Proofs of Retrievability with Public Verifiability and Constant Communication Cost in Cloud

ABSTRACT

For data storage outsourcing services, it is important to allow data owners to efficiently and securely verify that the storage sever stores their data correctly. To address this issue, several proof-of-retrievability(POR) schemes have been proposed wherein a storage sever must prove to a verifier that all of a client's data is stored correctly. While existing POR schemes offer decent solutions addressing various practical issues, they either have a non-trivial (linear or quadratic) communication complexity, or only support private verication - only the data owner can verify the remotely stored data. It remains open to design a POR scheme that achieves both public verifiability and constant communication cost simultaneously.

In this paper, we solve this open problem and propose the first POR scheme with public verifiability and constant communication cost. In our proposed scheme, the message exchanged between the prover and verifier is composed of a const number of the underlying group elements. Different from existing private POR construction, our scheme allows public verification and releases the data owners from burden of being staying online. Thorough analysis and experiments on Amazon S3 show that our proposed scheme is efficient and practical. We prove the security of our scheme based on Computational Diffie-Hellman Assumption, Strong Diffie-Hellman Assumption and Bilinear 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

Proofs of Retrievability, Cloud Storage, Public Verification, Integrity Check, Constant Communication, Polynomial Commitment

1. INTRODUCTION

Due to a number of unprecedented advantages: resource elasticity, on-demand self-service, location independent resource polling, etc[25], cloud storage is increasingly attracting organizations and individuals to outsource their local data to it. Currently, millions of users have been involved into cloud storage services including Amazon S3, Microsoft Skydrive, Google Cloud Storage and Dropbox. It is clear that cloud storage is a raising trend to become a pervasive service.

Despite the proliferation of cloud storage service, there are also increasing security concerns caused by the shortages of cloud servers. One significant concern is the data integrity, i.e., whether the cloud server indeed stores its clients' data correctly. Recently, a number of data loss events have been reported for those best-known storage providers[2, 1, 3, 4], including Amazon S3, Dropbox. To ensure the clients' confidence for the integrity of data previously stored on the cloud storage server, a reliable proof-of-retrievability(POR)[17] system is desirable. Specifically, in a POR system, the data storage server must prove that he actually stores the clients' data correctly. At the same time, the client can verify whether the proof generated by the server is valid. For the construction of a POR system, there are several aspects that mainly affect the performance of the system: 1) communication cost between server and client in a verification; 2) private verifiability or public verifiability; 3) storage overhead on server side and client side; 4) computation cost of server and client for each verification. Although several POR techniques[23, 11, 16] have been proposed, different limitations are existed in them, especially for the communication cost and the support of public verification.

By utilizing the idea of aggregating block integrity values, Shacham and Waters(SW)[23] proposed two fast POR schemes. In their POR constructions, one scheme supports the private POR verification and the other one achieves the public POR verification. However, both public and private SW schemes suffer from the communication cost, which is linear to the number of elements in the data block. For the integrity check of large data files, this kind of communication cost will obviously introduce high latency to the schemes and even makes them unaffordable in some bandwidth limited scenarios(e.g., mobile devices paid by limited data plan). Therefore, it is desirable in SW schemes to remove the influence of s on the bandwidth consuming. To overcome the limitation in Ref.[23], Xu et al.[16] construct a private *POR* scheme with constant communication cost by utilizing a recent proposed polynomial commitment technique. The main drawback of this scheme is that it only supports private verification. This kind of verification will cause heavy burden on both communication and computation to the data owner if he wants to share his data with other clients. Because the owner has to stay online and perform verification for every integrity check request of other clients. In Ref.[10], Dodis et al. enhance SW schemes by reducing the challenge message size, but no change is made to the response size, which is linear to the number of elements in the data block. To our best knowledge, there is no existing solution proposed for the public *POR* with constant communication cost. It is still an open problem problem as mentioned in Ref.[16].

In this paper, we solve this problem and design a secure and practical *POR* scheme with constant communication cost and public verifiability, called *PCPOR*. The main idea of our scheme can be summarized as follows: a data owner first breaks an ensure coded file into n blocks $\{m_i\}$,$1 \leq i \leq n$ and generates a corresponding authentication tag $\{\sigma_i\}$ for each block. Then, all the data blocks and tags are outsourced to the cloud storage server. When a client wants to retrieve the data from the server and check whether the data is stored correctly, he can generate a challenge message and send it to the cloud server. By receiving the challenge message, the cloud server generates the proof for the correct data storage based on the message, the public key information and the previously stored tags, and then returns it to the client as response. After receiving the proof response, the client performs the verification algorithm to check data integrity using the public key information. In our *PCPOR* scheme, any client or auditing third-party can perform verification process without contacting the data owner, who can keep off-line after outsourcing his data. By tailoring a constant size polynomial commitment technique[18], our proposed scheme aggregates the proof information into a polynomial, which makes the size of proofs independent to the number of elements in data block and achieve better performance in storage cost than the existing *POR* schemes. Through the unique incorporation of techniques form Dotis et al.[10], our *PCPOR* reduces the challenge message complexity in Ref.[23] from $O(\lambda^2)$ to $O(\lambda)$. Experiments on Amazon EC2 Cloud platform show that our *PCPOR* scheme is efficient and practical. Through the security analysis based on Computational Diffie-Hellman Assumption(CDH), Strong Diffie-Hellman(SDH) Assumption and Bilinear Strong Diffie-Hellman(BSDH) Assumption, we prove that our proposed scheme is secure.

Main contributions of this paper can be summarized as below.

- We construct the first secure and efficient *POR* scheme with constant communication cost and public verifiability.
- Our proposed scheme solves the open problem pointed out in Ref.[16] and omits the trade-off between communication cost and storage cost in the public SW scheme[23].
- We formally prove the security of our scheme. The advantages of our proposed scheme are validated via both numerical analysis and experimental results.

The rest of this paper is organized as follows: In section 2, We review and discuss the related works. Section 3 describes the system model, security model and assumptions. We introduce the technique preliminaries of our work in Section 4, which is followed by the the construction and security proof of our proposed scheme in Section 5. We evaluate the performance of our scheme in Section 6. Discussions about our paper are provided in Section 7 and we conclude the paper in Section 8

2. RELATED WORK

In Ref.[17], Juel et al. first defined the *POR* formally, which allows a storage server to convince a client that it can correctly retrieve a file previously stored at the server. In their proposed *POR* scheme, disguised blocks hidden among regular file blocks are utilized to detect data modified by the server. However, the communication cost in this scheme is inevitably large, i.e. $O(s\lambda^2)$, where λ is the security parameter and s is the number of elements in each erasure coded file block. This is because there are $O(\lambda)$ file blocks in the proof response, each of which has a size of $O(\lambda)$. What is more, the number of challenges supported by this scheme is a fixed priori and thus limits its application. With similar purpose of Ref.[17], Ateniese et al.[5] proposed an efficient but weaker provable data possession model using homomorphic authentication tag. However, an adversary was later introduced for this scheme by Shacham and Waters[23], which can answer a fraction of queries correctly with non-negligible probabilities.

To omit the limitation in Juel et al.'s *POR* scheme[17], Shacham and Waters(SW)[23] proposed two fast *POR* schemes based on the homomorphic linear authenticators[5], which enables the storage server to reduce the proof complexity by aggregating the authentication tags of individual file blocks. In their constructions, one proposed *POR* schemes supports private verification based on pseudorandom functions(PRFs)[13] and the other one achieves public verifiability by utilizing BLS signatures[7]. Compared to scheme in Ref.[17], the communication cost for proof response in Ref.[23] is reduced from $O(s\lambda^2)$ to $O(s\lambda)$ and can support unlimited number of challenges, where s is the number of elements in each erasure coded file block. At the same time, they first provide a security proof against arbitrary adversaries in the formal *POR* model. However, in SW schemes, the communication size for proof response is still linear to the value of s and the challenge cost is increased to $O(\lambda^2)$. When considering the integrity verification of large data files, the bandwidth consuming will introduce high delay, which can be worse and even unaffordable in some scenarios with restricted bandwidth, such as mobile devices paid by limited data plan.

Following SW schemes[23], several *POR* schemes are proposed recently to enhance it in terms of communication cost. In Ref.[10], by using a $(\gamma, \delta) - hitter$ introduced by Goldreich[19], Dodis et.al. reduce the size of challenge message in SW scheme from $O(\lambda^2)$ to $O(\lambda)$. Nevertheless, there is still no change made in this scheme to the response size, which is linear to the number of elements in a data block: $O(s\lambda)$. To further improve *POR* scheme and overcome the limitations in previous ones, Xu et.al[16] proposed a private *POR* scheme with constant communication cost based on a

Notation	Description
Chall	Challenge message for integrity check.
Prf	Proof response for integrity check.
PK, SK	System public key and private key
spk, *ssk*	Public key and private key for signature scheme
s	The number of elements in a data block.
n	The number of data blocks in an encoded. data file.
$\|\|G\|\|$	Size of a group element on G
l	Number of blocks accessed in a verification.
$f_{\vec{m}(x)}$	A polynomial $m_0 + m_1 x + \cdots + m_{d-1} x^{d-1}$, where $\vec{m}$ is the coefficient with dimension d.

Table 1: Notations used in this paper

recent proposed polynomial commitment technique[18] and the result from Ref.[10]. By aggregating the proofs into a polynomial, the limitation of communication cost in SW's private POR scheme is overcome in Xu et.al.'s construction[16]. The main drawback of this proposed scheme is that the data owner has to stay online and perform onerous works of verification for the each request of integrity check from other clients, which will inevitably introduce heavy communication cost and computation cost to the data owner, especially when multiple clients submit integrity check at the same time. How to support *POR* scheme with public verifiability and constant communication cost is pointed out as an open problem.

3. MODEL AND ASSUMPTION

The description of key notations used in this paper is summarized in Table 1

3.1 System Model

In this work, we follow the *POR* model that is consistent with most existing *POR* schemes[17, 23, 8, 16]. We consider a *POR* system that has three participating entities: *Data owner*, *Client*, and *Cloud server*. Data owner has a collection of data and stores them on cloud server after ensure coding together with the corresponding authentication tags. The client who shares the stored data with the owner can access it with integrity check, which can also be performed as an independent procedure. To check the integrity of data files, the client generates a challenge message and sends it to the cloud server. The cloud server then responses the computed proof for the selected file blocks to the client. After receiving the proof, the client can verify the integrity of data files through the verification algorithm. W.l.o.g., we define the 1-round version of our *POR* model, which contains four algorithms, *KeyGen*, *Setup*, *Prove* and *Verify*, as below:

- **KeyGen:** Given a selected security parameter λ, the randomized *KeyGen* algorithm outputs the system public key and private key as (PK, SK).

- **Setup:** Given a data file $M \in \{0,1\}^*$ and the public-private key pair (PK, SK), the *Setup* algorithm generates the encoded file $\hat{M}$ as well as the corresponding authentication tag σ, which will be stored on the server.

- **Prove:** Given the public key PK, encoded file $\hat{M}$, authentication tag σ and a challenge message *Chall*, *Prove* algorithm produces a proof response *Prf*.

- **Verify:** Given the public key PK and the *Prf*, the Verify algorithm checks the data integrity and outputs result as either accept or reject.

3.2 Security Model

We consider the storage server as untrusted and potentially malicious, which is consistent with existing *POR* schemes[17, 23, 16]. In our construction, we would like our *POR* scheme to be correct and sound. For the correctness of our scheme, we require that our *Verify* algorithm accepts a valid proof generated from all key pairs (PK, SK), all files $M \in \{0,1\}^*$, all encoded files $\hat{M}$ and authentication tags σ. For the soundness of our scheme, if any malicious cloud server can generate a proof and convince the *Verify* algorithm that it actually stores $\hat{M}$ correctly, it has to yield up the right $\hat{M}$ for the proof generation. By following Ref.[17, 23, 16], we define the security game for the soundness of our *POR* scheme as below.

DEFINITION 3.1. *Let* $\nabla = (KeyGen, Setup, Prove, Veiry)$ *be a POR scheme and A be a probabilistic polynomial-time adversary. Consider the following security game among a trust authority, a challenger and A.*

- The trust authority runs $KeyGen(1^\lambda) \rightarrow (PK, SK)$ and gives PK to the adversary A.

- The adversary A choses a data file M and sends it to the trust authority. The authority then runs *Setup*$(M, SK, PK) \rightarrow (\sigma, \hat{M})$ and responses the encoded data file $\hat{M}$ together with the authentication tag σ back to A.

- With regard to the data file M chosen by adversary A, the challenger picks a random challenge *Chall* and sends it to A.

- According to the received challenge *Chall*, the adversary A produces a proof response *Prf* by running an arbitrary algorithm *Art*$(\hat{M}, \sigma, PK) \rightarrow$ *Prf* rather than the *Prove* algorithm. The proof response *Prf* is sent back to the challenger.

- The challenger verifies *Prf* by running *Verify* algorithm with *Prf* and PK. The output of $Verify$(*Prf*, PK) is denoted as *Rst*.

- The adversary wins the game if and only if he can produce a Prf with data file $\hat{M}', \hat{M}' \neq \hat{M}$ and make the challenger generate *Rst* as *accept* through the *Verify* algorithm.

We say that ∇ is sound if any probabilistic polynomial-time adversary A can win the game with at most a negligible probability.

3.3 Assumption

DEFINITION 3.2. ***Computational Diffie-Hellman (CDH) Assumption[9]***

Let $a, b \overset{R}{\leftarrow} Z_p^*$. *Given input as* (g, g^a, g^b), *where* g *is a generator of a cyclic group* G *of order* p. *It is computationally intractable to compute the value* g^{ab}.

DEFINITION 3.3. ***t-Strong Diffie-Hellman (t-SDH) Assumption****[6]*

Let $\alpha \xleftarrow{R} Z_p^*$. Given input as a $(t+1)-$tuple $(g, g^{\alpha}, \cdots, g^{\alpha^t}) \in G^{t+1}$, where G is a cyclic group of prime order p and g is the generator of G. For any probabilistic polynomial time adversary(PPT_{Adv}), the probability $Pr[PPT_{Adv}(g, g^{\alpha}, \cdots, g^{\alpha^t}) = (c, g^{\frac{1}{\alpha+c}})]$ is negligible for any value of $c \in Z_p^*/-\alpha$.

DEFINITION 3.4. ***t-Bilinear Strong Diffie-Hellman (t-SDH) Assumption****[14]*

Let $\alpha \xleftarrow{R} Z_p^*$. Given input as a $(t+1)-$tuple $(g, g^{\alpha}, \cdots, g^{\alpha^t}) \in G^{t+1}$, where G is a multiplicative cyclic group of prime order p and g is the generator of G. For any probabilistic polynomial time adversary(PPT_{Adv}), the probability $Pr[PPT_{Adv}$ $(g, g^{\alpha}, \cdots, g^{\alpha^t}) = (c, e(g,g)^{\frac{1}{\alpha+c}})]$ is negligible for any value of $c \in Z_p^*/-\alpha$.

4. TECHNIQUE PRELIMINARIES

4.1 Bilinear Map

A bilinear map[7] is a map $e : G \times G \rightarrow G_1$, where G and G_1 are two multiplicative cyclic groups of the same prime order p. A bilinear map has the following properties:

- **Bilinear:** For all $g_1, g_2 \in G$ and $a, b \xleftarrow{R} Z_p^*$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
- **Computable:** There exists an computable algorithm that can compute e efficiently.
- **Non-degenerate:** For $g \in G$, $e(g,g) \neq 1$

4.2 Constant Size Polynomial Commitment

A secure polynomial commitment scheme enables a committer to commit to a polynomial with a short string, which can be used by a verifier to confirm claimed evaluations of the committed polynomial. By utilizing an algebraic property of polynomials $f(x) \in Z[x]$: $(x-r)$ perfectly divides the polynomial $f(x) - f(r), r \xleftarrow{R} Z_p^*$, Kate et.al.[18] proposed a polynomial commitment scheme with constant communication size. In their scheme, a committer of polynomial $f(x)$ can generate a proof with constant size to verify the correctness of the polynomial evaluation $f(r)$, where $x = r$ is an index on the polynomial. Specifically, we summarize Kate et.al.'s scheme[18] as below.

- *Setup*$(1^{\lambda}, t)$**:** Given a security parameter λ and a fixed number t, a trust authority generates a public-private key pair (PK, SK):

  $$PK = (G, G_1, g, g^{\alpha}, \cdots, g^{\alpha^t}) \ SK = \alpha \xleftarrow{R} Z_p^*$$

  where G and G_1 are two multiplicative cyclic groups with prime order p(λ bits security), g is the generator of G and $e : G \times G \rightarrow G_1$.
- $Commit(PK, f_{\vec{m}}(x))$**:** For polynomial $f_{\vec{m}}(x) \in Z_p[x]$ with coefficient vector $\vec{m} = (m_0, m_1, \cdots, m_{s-1}) \xleftarrow{R} Z_p^*$, a committer computes the commitment $C = g^{f_{\vec{m}}(x)} \in G$ and publishes C.
- $CreateWitness(PK, f_{\vec{m}}(x), r)$**:** For any index $r \xleftarrow{R} Z_p^*$, the committer divides polynomial $f_{\vec{m}}(x) - f_{\vec{m}}(r)$ with $(x-r)$ and outputs $\vec{w}$ as result, where $\vec{w} = (w_0, w_1, \cdots, w_{s-1})$ and $f_{\vec{w}}(x) \equiv \frac{f_{\vec{m}}(x) - f_{\vec{m}}(r)}{(x-r)}$. Then, the witness ψ is computed as $\psi = g^{f_{\vec{w}}(x)}$ based on PK.
- $VerifyEval(PK, C, r, f_{\vec{m}}(r), \psi)$**:** A verifier verifies that $f_{\vec{m}}(r)$ is the evaluation at the index r of the polynomial committed to by C as:

  $$e(C, g) \overset{?}{=} e(\psi, g^{\alpha}/g^{r}) \cdot e(g,g)^{f_{\vec{m}}(r)}$$

Due to the space limitation, please refer to Ref.[18] for the detail correctness and security proofs of this scheme.

5. CONSTRUCTION OF PCPOR

5.1 Scheme Description

Let $e : G \times G \rightarrow G_1$ and H be the one-way hash function[15], where G is a multiplicative cyclic group of prime order p and g be a generator of G. We define $f_{\vec{c}(x)}$ as a polynomial with coefficient vector $\vec{c} = (c_0, c_1, \cdots, c_{s-1})$ and describe our *PCPOR* scheme as follows.

- $KeyGen(1^{\lambda}) \rightarrow (PK, SK)$**:**

  Choose a random $(\lambda+1)$ bits safe prime p and generate a random signing keypair $((spk, ssk) \xleftarrow{R} SKg)$ using BLS signature[6]. Choose two random numbers $\alpha, \epsilon \xleftarrow{R} Z_p^*$, compute $v \leftarrow g^{\epsilon}$,$\kappa \leftarrow g^{\alpha\epsilon}$ and $\{g^{\alpha^j}\}_{j=0}^{s-1}$. Then, the public and private keys are

  $$PK = \{p, v, \kappa, spk, \{g^{\alpha^j}\}_{j=0}^{s-1}\}, SK = \{\epsilon, ssk, \alpha\}$$
- *Setup*$(PK, SK, M) \rightarrow (M^*, \sigma, \tau)$**:**

  Given a data file M, obtain M' by applying ensure code(e.g.Reed-Solomon code[22]). Split M' into n blocks, each of which is s elements long: $\{m_{ij}\}_{1 \leq i \leq n, 0 \leq j \leq s-1}$. Choose a random file name *name* from some sufficiently large domain($e.g.Z_p^*$) and s random number $u_0 \cdots u_{s-1} \xleftarrow{R} Z_p^*$. Let τ_0 be "$name||n||u_0|| \cdots ||u_{s-1}$"; the file tag τ is τ_0 together with a signature on τ_0 under ssk: $\tau \leftarrow \tau_0||SSig_{ssk}(\tau_0)$. For each data $block_i, 1 \leq i \leq n$, an authentication tag is computed as:

  $$\sigma_i = (g^{H(name||i)} \cdot \prod_{j=0}^{s-1} g^{u_j m_{ij} \alpha^j})^{\epsilon} \qquad (1)$$
  $$= (g^{H(name||i)} \cdot g^{f_{\vec{\beta}_i}(\alpha)})^{\epsilon}$$

  where $\beta_{i,j} = m_{i,j}u_j$ and $\vec{\beta_i} = \{\beta_{i,0}, \beta_{i,1}, \cdots, \beta_{i,s-1}\}$. The processed file M^* together with the authentication tags σ_i are outsourced for storage, where M^* is $\{m_{i,j}\}, 1 \leq i \leq n, 0 \leq j \leq s-1$.
- $Verify(PK, \tau) \rightarrow Chall$**:**

  Stage 1: Verify the signature on τ: if the signature is not valid, reject and halt; otherwise, parse τ to recover $name, n, u_0, u_1, \cdots, u_{s-1}$. Now, choose a random element $\rho \xleftarrow{R} Z_p^*$ and a random $l-$elements subset I of the set $[1, n]$ using Goldreich[13]'s $(\gamma, \delta) - hitter$. Produce the challenge message as

$$Chall = \{r, \rho, I\}$$

where, $r \overset{R}{\leftarrow} Z_p^*$. Challenge the data storage server with $Chall$.

- $Prove(PK, M^*, Chall, \tau, r) \rightarrow (\psi, y, \sigma)$**:**

  Parse the challenge message $Chall$ as $\{r, \rho, I\}$ and generate a l-element set Q as (i, v_i), with the $i \in [1, n]$, where $v_i = \rho^i \ mod \ p$ and $\rho \overset{R}{\leftarrow} Z_p^*$. Parse the processed file M^* as $m_{i,j}, 1 \leq i \leq n, 1 \leq j \leq s$ together with σ_i and compute

  $$\sigma = \prod_{(i,v_i) \in Q} \sigma_i^{v_i} \tag{2}$$

  We denote vector

  $$\vec{A} = (u_0 * \sum_{(i,v_i) \in Q} v_i m_{i,0}, \cdots, u_{s-1} * \sum_{(i,v_i) \in Q} v_i m_{i,s-1})$$

  Compute

  $$y = f_{\vec{A}}(r) \tag{3}$$

  As we mentioned before, polynomials $f(x) \in Z[x]$ have the algebraic property that $(x-r)$ perfectly divides the polynomial $f(x) - f(r), r \overset{R}{\leftarrow} Z_p^*$. Now, divide the polynomial $f_{\vec{A}}(x) - f_{\vec{A}}(r)$ with $(x-r)$ using polynomial long division, and denote the coefficients vector of the resulting quotient polynomial as $\vec{w} = (w_0, w_1, \cdots, w_{s-2})$, that is, $f_{\vec{w}}(x) \equiv \frac{f_{\vec{A}}(x) - f_{\vec{A}}(r)}{x-r}$. Produce

  $$\psi = \prod_{j=0}^{s-2} (g^{\alpha^j})^{w_j} = g^{f_{\vec{w}}(\alpha)} \tag{4}$$

  Response $Prf = \{\psi, y, \sigma\}$.

- $Verify(PK, Prf) \rightarrow Rst$**:**

  Stage 2: After receiving the proof response *Prf*, compute

  $$\eta_i = v^{-H(name||i)v_i}, (i, v_i) \in Q \tag{5}$$

  $$\eta = \prod_{(i,v_i) \in Q} \eta_i \tag{6}$$

  where $v = g^\epsilon$ in PK, $i \in [1, n]$, $v_i = \rho^i \ mod \ p$. Parse *Prf* as $\{\psi, y, \sigma\}$ and check

  $$e(\psi, \kappa \cdot v^{-r}) \overset{?}{=} e(\sigma \cdot \eta, g) \cdot e(g^{-y}, v) \tag{7}$$

  where $\kappa = g^{\epsilon\alpha}$ in PK. If Eq.7 holds, then output Rst as accept; otherwise, output Rst as reject.

- **Correctness** For a storage server who honestly responses the challenge with a $Prf = \{\psi, y, \sigma\}$, we can analyze the correctness of Eq.7 from the left part and right part as:
  Left Part:

  $$\begin{aligned} & e(\psi, \kappa \cdot v^{-r}) \\ & = e(g^{f_{\vec{w}}(\alpha)}, g^{\epsilon(\alpha - r)}) \\ & = e(g, g)^{\frac{f_{\vec{A}}(\alpha) - f_{\vec{A}}(r)}{\alpha - r} \cdot \epsilon(\alpha - r)} \\ & = e(g, g)^{\epsilon(f_{\vec{A}}(\alpha) - f_{\vec{A}}(r))} \end{aligned} \tag{8}$$

  Right Part:

  $$\begin{aligned} & e(\sigma \cdot \eta, g) \cdot e(g^{-y}, v) \\ & = e(g^{\epsilon(\sum_{(i,v_i) \in Q} H(name||i)v_i + f_{\vec{A}}(\alpha)) + \epsilon \sum_{(i,v_i) \in Q} -H(name||i)v_i} \\ & , g) \cdot e(g^{-f_{\vec{A}}(r)}, g^\epsilon) \\ & = e(g, g)^{\epsilon f_{\vec{A}}(\alpha)} \cdot e(g, g)^{-\epsilon f_{\vec{A}}(r)} \\ & = e(g, g)^{\epsilon(f_{\vec{A}}(\alpha) - f_{\vec{A}}(r))} \end{aligned} \tag{9}$$

  where $v = g^\epsilon$ and $\kappa = g^{\epsilon\alpha}$. From the above Eq.8 and Eq.9, it is easy to see that the scheme is correct if the the storage sever generate the Prf honestly.

5.2 Security Proof

5.2.1 The underlying authenticator is unforgeable

Theorem 5.1. *If t-Strong Diffie-Hellman(t-SDH) Assumption holds and $g^{f_{\vec{c}}(x)}$ can be forged by an existed probabilistic polynomial time adversary A, we can construct an algorithm B that uses A to efficiently compute the solution to t-SDH problem.*

Proof. Suppose there exists a probabilistic polynomial time adversity A can forge $f_{\vec{c}}^1(\alpha)$ such that $g^{f_{\vec{c}}^1(\alpha)} = g^{f_{\vec{c}}(\alpha)}$, where $\vec{c}$ is the coefficient vector, he/she obtains $g^{f_{\vec{c}}^2(\alpha)} = g^{f_{\vec{c}}(\alpha)} / g^{f_{\vec{c}}^1(\alpha)} = g^{f_{\vec{c}}(\alpha) - f_{\vec{c}}^1(\alpha)} \in Z_p[x]$. Since $f_{\vec{c}}^1(\alpha) = f_{\vec{c}}(\alpha)$ and $f_{\vec{c}}^2(\alpha) = 0$, i.e., α is a root of polynomial $f_{\vec{c}}^2(x)$ and by factoring $f_{\vec{c}}^2(x)$[24], B can easily find $SK = \alpha$ and solve the instance of the t-SDH problem given by the system parameters. □

Theorem 5.2. *If the signature scheme used for file tags is existentially unforgeable, Computational Diffie-Hellman(CDH) problem is hard, t-Strong Diffie-Hellman(t-SDH) Assumption and t-Bilinear Strong Diffie-Hellman(t-BSDH) Assumption hold. In our proposed scheme, the prover's response (y, ψ, σ) is unforgeable.*

Proof. Suppose a probabilistic polynomial time adversity can generate a response (y', ψ', σ') to forge (y, ψ, σ) after receiving a challenge message from the verifier, $(y', \psi', \sigma') \neq (y, \psi, \sigma)$. Since both the forged and the true responses can be accepted by the verification algorithm, we can get the following two equations:

$$e(\psi, \kappa \cdot v^{-r}) = e(\sigma \cdot \eta, g) \cdot e(g^{-y}, v) \tag{10}$$

$$e(\psi', \kappa \cdot v^{-r}) = e(\sigma' \cdot \eta, g) \cdot e(g^{-y'}, v) \tag{11}$$

Dividing Eq.10 with Eq.11, we obtain:

$$\frac{e(\psi, g)^{\epsilon(\alpha - r)}}{e(\psi', g)^{\epsilon(\alpha - r)}} = \frac{e(g, g)^{\epsilon E - \sum_{(i,v_i) \in Q} H(name||i)v_i - y}}{e(g, g)^{\epsilon E' - \sum_{(i,v_i) \in Q} H(name||i)v_i - y'}}$$

$$\left(\frac{e(\psi, g)}{e(\psi', g)}\right)^{\epsilon(\alpha - r)} = e(g, g)^{\epsilon(E - E') + y' - y} \tag{12}$$

where we denote σ as $g^{E\epsilon}$ and σ' as $g^{E'\epsilon}$ for simplicity.

Now we do a case analysis on whether $\sigma = \sigma'$.

Case 1: $\sigma \neq \sigma'$. Since $g^{E\epsilon} = \sigma$ and $g^{E'\epsilon} = \sigma'$, we get $E \neq E'$. As $\left(\frac{e(\psi, g)}{e(\psi', g)}\right)^{\epsilon(\alpha - r)}$, $e(g, g)^{y' - y}$ and $e(g, g)^{E\epsilon} = e(g, \sigma)$ are known to the adversary, we rewrite Eq.12 as

$$\Upsilon = e(g, g)^{\epsilon E'} \tag{13}$$

where we denote $\Upsilon = e(g,g)^{E\epsilon+y'-y} / \left(\frac{e(\psi,g)}{e(\psi',g)}\right)^{\epsilon(\alpha-r)}$ as a known value to the adversary.

Recall that in this proof, the Computational Diffie-Hellman (CDH) problem is hard. If the any probabilistic polynomial time adversity A can find E' with non-negligible probability and make the Eq.13 hold, we can construct an algorithm B that uses A to solve the instance of CDH problem. Specifically, given $E' \neq E$ found by A, which makes Eq.13 hold, B can get $\Upsilon = e(g,g)^{\epsilon E'}$ as solution for CDH problem of $e(g,g)^{\epsilon}$ and $e(g,g)^{E'}$. Therefore, no probabilistic polynomial time adversity can find a valid forged response $(y,\psi,\sigma) \neq (y',\psi',\sigma')$ and $\sigma \neq \sigma'$ with non-negligible probability.

Case 2: $\sigma = \sigma'$. In this case, we can rewrite Eq.12 as:

$$\left(\frac{e(\psi,g)}{e(\psi',g)}\right)^{\epsilon(\alpha-r)} = e(g,g)^{y'-y} \qquad (14)$$

We further do a case analysis on whether $y = y'$.

Case 2.1: $y = y'$. As $(y,\psi,\sigma) \neq (y',\psi',\sigma')$, $\sigma = \sigma'$ and $y = y'$, we can obtain that $\psi \neq \psi'$. In this case, since $y = y'$, we further rewrite the Eq.14 as:

$$\left(\frac{e(\psi,g)}{e(\psi',g)}\right)^{\epsilon(\alpha-r)} = 1 \qquad (15)$$

We know that $\psi \neq \psi'$, i.e., $\frac{e(\psi,g)}{e(\psi',g)} \neq 1$, and $\epsilon \neq 0$, we can infer $\alpha = r$ from Eq.15. In our scheme, r is known to the adversary(i.e., the adversary can find $SK = \alpha$). As we mentioned in Theorem 5.1, by finding $SK = \alpha$, we can solve the instance of the t-SDH problem by given the system parameters. Therefore, no valid forged response $(y,\psi,\sigma) \neq (y',\psi',\sigma')$ and $y = y'$ can be found by probabilistic polynomial time adversity A with non-negligible probability.

Case 2.2: $y \neq y'$. From Eq.14 and $y \neq y'$, we can imply that $\alpha \neq r$. In this case, we show how to construct an algorithm B, using the existed adversary, that can break the t-Bilinear Strong Diffie-Hellman(t-BSDH) Assumption with a valid solution $(-r, \left(\frac{e(\psi,g)}{e(\psi',g)}\right)^{\frac{1}{y'-y}})$.

We denote ψ as g^{θ} and ψ' as $g^{\theta'}$, and then we can rewrite Eq.14 as :

$$\begin{aligned} \left(\frac{e(\psi,g)}{e(\psi',g)}\right)^{\epsilon(\alpha-r)} &= \frac{e(g,g)^{-y}}{e(g,g)^{-y'}} \\ \theta\epsilon(\alpha-r)+y &= \theta'\epsilon(\alpha-r)+y' \\ \frac{\epsilon(\theta-\theta')}{y'-y} &= \frac{1}{\alpha-r} \end{aligned} \qquad (16)$$

Therefore, algorithm B can compute

$$\left(\frac{e(\psi,v)}{e(\psi',v)}\right)^{\frac{1}{y'-y}} = e(g,g)^{\frac{\epsilon(\theta-\theta')}{y'-y}} = e(g,g)^{\frac{1}{\alpha-r}} \quad (17)$$

and returns $(-r, e(g,g)^{\frac{1}{\alpha-r}})$ as a solution for t-BSDH instance. It is easy to see that the success probability of solving the instance is the same as the success probability of the adversity, and the time required is a small constant larger than the time required by the adversary.

Therefore, *Theorem* 5.2 is proved. □

6. PERFORMANCE EVALUATION

To measure the impact of our improvements from the existing schemes and evaluate the practicality of our proposed *PCPOR* scheme, we numerically analyze it and implement it using Amazon EC2 and S3 cloud platform.

6.1 Numerical Analysis

In this section, we numerically evaluate the performance our proposed *PCPOR* scheme in terms of communication cost, computation cost and storage cost. We compare our *PCPOR* scheme with the existing *POR* techniques[23, 10, 16] and summarize the result in Table 2. For simplicity, in the following part of this paper, we denote the complexity of one multiplication operation on Group G as MUL and that of one exponentiation operation on Group G as EXP[1]. Furthermore, we use ZADD and ZMUL to represent the addition and multiplication operations on Z_p^* respectively. PRF is used to denote pseudorandom function and $||G||$ denotes the size(in number of bits) of a group element on G.

6.1.1 Communication

In our proposed *PCPOR* scheme, the communication cost comes from the challenge message *Chall* and the proof response *Prf* in each verification request. For the challenge message, it consists of a $l-$elements subset I and two random elements $\rho, r \overset{R}{\leftarrow} Z_p^*$. By utilizing Goldreich[13]'s $(\gamma,\delta)-$ *hitter*[2], we can represent the subset I with $log^{|F|} + 3log^{(1/\delta)}$ bits, where $|F|$ is the size of the encoded data file, and δ is the error probability. In particular, given a data file less than 1024 TB(i.e. $|F| = 2^{43}$ bits, which is enough for most practical scenarios) and set $\delta = 2^{-80}$, the subset I can be represented with 283 bits. As a result, the total cost of *Chall* in our *PCPOR* scheme is $2\lambda + 283$ bits when the data file size is smaller than 1024 TB. For the proof response, by aggregating proof information into 3 tags ψ, σ and y, the total size of the proof response is $2||G|| + \lambda$ bits, where ψ and σ are two group elements and y the result of a polynomial. Therefore, the total complexity of communication cost in our *PCPOR* scheme is $O(||G||)$. When setting reasonable parameters for our *PCPOR* scheme(e.g. $\lambda = 160$, $||G|| = 1024$ bits and the data file is less than 1024 TB), the total communication cost becomes constant size.

Now, we compare the existing *POR* schemes[23, 10, 16] with our *PCPOR* scheme and summarize the result in Table 2. In Ref.[23], the complexity of challenge message and proof response are $O(\lambda^2)$ and $O(s\lambda)$ respectively, where s is the number of elements in each encoded block. Compared to our *PCPOR* scheme, which has constant communication cost in for both challenge and response processes, this scheme have a communication size of proof response linear to s and a challenge message cost λ times to ours. In the *POR* scheme proposed by Dodis et al.[10], the size of challenge message is reduced to $O(\lambda)$, which is the same as in

[1] When the operation is on the elliptic curve, EXP means scalar multiplication operation and MUL means one point addition operation.

[2] In Goldreich[13]'s $(\gamma,\delta) - hitter$, it is guaranteed that any subset $Sub \subset [1,n]$ with size $|Sub| \geq |F|(1 - gamma)$, $Pr[I \cap Sub \neq \varnothing]$. For more details, please refer to Ref.[13].

Scheme	Public verifiability	Comm. complexity (Challenge) (bits)	Comm. complexity (Response) (bits)	Comp. Cost (Server)	Comp. Cost (Challenger)	Storage Cost (Server) (bits)	Storage Cost (Challenger) (bits)	Data Preparation
[23]	Yes	$O(\lambda^2)$	$O(s\lambda)$	l(MUL+EXP)+ sl(ZADD+ZMUL)	$(s+l)$MUL+ $(s+l)$EXP+ $2Pairing$	$(1+\frac{1}{s})\|F\|$	$\lambda + \|\|G\|\|$	$(s+1)$MUL+ $(s+2)n$EXP
[10]	No	$O(\lambda)$	$O(s\lambda)$	$(sl+l)$(ZMUL +ZADD)	$(s+l)$(ZMUL+ ZADD)+lPRF	$(1+\frac{1}{s})\|F\|$	2λ	lPRF+ n(ZMUL+ZADD)
[16]	No	$O(\lambda)$	$O(\lambda)$	$(s-1)$(EXP+MUL)+ $(s+l+sl)$(ZMUL +ZADD)	2EXP+lPRF l(ZMUL+ZADD)	$(1+\frac{1}{s})\|F\|$	$3\lambda + 80$	lPRF+ n(ZMUL+ZADD)
PCPOR	Yes	$O(\lambda)$	$O(\|\|G\|\|)$	$(l+s-1)$MUL+ $(2l+s-1)$EXP+ slZADD+$s(l+2)$ZMUL	lMUL+ $(2l+2)$EXP+ $3Pairing$	$(1+\frac{1}{s})\|F\|$	$\lambda + 4\|\|G\|\|$	$(s+1)$MUL+ $(2s+2)n$EXP

Table 2: Complexity Summary: in this table, MUL is one multiplication operation on Group G, EXP is one exponentiations operation on Group G, PRF denotes the pseudorandom function, ZADD and ZMUL represent the addition and multiplication operations on Z_p^* respectively; λ is the security parameter chosen for the system, $||G||$ is the size(in number of bits) of a group element on G, $|F|$ is size of data file, n is number of encoded blocks for the data file, s is the number of elements in each block and l is number of blocks selected for verification. Note: if we fix the security parameter λ, the communication complexity in our $PCPOR$ scheme becomes constant.

.

our $PCPOR$ scheme. However, the cost of proof response in their scheme is still $O(s\lambda)$. Considering only private verification, Xu et al.[16]'s POR scheme reduces communication cost to constant size in each single verification. Nevertheless, it is worthy of notice that their scheme only supports private verification, which centralizes all the verification tasks to the data owner and thus introduces onerous burden in communication to it. Specifically, if the data owner wants to share his outsourced data with other individuals/organizations, it has to stay online and processes all verifications by himself. As a result, the communication cost for the owner increases linearly to the number of verification requests at the same time. Differently, with the public verifiability of our $PCPOR$ scheme, each challenger is able to conduct the verification independently with constant communication cost and the data owner can go off-line after outsourcing his data.

6.1.2 Computation

As mentioned in Section 5.1, our $PCPOR$ scheme is composed of 4 algorithms: *KeyGen*, *Setup Prove* and *Verify*. Among these algorithms, *KeyGen* and *Setup* are performed by the data owner as data preparation process. To generate the public key PK as well as the private key SK for the system, the data owner performs $(s+3)$EXP operations using the *KeyGen* algorithm. In the *Setup* procedure, to process an encoded data file with n blocks, each of which has s elements, $(2s+2)n$EXP and $(s+1)$MUL operations are needed to generate the authentication tags. Note that the data preparation process in our $PCPOR$ scheme is one-time cost for the data owner, it can go off-line after finishing this process. For the computation cost in each verification request, the server needs to perform $(l+s-1)$MUL, $(2l+s-1)$EXP, slZADD and $s(l+2)$ZMUL operations to generate the proof response, where l is number of blocks chosen in each verification. After receiving the proof information, the challenger first conducts lMUL and $2l$EXP operations to generate η. Then, 2 more EXP and 3 *Pairing* operations are needed for the final verification. Therefore, as shown in Table 2, the total computation cost for the challenger in one verification is lMUL+$(2l+2)$EXP+$3Pairing$.

We now compare our $PCPOR$ scheme with the existing POR schemes[23, 10, 16] and show the result in Table 2. Beginning with the public POR scheme in Ref.[23], although our $PCPOR$ will introduce $s-1$ more MUL, $l+s-1$ more EXP and $2s$ more ZMUL operations to the server side, it achieves the same computation cost level for the challenger side(client). As the cloud sever side is always much powerful than the challenger side(e.g. Amazon EC2 vs Mobile devices), the additional computations brought to server side in our $PCPOR$ can be easily handled in practical scenarios and will have little influence on the scheme's performance. When considering the two private POR schemes[10, 16] as shown in Table 2, which have almost the same computation cost for challenger side except for 2 more EXP operations in Ref.[16], our $PCPOR$ scheme will cause relative higher computation cost by replacing the operations on Z_p^* to operations on G in each verification. However, it is notable that these private POR schemes[10, 16] have to centralize all the verification tasks to the data owner(i.e., the computation cost on the data owner is linear to the number of simultaneous verification requests), which requires the data owner to keep online and causes heavy computation burden on it. On the contrary, in our $PCPOR$, the computation tasks for each verification is distributed to the challengers without contacting the data owner. This public verifiability makes our scheme easy to scale and have practical computation cost on each challenger. For the computation cost for data preparation, our $PCPOR$ scheme introduces sn more EXP operations compared to the public POR scheme in Ref.[23]. Since the data preparation process is one-time cost in our scheme and will not influence the realtime verification process, the additional operations in this process can be accepted in practical scenarios. Compared to the private POR schemes in Ref.[10, 16], as shown in Table 2, our $PCPOR$ scheme and scheme in Ref.[23] need relative higher computation cost for the preparation due to the requirement of group operations for the public verifiability. Nevertheless, as we mentioned above in this section, these additional one-time computation cost in data preparation process is reasonable and acceptable.

6.1.3 Storage

In this section, we first analyze the storage cost of our $PCPOR$ scheme on both challenger side and server side, and then compare it with the existing POR schemes[23, 10, 16]. The results of our analysis are summarized in Table 2. For the challenger side, our $PCPOR$ scheme only requires the challenger to store partial public key $PK : \{p, v, k, spk, g\}$ to generate the challenge message $Chall$ and perform verification algorithm $Verify$. Thus, the size of storage cost for each challenger is $4||G|| + \lambda$ bits. Compared to the existing POR schemes[23, 10, 16], which require $||G|| + \lambda$ bits, 2λ bits and $3\lambda + 80$ bits respectively for the challenger side storage cost, our $PCPOR$ scheme achieves the same storage cost level as demonstrated in Table 2. For storage overhead on the server side, it mainly comes from the authentication tags for the encoded data blocks. In our $PCPOR$ scheme, each authentication tag is a group element with λ bits, thus the total size for tags is $n\lambda$ bits. As the total encoded data file size $|F| = ns\lambda$ bits, we represent the total storage overhead on server side as $(1 + \frac{1}{s})|F|$ bits, which equals to existing schemes'[23, 10, 16] storage cost on server sides when the values of s and λ are same.

Note that the communication cost in our $PCPOR$ scheme is independent to s as we mentioned in Section 6.1.1, which enables our scheme to reduce the storage cost on server side by adjusting the value of s. However, in Ref.[23, 10], as the communication cost for proof response is linear to the value of s, reducing the storage cost will lead to the sacrifice of communication performance.

6.2 Experimental Evaluation

6.2.1 Experiment Setup

We implemented our proposed $PCPOR$ scheme on Amazon S3 Cloud Platform using C++ with GNU MP library[12] and the Pairing-Based Cryptography library[20]. To enhance the efficiency for data preparation process in our scheme, we utilize the $Phoenix{+}{+}$ library[21]. The test machine for data owner and challenger are laptops running Mint Linux 13 with 2.50GHz Intel i5-2520M CPU and 8GB memory. For the cloud server, we utilize node on Amazon EC2 running Red Hat Enterprise Linux 6.3 with 8 Cores CPU and 30GB memory. The size of test data files(after error erasure encoding) in our experiments varies from 128MB to 2048MB. We do not conduct experiments on more large dataset because our experiment results show that the increase of encoded data file size has little influence on our scheme's communication cost and computation cost. We set the security parameter $\lambda = 160$, which achieves 1024bit security on Group. By following the previous work[16], we change the number of elements in each block s from 40 to 1280. The number of data blocks to access in each verification l from 100 to 1000 based on our error detection probability analysis in Section 7. Each single experiment case is repeated for 10 times and we use the average values as the final results.

6.2.2 Experimental Results

In this section, we provide the experimental results for our $PCPOR$ scheme and summarize them in Figure 1 and Figure 2.

KenGen: Since the public key PK and private key SK in our scheme need $(s + 3)$ EXP operations, the time needed for generating the system PK and SK become linear to s, where s is the number of group elements in one data block. As we show in Figure 1(a), the key generation time varies from 0.24s to 7.79s when we change the value of s from 40 to 1280. However, in our $PCPOR$ scheme, as the key generation is one-time cost and do not to be repeated in the following verification processes, the cost for this process can be easily handled in practical scenarios.

Setup: To measure the computation cost for the data preparation, we use encoded data files with different size(128MB to 2048MB) and vary the number of group elements in one data block(40 to 1280). As demonstrated in Figure 1(b), the preparation time is mainly determined by the encoded data file size and slightly influenced by the value of s, which is consistent with our analysis in Section 6.1.2. Since the computation cost in this process is mainly from the $(2sn + 2n)$ EXP operations and the value of $2ns$ is determined by size of data file when λ is fixed(i.e. $|F| = ns\lambda$), the increase of the encoded data file size greatly influences the total computation cost. At the same time, the increase of s will lead to the decrease of n, thus the cost is slightly reduced when we increase the value of s in our experiments(e.g. for 1024MB data file, the cost is reduced from 356.16s to 347.74s when we vary s from 40 to 1280). Considering 2048MB encoded data file in our experiments, we spend about 700s to process the file when s is from 40 to 1280 on single machine. As it is easy for cloud server to process such proof generation in parallel, this cost is acceptable in practical scenarios. For larger data file, our setup process is easy to be parallelized to enhance efficiency. It is notable that this setup procedure in one-time cost in our $PCPOR$ scheme and will not influence the following verification performance.

Prove: For the measurement of proof generation efficiency in our $PCPOR$ scheme, Figure 2(a) and Figure 2(b) show that the computation cost is mainly affected by the values of l and s, where l the number of blocks accessed in a verification. At the same time, by comparing the results in Figure 2(a) and Figure 2(b), it is easy to see that the proof generation performance has few relationship with the size of encoded data file[3]. Specifically, as demonstrated in Figure 2(a) and Figure 2(b), the computation cost for generating proof on server side increases linearly to l or s when one of them is fixed, which is consistent with the previous analysis in Section 6.1.2. This is because the server needs to compute proof information for all the selected blocks in a verification and the value of s determines the number of terms in the polynomial for proof. By changing s from 40 to 1280 and l from 100 and 1000, the server spends 0.48s to 7.23s to generate the proof response with single node. As we mentioned above, since the size of encoded data file has few influence on the proof generation in our $PCPOR$ scheme, the computation cost for this process is reasonable in practical usages for different size of data files.

Verify: In our experiment for the verification process, we find that the factor mainly affect the computation cost is the value of l, which supports our previous analysis in Section 6.1.2. As shown in Figure 2(c) and Figure 2(d), the computation cost for each verification on the challenger side is linear to the value of l and has few relationship with the value of s as well as the size of encoded data file. By varying

[3] From our experiments, we find the size of encoded data file has few influence on proof generation, we only put the results of 128MB and 1024MB data file here due to the space limitation. The full experiment results for this part can be found in Appendix10.1.

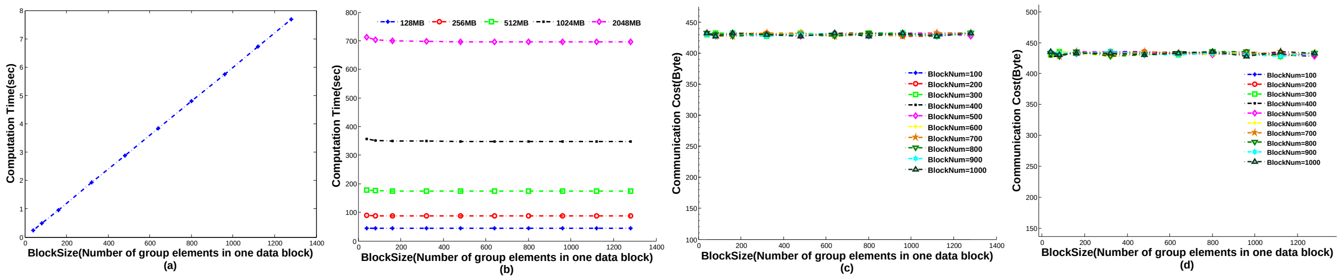


Figure 1: (a) Time Cost for Key Generation (b)Time Cost for Setup (c) Communication Cost for a verification(encoded data file is 128MB) (d) Communication Cost for a verification(encoded data file is 1024MB)

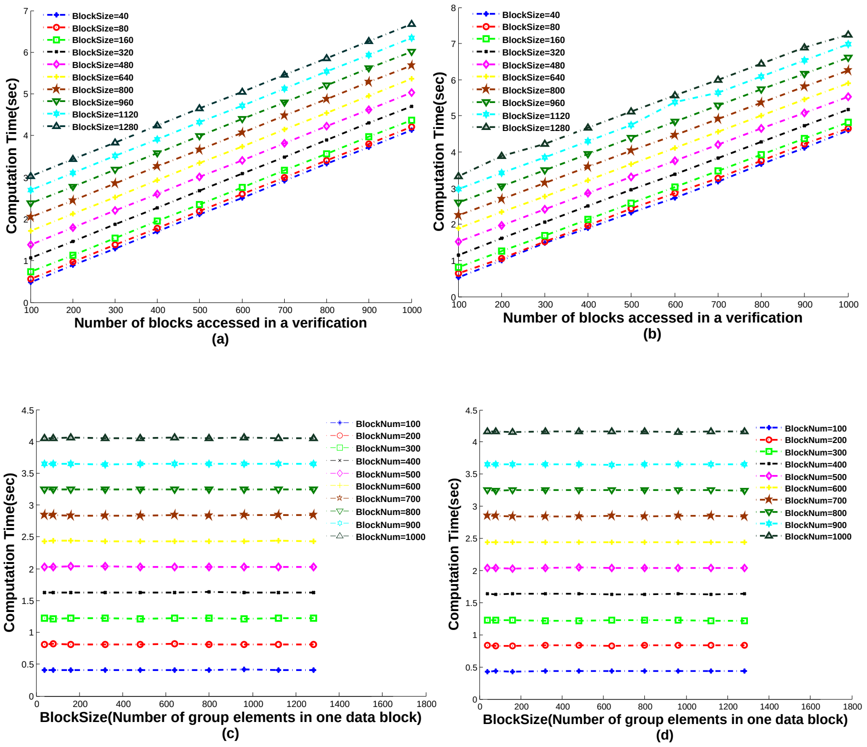


Figure 2: (a) Time cost for proof generation on server side(encoded data file is 128MB) (b)Time cost for proof generation on server side(encoded data file is 1024MB) (c) Client verification time(encoded data file is 128MB) (d)Client verification time(encoded data file is 1024MB)

l from 100 to 1000, the verification costs up to 4.13s for a challenger under different value of s(i.e. from 40 to 1280) and encoded data file size(from 128MB to 2048MB)[4], which is acceptable for practical scenarios.

Communication: To verify the constant communication cost in our *PCPOR* scheme, we vary all the factors(i.e., values of s, l and size of encoded data file) that may influence the communication size in *POR* schemes, which do limit the previous *POR* schemes[23, 10] as shown in Section 6.1.2. As demonstrated in Figure 1(c) and Figure 1(d), the communication cost in our *PCPOR* scheme keeps as a constant value when we change l from 100 to 1000 and s from 40 to 1280. By varying the size of encoded data file from 128MB to 2048MB[5], it increases less than 1byte for communication cost, and thus can still be treated as constant value compared to total cost(i.e. around 429bytes) in *PCPOR* scheme. Note that, although we do not perform experiment on more large data files, from our analysis in Section 6.1.1, it is easy to obtain that a 1024TB data file will only introduce 20bits additional communication cost, which is negligible compared to 1024MB data file under the same λ. Therefore, we can consider communication cost in

[4] We only put the results of 128MB and 1024MB data file here due to the space limitation. The full experiment results for this part can be found in Appendix10.2.

[5] We only put the results of 128MB and 1024MB data file here due to the space limitation. The full experiment results for this part can be found in Appendix10.3.

our *PCPOR* scheme as constant.

From the above experiment results, it shows that the size of encoded data file only influence our *PCPOR* scheme in data setup process, which is one-time cost and does not have any effect on the following verification procedures. Therefore, with the efficient computation performance and constant communication cost, our *PCPOR* scheme has high scalability for different size of data file and bandwidth conditions.

7. DISCUSSION

In this section, we discuss about the error detection probability of our *PCPOR* scheme. As we mentioned in the *Setup* algorithm of *PCPOR* scheme, Reed-Solomon code with rate ℓ is adopted for the data file encoding. For a ℓ Reed-Solomon encoded data file, any ℓ fraction of encoded data blocks can recover the original file. If a data file encoded with ℓ Reed-Solomon code cannot be recovered from the erasure decoding, the probability of accessing a uncorrupted encoded data block will be less then ℓ. In this case, when we randomly choose l independent encoded data blocks and all these blocks are uncorrupted, the probability should be less then ℓ^l.

In our *PCPOR* scheme, we can set $\ell = 0.98$ as previous previous *POR* scheme[16] does. In this case, by checking 200 encoded data blocks, the challenger can have at least 98.24% confidence that the stored data on the server is not corrupted if the $Verify$ algorithm outputs result as accpet. 99.99% confidence can be guaranteed if 1000 encoded data blocks are checked by the challenger.

8. CONCLUSIONS

Proofs of Retrievability(*POR*) technique enables individuals/organizations to ensure the integrity of their outsourced data on a untruest server(e.g.public cloud storage platform). Nevertheless, the existing *POR* schemes either have limitation on communication cost, which is linear to the number of elements in a data block, or only consider private verification, These limitations cause a severe scalability issue in data file size or user number for practical use. In this work, we proposed the first public *POR* scheme with constant communication cost: *PCPOR*. By uniquely incorporating the polynomial commitment technique[18] and other cryptograhphic techniques, our *PCPOR* scheme achieves constant communication size, efficient computation performance as well as low storage overhead, which omit the limitations in previous public *POR* schemes and make it become practical in bandwidth constraint scenarios. What is more, based on the simultaneous realized public verifiability in our *PCPOR* scheme, we release the data owner from onerous verification tasks and make each client check integrity independently, which need to be centralized to the data owner in previous private *POR* scheme with constant communication size. Our security proof based on Computational Diffie-Hellman Assumption, Strong Diffie-Hellman Assumption and Bilinear Strong Diffie-Hellman Assumption shows that our *PCPOR* scheme is secure. We conduct experiments on cloud platforms and demonstrate our *PCPOR* scheme is practical and scalable.

9. REFERENCES

[1] Amazon forum. major outage for amazon s3 and ec2, https://forums.aws.amazon.com/thread.jspa?threadID =19714&start=15&tstart=0.

[2] Amazon web service. summary of the amazon ec2 and amazon rds service disruption in the us east region, http://aws.amazon.com/message/65648/.

[3] Business insider. amazon's cloud crash disaster permanently destroyed many customers' data, http://www.businessinsider.com/amazon-lost-data-2011-4.

[4] Dropbox. dropbox forums on data loss topic, http://forums.dropbox.com/tags.php?tag=data-loss.

[5] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable data possession at untrusted stores. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 598–609, New York, NY, USA, 2007. ACM.

[6] D. Boneh and X. Boyen. Short signatures without random oracles. pages 56–73. Springer-Verlag, 2004.

[7] D. Boneh, B. Lynn, and H. Shacham. Short signatures from the weil pairing. In *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '01, pages 514–532, London, UK, UK, 2001. Springer-Verlag.

[8] K. D. Bowers, A. Juels, and A. Oprea. Proofs of retrievability: theory and implementation. In *Proceedings of the 2009 ACM workshop on Cloud computing security*, CCSW '09, pages 43–54, New York, NY, USA, 2009. ACM.

[9] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Trans. Inf. Theor.*, 22(6):644–654, Sept. 1976.

[10] 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, Berlin, Heidelberg, 2009. Springer-Verlag.

[11] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. Dynamic provable data possession. In *Proceedings of the 16th ACM conference on Computer and communications security*, CCS '09, pages 213–222, New York, NY, USA, 2009. ACM.

[12] GMP. http://gmplib.org/.

[13] O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *J. ACM*, 33(4):792–807, Aug. 1986.

[14] V. Goyal. Reducing trust in the pkg in identity based cryptosystems. In *Proceedings of the 27th annual international cryptology conference on Advances in cryptology*, CRYPTO'07, pages 430–447, Berlin, Heidelberg, 2007. Springer-Verlag.

[15] P. Hawkes, M. Paddon, and G. G. Rose. On corrective patterns for the sha-2 family, 2004.

[16] X. Jia and C. Ee-Chien. Towards efficient provable data possession. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, Seoul, Korea, 2012.

[17] A. Juels and B. S. Kaliski, Jr. Pors: proofs of retrievability for large files. In *Proceedings of the 14th ACM conference on Computer and communications security*, CCS '07, pages 584–597, New York, NY, USA, 2007. ACM.

[18] A. Kate, G. M. Zaverucha, and I. Goldberg. Constant-size commitments to polynomials and their applications. In *ASIACRYPT*, pages 177–194, 2010.

[19] G. Oded. A sample of samplers - a computational perspective on sampling (survey). *Electronic Colloquium on Computational Complexity (ECCC)*, 4(20), 1997.

[20] PBC. http://crypto.stanford.edu/pbc/.

[21] Phoenix++. http://mapreduce.stanford.edu/.

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

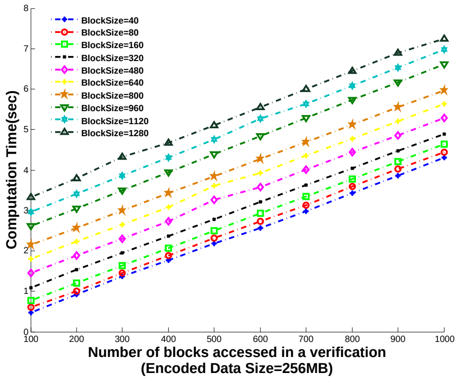
[23] H. Shacham and B. Waters. Compact proofs of retrievability. In *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology*, ASIACRYPT '08, pages 90–107, Berlin, Heidelberg, May 2008. Springer-Verlag.

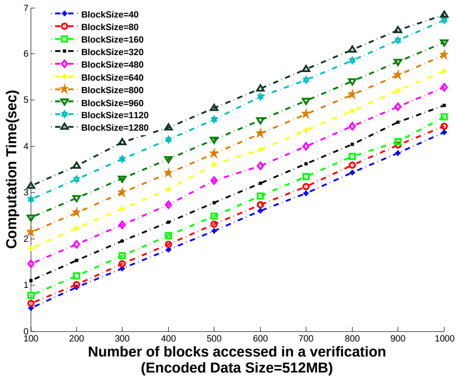
[24] V. Shoup. *A computational introduction to number theory and algebra*. Cambridge University Press, New York, NY, USA, 2005.

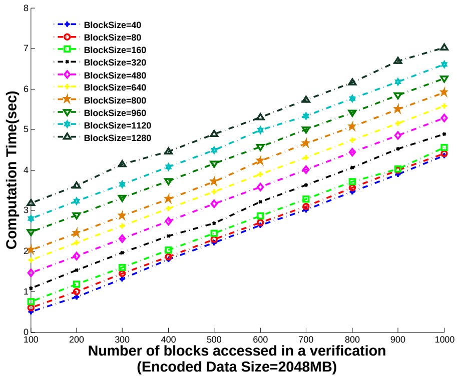
[25] G. Timothy and M. M. Peter. The nist definition of cloud computing. NIST SP - 800-145, September 2011.

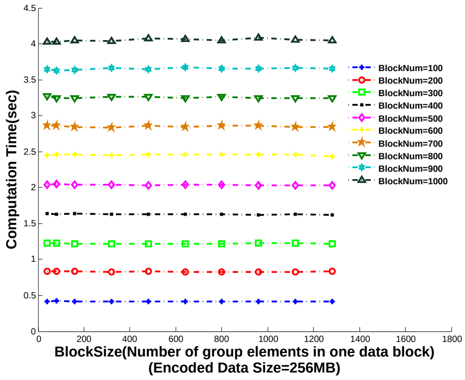
10. APPENDIX

10.1 Full Experiment Results For Proof Generation

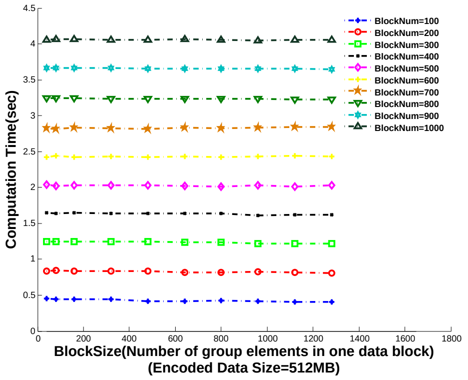


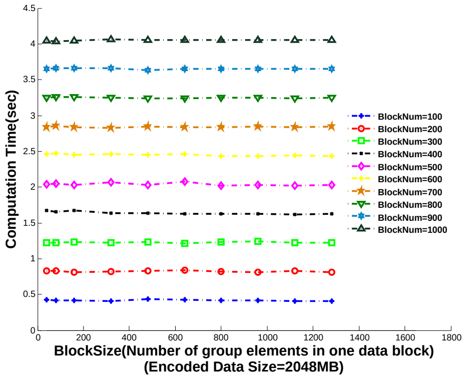




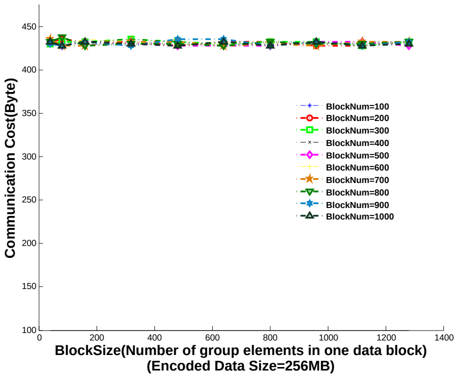


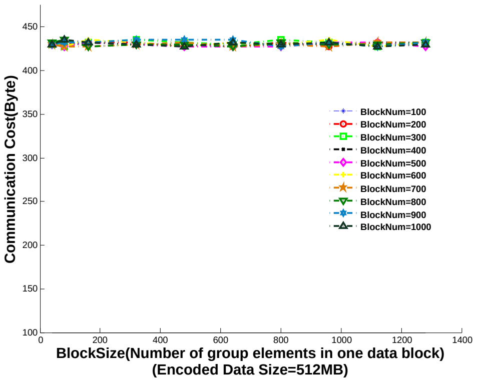
10.2 Full Experiment Results For Verification

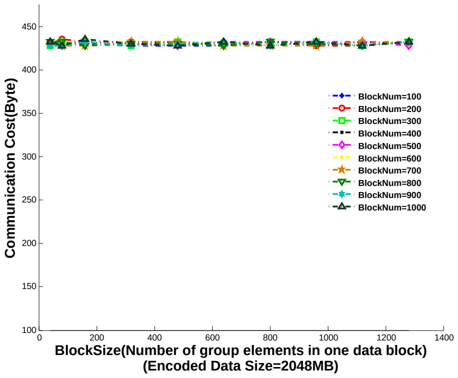




10.3 Full Experiment Results For Communication Cost


Communication Cost(Byte)
450
400
350
300
250
200
150
100
0
200
400
600
800
1000
1200
1400
BlockSize(Number of group elements in one data block)
(Encoded Data Size=256MB)
BlockNum=100
BlockNum=200
BlockNum=300
BlockNum=400
BlockNum=500
BlockNum=600
BlockNum=700
BlockNum=800
BlockNum=900
BlockNum=1000


Communication Cost(Byte)
450
400
350
300
250
200
150
100
0
200
400
600
800
1000
1200
1400
BlockSize(Number of group elements in one data block)
(Encoded Data Size=512MB)
BlockNum=100
BlockNum=200
BlockNum=300
BlockNum=400
BlockNum=500
BlockNum=600
BlockNum=700
BlockNum=800
BlockNum=900
BlockNum=1000


Communication Cost(Byte)
450
400
350
300
250
200
150
100
0
200
400
600
800
1000
1200
1400
BlockSize(Number of group elements in one data block)
(Encoded Data Size=2048MB)
BlockNum=100
BlockNum=200
BlockNum=300
BlockNum=400
BlockNum=500
BlockNum=600
BlockNum=700
BlockNum=800
BlockNum=900
BlockNum=1000