding genuine output (y, ψ, σ) which is generated by an honest prover are accepted by the verifier algorithm Verify w.r.t. the challenge $\mathtt{Chall} = (\{(i, \nu_i) : i \in C\},\ r)$, the two tuples satisfy the Equation (7) (on page 6):

$$\psi^{\alpha - r} = g^{\tau^{-1}\left(\sigma - \sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i)\right) - y} \mod q \quad (8)$$

$$\psi'^{\alpha - r} = g^{\tau^{-1}\left(\sigma' - \sum_{i \in C} \nu_i \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i)\right) - y'} \mod q \quad (9)$$

Dividing Equation (8) with Equation (9), we obtain

$$\left(\frac{\psi}{\psi'}\right)^{\alpha - r} = g^{\tau^{-1}(\sigma - \sigma') + y' - y} \quad (10)$$

Now we do a case analysis on whether σ' is equal to σ.

Case 1: $\sigma' \neq \sigma \mod p$**.** If a computationally unbounded adversary can find a valid forgery response $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' \neq \sigma$ with non-negligible probability, then it can find the value of τ from Eq (10) with non-negligible probability.

Recall that in this proof, the pseudorandom function PRF is replaced by a true randomness generator. Thus, the secret value τ is hidden perfectly in the authentication tags $t_i = \mathsf{PRF}_{\mathsf{seed}}(\mathtt{id} \| i) +$

$\tau f_{\vec{m}_i}(\alpha) \mod p$. Any malicious adversary (playing the role of Bob) cannot find the value of $\tau \in \mathbb{Z}_p^*$ after interacting with Alice by running the scheme EPOR with probability larger than $\frac{1}{p-1}$, even if it is computationally unbounded. Therefore, there is no PPT adversary that can find a valid forgery response $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' \neq \sigma$ with non-negligible probability.

Case 2: $\sigma' = \sigma \mod p$**.** In case 2, we rewrite the Eq (10) as below

$$\left(\frac{\psi}{\psi'}\right)^{\alpha - r} = g^{y' - y} \mod q. \tag{11}$$

Now, we do a case analysis on whether y' is equal to y.

Case 2.1: $y' = y \mod p$**.** The equality that $y' = y$, implies $\psi' \neq \psi$, since $(y', \psi', \sigma') \neq (y, \psi, \sigma)$ and $\sigma' = \sigma$. Note that the verifier algorithm Verify accepts the forgery output (genuine output respectively) only if ψ' (ψ respectively) is a quadratic residue modulo q. In the subgroup $\mathcal{QR}$ of quadratic residue modulo q, all elements, except unity element 1, have multiplicative order p modulo q. We know that $\psi'/\psi \neq 1$, so the element $\psi'/\psi \in \mathcal{QR}_q$ has multiplicative order p. Thus, Equation (11) and $y' = y \mod p$ together imply $\alpha = r \mod p$. Thus adversary $\mathcal{A}$ can find $(c, w = g^{1/(r+c)} = g^{1/(\alpha+c)})$ which is a valid solution to the s-SDH problem[6].

Case 2.2: $y' \neq y \mod p$**.** Equation (10) and $y' \neq y \mod p$ together imply that $\alpha \neq r \mod p$. In this case, adversary $\mathcal{A}$ computes $(c = -r, w = \left(\frac{\psi}{\psi'}\right)^{1/(y'-y)} \mod q)$ as solution to the SDH problem. Next, we will show that this solution to SDH problem is valid.

Substituting $\frac{\psi}{\psi'}$ with $\left(w^{y'-y} \mod q\right)$ into the Equation (10), we have

$$w^{(y'-y)(\alpha-r)} = g^{y'-y} \mod q \tag{12}$$

Since $y' - y \neq 0 \mod p$ and $\alpha - r \neq 0 \mod p$, their inverses $1/(y' - y) \mod p$ and $1/(\alpha - r) \mod p$ exist. Therefore, the following equality can be derived from Equation (12):

$$w = \left(g^{y'-y}\right)^{\frac{1}{(y'-y)(\alpha-r)}} = g^{\frac{1}{\alpha-r}} \mod q \tag{13}$$

The above Equation (13) shows that the adversary $\mathcal{A}$'s output $(c = -r, w)$ is a valid solution to the SDH problem.

Therefore, Lemma 2 is proved. □

We remark that the above proof in Case 2.2 borrows ideas from the proof in Kate *et al.* [22].

A.2 User file can be retrieved through erasure decoding

The proof of retrievability through erasure decoding is sketched as below. Full proof will be provided in the full version of this paper.

As discussed in the overview in Section 2.1.3, the proposed scheme EPOR applies three encoding schemes subsequently upon an input data file $\mathbf{M}$, using the jargon of [13]:

1. Primary encoding: In the setup, the data owner Alice applies an error erasure encoding scheme on her data file $\mathbf{M}$, and obtain a encoded file which consists of n data blocks $\{\vec{m}_i \in \mathbb{Z}_p^s : i \in [0, n-1]\}$.
2. Secondary encoding: In a verification, Alice (i.e. verifier) chooses a challenge (C, ν, r), where $C \subset [0, n-1]$ is a subset of size ℓ and $\nu, r \in \mathbb{Z}_p$. The cloud storage server Bob (i.e. prover) finds all data blocks with index in set C. Let us rename the selected data blocks (ordered by increasing block index) as sequence $(\vec{x}_0, \ldots, \vec{x}_{\ell-1})$. Bob computes a vector $\vec{\nu} = (\nu^0, \ldots, \nu^{\ell-1})$ and a Reed-Solomon codeword of $\vec{x}_i$'s:

$$\vec{\mu} := \sum_{i=0}^{\ell-1} \nu^i \vec{x}_i \mod p.$$

   The linear homomorphism of the underlying homomorphic linear authenticator allows Bob to compute an authentication tag σ for $\vec{\mu}$.
3. Tertiary encoding: The third part r of the challenge (C, ν, r), will be used for the tertiary encoding. Instead of sending $(\vec{\mu}, \sigma)$ as response to Alice like [13, 26], in our proposed scheme, Bob will apply a Reed-Solomon encoding with parameter r on the long message $\vec{\mu}$, to obtain a short message: $y := \sum_{i=0}^{s-1} r^i \mu_i$, where $\vec{\mu} = (\mu_0, \ldots, \mu_{s-1})$. A short authentication tag σ_y for y can be computed from $(\vec{\mu}, \sigma)$ using the idea of [22]. As a result, only the short message-tag pair (y, σ_y) is sent back to Alice as response.

We may view our tertiary encoding for input data file $\mathbf{M}$ in this way: Take the selected blocks $\mathbf{X} := (\vec{x}_0, \ldots, \vec{x}_{\ell-1})$ as data file. Let the primary encoding[7] w.r.t. file $\mathbf{X}$ be:

$$\left\{ \sum_{i=0}^{\ell-1} \nu^i \vec{x}_i \mod p : \nu \in \mathbb{Z}_p \right\}.$$

In each verification, a verifier chooses a challenge (ν_0, r), where ν_0 specifies one block in the encoded file of $\mathbf{X}$. The prover should apply a Reed-Solomon code on the selected block $\vec{\mu} := \sum_{i=0}^{\ell-1} \nu_0^i \vec{x}_i \mod p$ with parameter r to obtain a short message $y := \sum_{i=0}^{s-1} r^i \mu_i$, where $\vec{\mu} = (\mu_0, \ldots, \mu_{s-1})$. In summary, the tertiary encoding w.r.t. input data file $\mathbf{M}$ can be considered as the secondary encoding w.r.t. file $\mathbf{X}$.

Consequently, the security of the proposed scheme can be proved using results of [13] (Precisely, apply Lemma 6 of [13] *twice*[8] and then apply Lemma 7 of [13].)

[6] From the input of SDH problem, one can simulate scheme EPOR. Details are saved due to space constraint.

[7] This primary encoding will be computed on the fly.

[8] In contrast, the proof of the improved version of SW scheme in [13] only applies Lemma 6 for one time.