SCIENCE CHINA
Information Sciences

• RESEARCH PAPERS •

August 2011 Vol. 54 No. 8: 1608–1617
doi: 10.1007/s11432-011-4293-9

# Zero-knowledge proofs of retrievability

ZHU Yan[1,2*], WANG HuaiXi[3], HU ZeXing[1], AHN Gail-Joon[4] & HU HongXin[4*]

[1]*Institute of Computer Science and Technology, Peking University, Beijing* 100871*, China;*
[2]*Beijing Key Laboratory of Internet Security Technology, Peking University, Beijing* 100871*, China;*
[3]*School of Mathematical Sciences, Peking University, Beijing* 100871*, China;*
[4]*School of Computing, Informatics, and Decision Systems Engineering, Arizona State University, Tempe, AZ* 85287*, USA*

Received April 26, 2010; accepted December 14, 2010; published online May 31, 2011

**Abstract** Proof of retrievability (POR) is a technique for ensuring the integrity of data in outsourced storage services. In this paper, we address the construction of POR protocol on the standard model of interactive proof systems. We propose the first interactive POR scheme to prevent the fraudulence of prover and the leakage of verified data. We also give full proofs of soundness and zero-knowledge properties by constructing a polynomial-time rewindable knowledge extractor under the computational Diffie-Hellman assumption. In particular, the verification process of this scheme requires a low, constant amount of overhead, which minimizes communication complexity.

**Keywords** cryptography, integrity of outsourced data, proofs of retrievability, interactive protocol, zero-knowledge, soundness, rewindable knowledge extractor

**Citation** Zhu Y, Wang H X, Hu Z X, et al. Zero-knowledge proofs of retrievability. Sci China Inf Sci, 2011, 54: 1608–1617, doi: 10.1007/s11432-011-4293-9

## 1 Introduction

A proof of retrievability (POR) [1] is a cryptographic proof technique for a storage provider to prove that clients' data remain intact. In other words, the clients can fully retrieve their data and have confidence to use the recovered data. This highlights a strong need to seek an effective solution for checking whether their data have been tampered with or deleted without downloading the latest version of data. This technique is important for the storage-outsourced data, especially large files or achieves. For example, with a wide spread of cloud computing, cloud storage service has become a new profit growth point by providing a comparably low-cost, scalable, location-independent platform for managing clients' data. However, if such an important service is vulnerable to malicious attacks, it would bring irretrievable losses to the clients since their data and archives are stored into an uncertain storage pool outside the enterprises. Therefore, it is necessary for cloud service providers to make use of the POR technique to provide a secure management of their storage services.

Since a formal model for the proof of retrievability was introduced by Juels and Kaliski [1], some schemes [2–5] have been proposed in recent years. In these schemes, Shacham and Waters [6] proposed the compact proofs of retrievability (CPOR) schemes, considered as a representative work with a general

*Corresponding authors (email: yan.zhu@pku.edu.cn; hxhu@asu.edu)

 

framework and diverse characters: 1) a file is split into blocks and each block corresponds to a signature tag; 2) a verifier can verify the integrity of file in a random sampling approach, which is of utmost importance for large or huge files; and 3) a homomorphic property is used to aggregate the tags into a constant size response, which minimizes network communication.

Although various adversary models have been proposed to prove the security of POR schemes, these existing schemes do not follow the standard model of interactive proof systems (IPS) [7] so that the security of verification process, especially the soundness of verification, cannot be guaranteed. This means that a prover could deceive a verifier for the forged data through the verification protocol. More importantly, the data confidentiality of outsourced storage cannot be ensured by the public verification processes, in which the verifier can easily gain all verified data by analyzing the responses of public challenges. Hence, it is necessary to construct an efficient POR scheme on standard model of interactive proof systems so as to prevent the prover fraud and protect the data privacy.

**Related works.** To check the availability and integrity of the storage-outsourced data, Juels and Kaliski [1] first presented a proof of retrievability (POR) scheme which largely relies on preprocessing steps the client conducts before sending a file to the server: "sentinel" blocks are randomly inserted to detect corruption; the file is encrypted to hide these sentinels; and error-correcting codes are used to recover data from corruption. Unfortunately, this scheme can only handle a limited number of queries, which have to fix a priori and can only be applied to encrypted files.

Similar to POR, Ateniese et al. [2] proposed a PDP model for ensuring possession of files on untrusted storages and provided a RSA-based scheme for the static case where it achieves $O(1)$ communication costs. They also proposed a publicly verifiable version, which allows anyone, not just the owner, to challenge the server for data possession. This property greatly extends application areas of PDP protocol due to the separation of data owners and users. However, similar to replay attacks, these schemes are insecure in a dynamic scenario because of the dependence on the index of blocks. To solve this problem, Chris Erway et al. [8] introduced two Dynamic PDP schemes with a Hash function tree to realize $O(\log n)$ communication and computational costs for a file consisting of $n$ blocks.

Based on the works of Juels et al. [1] and Ateniese et al. [2], Shacham and Waters [6] proposed a general model based on a data fragmentation idea, called Compact POR (CPOR), which uses homomorphic property to aggregate a proof into $O(1)$ authenticator value and $O(t)$ computation costs for $t$ challenge blocks. In fact, this model, considered to be a general representative for existing schemes, is readily converted to MAC-based, ECC or RSA schemes, which are built from BLS signature [9] and random oracle model, and have the shortest query and response with public verifiability. However, this model was not constructed on interactive proof systems and an adversary can make use of the public verification protocol to gain the storage-outsourced data.

Furthermore, some other POR schemes and models, such as [4, 5, 10], have been recently proposed. Dodis et al. [4] discussed several variants of this problem (such as bounded-use vs. unbounded-use, knowledge soundness vs. information-soundness), and gave nearly optimal POR schemes for each of these variants. Wang et al. [5] presented a dynamic scheme with $O(\log n)$ costs by integrating above CPOR scheme and Merkle Hash Tree (MHT) in DPDP. Bowers et al. [10] proposed a theoretical framework for the design of POR based on Juels-Kaliski and Shacham-Waters works, which supports a fully Byzantine adversary model on the adversarial noisy channel assumption and the error-correction coding methods.

**Contributions.** In this paper, we focus on the construction of POR protocol to prevent the fraudulence of prover and the leakage of verified data. We introduce the first formal definition of interactive proofs of retrievability (IPOR) on the standard model of interactive proof systems. In terms of this definition, we provide a practical zero-knowledge POR (ZK-POR) solution to prevent data leakage in the public verification process. We also prove the soundness and zero-knowledge propertis of this scheme by constructing a polynomial-time knowledge *Extractor*, having rewindable black-box access to the prover, under the computational Diffie-Hellman (CDH) assumption. The performance analysis shows that our commitment/challenge/response protocol transmits a small, constant amount of data, which minimize network communication. Thus, our scheme supports a public remote checking for the large-size private

archive files in widely-distributed storage systems.

**Organization.** The rest of the paper is organized as follows. In section 2, we describe some basic notations, common POR structure, and the attack for existing schemes. In section 3, we define a formal model of IPOR based on interactive proof systems. A practical ZK-POR scheme is proposed for the IPOR model in section 4. We describe the security analysis and performance evaluation of our scheme in section 5 and section 6, respectively. Finally, we conclude this paper in section 7.

## 2 Preliminaries

Let $\mathcal{H} = \{H_k\}$ be a keyed hash family of functions $H_k : \{0,1\}^* \to \{0,1\}^n$ index by $k \in \mathcal{K}$. We say that algorithm $\mathcal{A}$ has advantage $\epsilon$ in breaking the collision-resistance of $\mathcal{H}$ if

$$\Pr[\mathcal{A}(k) = (m_0, m_1) : m_0 \neq m_1, H_k(m_0) = H_k(m_1)] \geqslant \epsilon,$$

where the probability is over the random choice of $k \in \mathcal{K}$ and the random bits of $\mathcal{A}$. This hash function can be obtained from the hash function of BLS signatures [9].

**Definition 1** (Collision-resistant hash)**.** A hash family $\mathcal{H}$ is $(t, \epsilon)$-collision-resistant if no $t$-time adversary has advantage at least $\epsilon$ in breaking the collision-resistance of $\mathcal{H}$.

We set up our systems using bilinear pairings proposed by Boneh and Franklin [11]. Let $\mathbb{G}$ and $\mathbb{G}_T$ be two multiplicative groups using elliptic curve conventions with large prime order $p$. The function $e$ is a computable bilinear map $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ with the following properties: for any $G, H \in \mathbb{G}$ and all $a, b \in \mathbb{Z}_p$, we have 1) bilinearity: $e(G^a, H^b) = e(G, H)^{ab}$; 2) non-degeneracy: $e(G, H) \neq 1$ unless $G$ or $H = 1$; and 3) computability: $e(G, H)$ is efficiently computable.

**Definition 2** (Bilinear map group system)**.** A bilinear map group system is a tuple $\mathbb{S} = \langle p, \mathbb{G}, \mathbb{G}_T, e\rangle$ composed of the objects as described above.

Shacham and Waters [6] proposed a general CPOR model as follows: Given a file $F$, the client splits $F$ into $n$ blocks $(m_1, \ldots, m_n)$ and each block $m_i$ is further split into $s$ sectors $(m_{i,1}, \ldots, m_{i,s}) \in \mathbb{Z}_p^s$ for some sufficiently large $p$. Let $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map, $g$ be a generator of $\mathbb{G}$, and $H : \{0,1\}^* \to \mathbb{G}$ be the BLS hash. The secret key is $sk = x \in_R \mathbb{Z}_p$ and the public key is $pk = (g, v = g^x)$. The client chooses $s$ random $u_1, \ldots, u_s \in_R \mathbb{G}$ as the verification information $t = (Fn, u_1, \ldots, u_s)$, where $Fn$ is the file name. For each $i \in [1, n]$, the tag at the $i$th block is $\sigma_i = (H(Fn||i) \cdot \prod_{j=1}^{s} u_j^{m_{i,j}})^x$. On receiving query $Q = \{(i, v_i)\}_{i \in I}$ for an index set $I$, the server computes and sends back $\sigma' \leftarrow \prod_{(i,v_i)\in Q} \sigma_i^{v_i}$ and $\mu = (\mu_1, \ldots, \mu_s)$, where $\mu_j \leftarrow \sum_{(i,v_i)\in Q} v_i m_{i,j}$. The verification equation is

$$e(\sigma', g) = e\Big(\prod\nolimits_{(i,v_i)\in Q} H(Fn||i)^{v_i} \cdot \prod\nolimits_{j=1}^{s} u_j^{\mu_j}, v\Big).$$

This scheme is not secure for the leakage of file information as follows:

**Attack 1.** An adversary can get the file and tag information by running or wiretapping $n$ times verification communication for a file with $n \times s$ sectors.

*Proof.* Let $s$ be the number of sectors in each block. After running or wiretapping $n$ times queries, an adversary can get $n$ times challenges $(Q^{(1)}, \ldots, Q^{(n)})$ and their the responses $((\sigma'^{(1)}, \mu^{(1)}), \ldots, (\sigma'^{(n)}, \mu^{(n)}))$, where $\mu^{(k)} = (\mu_1^{(k)}, \ldots, \mu_s^{(k)})$ for $k \in [1, n]$. For each $i \in [1, s]$, these responses can generate the equations

$$\begin{cases} \mu_i^{(1)} &= v_1^{(1)} m_{1,i} + \cdots + v_n^{(1)} m_{n,i}, \\ \quad\vdots & \qquad\qquad\vdots \\ \mu_i^{(n)} &= v_1^{(n)} m_{1,i} + \cdots + v_n^{(n)} m_{n,i}, \end{cases}$$

where $Q^{(k)} = \{(j, v_j^{(k)})\}_{j \in I}$ are known and $\forall j \notin I$, $v_j^{(k)} = 0$ for $k \in [1, n]$. The adversary can compute $\{m_{1,i}, \ldots, m_{n,i}\}$ by solving the equations iff the coefficient matrix $\{v_i^{(j)}\}_{n\times n}$ of equations is invertible.

After $s$ times solving these equations ($i \in [1, s]$), the adversary can obtain the whole file, $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$. Similarly, the adversary can get all tags $\sigma_1, \ldots, \sigma_n$ by using $\sigma'^{(1)}, \ldots, \sigma'^{(n)}$. Denote the inverse matrix of $\{v_i^{(j)}\}_{n\times n}$ by $\{w_{i,j}\}_{n\times n}$, all the tags can be easily computed following the equations $\sigma_j = \prod_{i=1}^{n} \sigma'^{(i)^{w_{i,j}}}$ for $j \in [1, n]$ .

## 3 Interactive proofs of retrievability

### 3.1 Definition

We present the definition of interactive proofs of retrievability (IPOR) based on interactive proof systems:

**Defintion 3** (Interactive-POR)**.** An interactive proof of retrievability scheme $\mathcal{S}$ is a collection of two algorithms and an interactive proof system, $\mathcal{S} = (\mathcal{K}, \mathcal{T}, \mathcal{P})$:

$\mathcal{K}eyGen(1^\kappa)$: It takes a security parameter $\kappa$ as input, and returns a secret key $sk$ or a public-secret keypair $(pk, sk)$;

$\mathcal{T}agGen(sk, F)$: It takes as inputs the secret key $sk$ and a file $F$, and returns the triples $(\zeta, \psi, \sigma)$, where $\zeta$ denotes the secret used to generate the verification tags, $\psi$ is the set of public verification parameters $u$ and index information $\chi$, i.e., $\psi = (u, \chi)$; $\sigma$ denotes the set of verification tags;

$\mathcal{P}roof(P, V)$: It is a protocol of proof of retrievability between a prover (P) and a verifier (V). At the end of the protocol run, $V$ returns $\{0|1\}$, where 1 means the file is correctly stored on the server. It includes two cases:

• $\langle P(F, \sigma), V(sk, \zeta)\rangle$ is a private proof, where $P$ takes as input a file $F$ and a set of tags $\sigma$, and $V$ takes as input a secret key $sk$ and a secret of tags $\zeta$;

• $\langle P(F, \sigma), V\rangle(pk, \psi)$ is a public proof, where $P$ takes as input a file $F$ and a set of tags $\sigma$, and a public key $pk$ and a set of public parameters $\psi$ are the common input between $P$ and $V$,

where $P(x)$ denotes the subject $P$ holds the secret $x$ and $\langle P, V\rangle(x)$ denotes both parties $P$ and $V$ share a common data $x$ in a protocol.

This is a more generalized model than existing POR models. Since the verification process can be considered as an interactive protocol, this definition is not limited to the specific steps of verification, including scale, sequence, and the number of moves in protocol, so it can provide greater convenience for the construction of protocol. Further, this paper will only consider the construction of public proof protocol.

### 3.2 Security requirements

According to the standard definition of interactive proof system proposed by Bellare and Goldreich [7], the protocol $\mathcal{P}roof(P, V)$ has two requirements:

**Definition 4** (Security of IPOR)**.** A pair of interactive machines $(P, V)$ is called an available proof of retrievability for a file $F$ if $P$ is a (unbounded) probabilistic algorithm, $V$ is a deterministic polynomial-time algorithm, and the following conditions hold for some polynomial $p_1(\cdot), p_2(\cdot)$, and all $\kappa \in \mathbb{N}$:

• Completeness: For every $\sigma \in TagGen(sk, F)$,

$$\Pr[\langle P(F, \sigma), V\rangle(pk, \psi) = 1] \geqslant 1 - 1/p_1(\kappa); \tag{1}$$

• Soundness: For every $\sigma^* \notin TagGen(sk, F)$, every interactive machine $P^*$,

$$\Pr[\langle P^*(F, \sigma^*), V\rangle(pk, \psi) = 1] \leqslant 1/p_2(\kappa); \tag{2}$$

where $p_1(\cdot)$ and $p_2(\cdot)$ are two polynomials, and $\kappa$ is a security parameter used in $KeyGen(1^\kappa)$.

In this definition, the function $1/p_1(\kappa)$ is called completeness error, and the function $1/p_2(\kappa)$ is called soundness error. For non-triviality, we require $1/p_1(\kappa) + 1/p_2(\kappa) \leqslant 1 - 1/poly(\kappa)$.

The soundness means that it is infeasible to fool the verifier into accepting false statements. The soundness can also be regarded as a stricter notion of unforgeability for the file tags. Thus, the above

definition means that the prover cannot forge the file tags or tamper with the data if soundness property holds.

In order to protect the confidentiality of checked data, we are more concerned about the leakage of private information in the verification process. It is easy to find that data blocks and their tags could be obtained by the verifier in some existing schemes. To solve this problem, we introduce zero-knowledge property into IPOR system, as follows:

**Definition 5** (Zero-knowledge)**.** An interactive proof of retrievability scheme is computational zero knowledge if there exists a probabilistic polynomial-time algorithm $S^*$ (called *Simulator*) such that for every probabilistic polynomial-time algorithm $D$, for every polynomial $p(\cdot)$, and for all sufficiently large $\kappa$, it holds that

$$\left| \begin{array}{c} \Pr[D(pk,\psi,S^*(pk,\psi))=1]- \\ \Pr[D(pk,\psi,\langle P(F,\sigma),V^*\rangle(pk,\psi))=1] \end{array} \right| \leqslant 1/p(\kappa),$$

where $S^*(pk,\psi)$ denotes the output of simulator $S$ on common input $(pk,\psi)$ and $\langle P(F,\sigma),V^*\rangle(pk,\psi)$ denotes the output of interactive protocol between $V^*$ and $P(F,\sigma)$ on common input $(pk,\psi)$. That is, for all $\sigma \in TagGen(sk,F)$, the ensembles $S^*(pk,\psi)$ and $\langle P(F,\sigma),V^*\rangle(pk,\psi)$ are computationally indistinguishable.

Actually, zero-knowledge is a property that captures $P$'s robustness against attempts to gain knowledge by interacting with it. For the POR scheme, we make use of the zero-knowledge property to guarantee the security of data blocks and signature tags.

**Definition 6** (ZK-POR)**.** An IPOR is called zero-knowledge proof of retrievability (ZK-POR) if the completeness, knowledge soundness, and zero-knowledge property hold.

## 4 Construction of zero-knowledge proofs of retrievability

In our construction, the verification protocol has a 3-move structure: commitment, challenge and response. This protocol is similar to Schnorr's $\Sigma$ protocol [12], which is a zero-knowledge proof system. We present our IPOR construction as follows:

**KeyGen($1^\kappa$):** Let $\mathbb{S}=(p,\mathbb{G},\mathbb{G}_T,e)$ be a bilinear map group system with randomly selected generators $g,h\in_R\mathbb{G}$, where $\mathbb{G}$, $\mathbb{G}_T$ are two groups of large prime order $p$, $|p|=O(\kappa)$. Generate a collision-resistant hash function $H_k(\cdot)$ and chooses two random $\alpha,\beta\in_R\mathbb{Z}_p$ and computes $H_1=h^\alpha$ and $H_2=h^\beta\in\mathbb{G}$. Thus, the secret key is $sk=(\alpha,\beta)$ and the public key is $pk=(g,h,H_1,H_2)$.

**TagGen($sk,F$):** Splits the file $F$ into $n\times s$ sectors $F=\{m_{i,j}\}\in\mathbb{Z}_p^{n\times s}$. Chooses $s$ random $\tau_1,\ldots,\tau_s\in\mathbb{Z}_p$ as the secret of this file and computes $u_i=g^{\tau_i}\in\mathbb{G}$ for $i\in[1,s]$ and $\xi^{(1)}=H_\xi(\text{“}Fn\text{”})$, where $\xi=\sum_{i=1}^s\tau_i$ and $Fn$ is the file name. Builds an index table $\chi=\{\chi_i\}_{i=1}^n$ and fills out the item $\chi_i$ in $\chi$ for $i\in[1,n]$, where the index table $\chi=\{\chi_i\}_{i\in[1,n]}$ can be used to support some dynamic data operations, for example, we define $\chi_i=(B_i||V_i||R_i)$ and initially set $\chi_i=(B_i=i,V_i=1,R_i\in_R\{0,1\}^*)$, where $B_i$ is the sequence number of block, $R_i$ is the version number of updates for this block, and $R_i$ is a random integer to avoid collision. Then calculates its tag as

$$\sigma_i\leftarrow(\xi_i^{(2)})^\alpha\cdot g^{\sum_{j=1}^s\tau_j\cdot m_{i,j}\cdot\beta}\in\mathbb{G},$$

where $\xi_i^{(2)}=H_{\xi^{(1)}}(\chi_i)$ and $i\in[1,n]$. Finally, sets $u=(\xi^{(1)},u_1,\ldots,u_s)$ and outputs $\zeta=(\tau_1,\ldots,\tau_s)$, $\psi=(u,\chi)$ to a trusted third part (TTP), and $\sigma=(\sigma_1,\ldots,\sigma_n)$ to a storage service provider (SSP).

**Proof($P,V$):** This is a 3-move protocol between Prover (SSP) and Verifier (client) with the common input $(pk,\psi)$, which is stored in a TTP as follows:

• Commitment ($P\rightarrow V$): $P$ chooses a random $\gamma\in_R\mathbb{Z}_p$ and $s$ integers $\lambda_j\in_R\mathbb{Z}_p$ for $j\in[1,s]$, and sends theirs commitments $C=(H_1',\pi)$ to $V$, where $H_1'=H_1^\gamma$ and $\pi\leftarrow e(\prod_{i=1}^s u_j^{\lambda_j},H_2)\in\mathbb{G}_T$.

• Challenge ($P \leftarrow V$): $V$ chooses a random challenge set $I$ of $t$ indices along with $t$ random coefficients $v_i \in \mathbb{Z}_p^*$, where $t = |I|$. Let $Q = \{(i, v_i)\}_{i\in I}$ be the set of challenge index coefficient pairs. $V$ sends $Q$ to $P$.

• Response ($P \rightarrow V$): $P$ calculates the response $\theta$ and $\mu$ as

$$\sigma' \leftarrow \prod_{(i,v_i)\in Q} \sigma_i^{\gamma\cdot v_i}, \quad \mu_j \leftarrow \lambda_j + \gamma \cdot \sum_{(i,v_i)\in Q} v_i \cdot m_{i,j},$$

where $\mu = \{\mu_j\}_{j\in[1,s]}$. $P$ sends $\theta = (\sigma', \mu)$ to $V$.

**Verification:** Now the verifier $V$ can check whether or not the response was correctly formed by

$$\pi \cdot e(\sigma', h) \stackrel{?}{=} e\Big(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, H_1'\Big) \cdot e\Big(\prod_{j=1}^{s} u_j^{\mu_j}, H_2\Big). \tag{3}$$

In order to prevent the leakage of the stored data and tags in the verification process, the secret data $\{m_{i,j}\}$ are protected by a random $\lambda_j \in \mathbb{Z}_p$ and the tags $\{\sigma_i\}$ are randomized by a $\gamma \in \mathbb{Z}_p$. Moreover, the values $\{\lambda_j\}$ and $\gamma$ are protected by the simple commitment methods, i.e., $H_1^{\gamma}$ and $e(\prod_{i=1}^{s} u_j^{\lambda_j}, H_2)$, to avoid the adversary from gaining them.

## 5 Security proof of construction

Our scheme is an efficient interactive proof system with completeness and soundness properties as follows:

(1) Completeness: for every available tag $\sigma \in TagGen(sk, F)$ and a random challenge $Q = (i, v_i)_{i\in I}$, the completeness of protocol can be elaborated as follows:

$$\begin{aligned}
\pi \cdot e(\sigma', h) &= e(g,h)^{\beta\sum_{j=1}^{s}\tau_j\cdot\lambda_j} \cdot e\Big(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h\Big)^{\alpha\cdot\gamma} \cdot e(g,h)^{\gamma\cdot\beta\sum_{j=1}^{s}(\tau_j\cdot\sum_{(i,v_i)\in Q} v_i\cdot m_{i,j})} \\
&= e(g,h)^{\beta\sum_{j=1}^{s}\tau_j\cdot\lambda_j} \cdot e\Big(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h\Big)^{\alpha\cdot\gamma} \cdot e(g,h)^{\beta\sum_{j=1}^{s}(\tau_j\cdot\mu_j - \tau_j\cdot\lambda_j)} \\
&= e\Big(\prod_{(i,v_i)\in Q} (\xi_i^{(2)})^{v_i}, h^{\alpha\cdot\gamma}\Big) \cdot \prod_{j=1}^{s} e(u_j^{\mu_j}, h^{\beta}).
\end{aligned}$$

There exists a trivial solution when $v_i = 0$ for all $i \in I$. In this case, the above equation could not determine whether the processed file is available, because $\sigma' = 1$, $\mu_j = \lambda_j$, and $\pi_j = u_j^{\mu_j}$. Hence, the completeness of protocol holds

$$\Pr[\langle P(F,\sigma), V\rangle(pk, \psi) = 1] \geqslant 1 - 1/p^t,$$

where $t$ is the number of index coefficient pairs in $Q$. In fact, we require $v_i \in_R \mathbb{Z}_p^*$.

(2) Soundness: For every tag $\sigma^* \notin TagGen(sk, F)$, in order to prove the nonexistence of fraudulent $P^*$, to the contrary, we make use of $P^*$ to construct a knowledge extractor $\mathcal{M}$ [7, 13], which gets the common input $(pk, \psi)$ and rewindable black-box accesses to the prover $P^*$, and then attempts to break the computational Diffie-Hellman (CDH) assumption in $\mathbb{G}$: given $G, G_1 = G^a, G_2 = G^b \in_R \mathbb{G}$, output $G^{ab} \in \mathbb{G}$. We have the following theorem:

**Lemma 1.** Our IPOR scheme has $(t, \epsilon')$ knowledge soundness in random oracle and rewindable knowledge extractor model assuming the $(t, \epsilon)$-computational Diffie-Hellman (CDH) assumption holds in the group $\mathbb{G}$ for $\epsilon' \geqslant \epsilon$.

*Proof.* For some unavailable tags $\{\sigma^*\} \notin TagGen(sk, F)$, we assume that there exists an interactive machine $P^*$ that can pass verification with noticeable probability, that is, there exists a polynomial $p(\cdot)$ and all sufficiently large $\kappa$'s,

$$\Pr[\langle P^*(F, \{\sigma^*\}), V\rangle(pk, \psi) = 1] \geqslant 1/p(\kappa). \tag{4}$$

Using $P^*$, we build a probabilistic algorithm $\mathcal{M}$ (called knowledge Extractor) that breaks the Computational Diffie-Hellman CDH problem in a cyclic group $\mathbb{G} \in \mathbb{S}$ of order $p$. That is, given $G, G_1, G_2 \in_R \mathbb{G}$, output $G^{ab} \in \mathbb{G}$, where $G_1 = G^a$, $G_2 = G^b$. The algorithm $\mathcal{M}$ is constructed by interacting with $P^*$ as follows:

**Setup:** $\mathcal{M}$ chooses a random $r \in_R \mathbb{Z}_p$ and sets $g = G$, $h = G^r$, $H_1 = G_1^r$, $H_2 = G_2^r$ as the public key $pk = (g, h, H_1, H_2)$, which is sent to $P^*$.

**Learning:** given a file $F = \{m_{i,j}\}_{j\in[1,s]}^{i\in[1,n]}$, $\mathcal{M}$ first chooses $s$ random $\tau_i \in_R \mathbb{Z}_p$ and $u_i = G_2^{\tau_i}$ for $i \in [1, s]$. Secondly, $\mathcal{M}$ assigns the indices $1, \ldots, n$ into two sets $T = \{t_1, \ldots, t_{\frac{n}{2}}\}$ and $T' = \{t'_1, \ldots, t'_{\frac{n}{2}}\}$. Let $m_{t_i,j} \neq m_{t'_i,j}$ for all $i \in [1, n/2]$ and $j \in [1, s]$. Then, $\mathcal{M}$ builds an index table $\chi$ and $\xi^{(1)}$ in terms of the original scheme and generates the tag of each block, as follows:

- For each $t_i \in T$, $\mathcal{M}$ chooses $r_i \in_R \mathbb{Z}_p$ and sets $\xi_{t_i}^{(2)} = H_{\xi^{(1)}}(\chi_{t_i}) = G^{r_i}$ and $\sigma_{t_i} = G_1^{r_i} \cdot G_2^{\sum_{j=1}^{s} \tau_j \cdot m_{t_i,j}}$.
- For each $t'_i \in T'$, $\mathcal{M}$ uses $r_i$ and two random $r'_i, \zeta_i \in_R \mathbb{Z}_p$ to sets $\xi_{t'_i}^{(2)} = H_{\xi^{(1)}}(\chi_{t'_i}) = G^{r_i} \cdot G_2^{r'_i}$ and $\sigma_{t'_i} = G_1^{\zeta_i} \cdot G_2^{\sum_{j=1}^{s} \tau_j \cdot m_{t'_i,j}}$.

$\mathcal{M}$ checks whether $e(\sigma_{t'_i}, h) \stackrel{?}{=} e(\xi_{t'_i}^{(2)}, H_1) \cdot e(\prod_{j=1}^{s} u_j^{m_{t'_i,j}}, H_2)$ for all $t'_i \in T'$. If the result is true, then outputs $G^{ab} = G_2^a = (G^{\zeta_i} \cdot G_1^{r_i})^{(r'_i)^{-1}}$, otherwise $\mathcal{M}$ sends $(F, \sigma^* = \{\sigma_i\}_{i=1}^n)$ and $\psi = (\xi^{(1)}, u = \{u_i\}, \chi)$ to $P^*$.

**Hash queries:** At any time, $P^*$ can query the hash function $H_{\xi^{(1)}}(\chi_k)$, $\mathcal{M}$ responds with $\xi_{t_i}^{(2)}$ or $\xi_{t'_i}^{(2)}$ while ensuring consistency, where $k = t_i$ or $t'_i$.

**Output:** $\mathcal{M}$ chooses an index set $I \subset [1, \frac{n}{2}]$ and two subsets $I_1$ and $I_2$, where $I = I_1 \bigcup I_2$, $|I_2| > 0$. $\mathcal{M}$ constructs the challenges $\{v_i\}_{i\in I}$ and all $v_i \neq 0$. Then $\mathcal{M}$ simulates $V$ to run an interaction $\langle P^*, \mathcal{M}\rangle$ as follows:

- Commitment. $\mathcal{M}$ receives $(H'_1, \pi')$ from $P^*$;
- Challenge. $\mathcal{M}$ sends the challenge $Q_1 = \{(t_i, v_i)\}_{i\in I}$ to $P^*$;
- Response. $\mathcal{M}$ receives $(\sigma', \{\mu'_j\}_{j=1}^s)$ from $P^*$.

$\mathcal{M}$ checks whether or not each response is an effective result by eq. (3). If it is true, then $\mathcal{M}$ completes a *rewindable* access to the prover $P^*$ as follows:

- Commitment. $\mathcal{M}$ receives $(H''_1, \pi'')$ from $P^*$;
- Challenge. $\mathcal{M}$ sends the following challenge to $P^*$, $Q_2 = \{(t_i, v_i)\}_{i\in I_1} \bigcup \{(t'_i, v_i)\}_{i\in I_2}$;
- Response. $\mathcal{M}$ receives $(\sigma'', \{\mu''_j\}_{j=1}^s)$ or a special halting-symbol from $P^*$.

If the response is not a halting-symbol, then $\mathcal{M}$ checks whether the response is effective by eq. (3), $H'_1 \stackrel{?}{=} H''_1$, and $\pi' \stackrel{?}{=} \pi''$. If they are true, then $\mathcal{M}$ computes

$$\gamma = \frac{\mu''_j - \mu'_j}{\sum_{i\in I_2} v_i \cdot (m_{t'_i,j} - m_{t_i,j})}$$

for any $j \in [1, s]$ and verifies $H'_1 \stackrel{?}{=} H_1^{\gamma}$ to ensure this is an effective rewindable access. Finally, $\mathcal{M}$ outputs

$$G^{ab} = G_2^a = \left(\sigma'' \cdot \sigma'^{-\phi} \cdot G_1^{\gamma\cdot(\phi-1)\sum_{i\in I} r_i v_i}\right)^{\frac{1}{\gamma \cdot \sum_{i\in I_2} r'_i \cdot v_i}}, \tag{5}$$

where

$$\phi = \frac{\sum_{i\in I_1}\sum_{j=1}^{s} \tau_j m_{t_i,j} v_i + \sum_{i\in I_2}\sum_{j=1}^{s} \tau_j m_{t'_i,j} v_i}{\sum_{i\in I}\sum_{j=1}^{s} \tau_j m_{t_i,j} v_i}$$

and $\psi \neq 1$.

It is obvious that we set $\alpha = a$ and $\beta = b$ in the above construction. Since the tags $\sigma_{t_i}$ are available for any $t_i \in T$, the response in the first interaction satisfies the equation:

$$\pi' \cdot e(\sigma', h) = e\left(\prod_{i\in I} (\xi_{t_i}^{(2)})^{v_i}, H'_1\right) \cdot e\left(\prod_{j=1}^{s} u_j^{\mu'_j}, H_2\right)$$

$$= e(G^{\sum_{i\in I} r_i\cdot v_i}, H_1') \cdot e\Big(\prod\nolimits_{j=1}^{s} u_j^{\mu_j'}, H_2\Big).$$

However, the values of $\{\sigma_{t_i'}\}$ are unavailable for all $t_i' \in T'$. In the second interaction, we require that $\mathcal{M}$ can rewind the prover $P^*$, i.e., the chosen parameters are the same in two protocol executions [7, 13]. In above construction, this property ensures $H_1' = H_1''$, $\pi' = \pi''$, and for all $i \in [1, s]$,

$$\mu_j'' - \mu_j' = \gamma \cdot \sum_{i\in I} v_i \cdot (m_{t_i',j} - m_{t_i,j}) = \gamma \cdot \sum_{i\in I_2} v_i \cdot (m_{t_i',j} - m_{t_i,j}).$$

By checking $H_1' = H_1^{\gamma}$ for all $\gamma$ computed from this equation, we can make sure of the consistence of $\lambda_i' = \lambda_i''$ for $i \in [1, s]$ in two executions. Thus, we have $e(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2) \cdot \pi'^{-1} = e(G_2, H_2)^{\sum_{i\in I}\sum_{j=1}^{s}\tau_j m_{t_i,j} v_i}$ and

$$e\Big(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2\Big) \cdot \pi''^{-1} = e(G_2, H_2)^{\sum_{i\in I_1}\sum_{j=1}^{s}\tau_j m_{t_i,j} v_i} \cdot e(G_2, H_2)^{\sum_{i\in I_2}\sum_{j=1}^{s}\tau_j m_{t_i',j} v_i}.$$

This means that $e(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2) \cdot \pi''^{-1} = (e(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2) \cdot \pi'^{-1})^{\phi}$. In terms of the responses, we have

$$\begin{aligned}
e(\sigma'', h) &= e\Big(\prod_{i\in I_1} (\xi_{t_i}^{(2)})^{v_i} \cdot \prod_{i\in I_2} (\xi_{t_i'}^{(2)})^{v_i}, H_1''\Big) \cdot e\Big(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2\Big) \cdot (\pi'')^{-1} \\
&= e\Big(\prod_{i\in I_1} (G^{r_i})^{v_i} \cdot \prod_{i\in I_2} (G^{r_i} \cdot G_2^{r_i'})^{v_i}, H_1''\Big) \cdot e\Big(\prod_{j=1}^{s} u_j^{\mu_j''}, H_2\Big) \cdot \pi''^{-1} \\
&= e\Big(\prod_{i\in I} G^{r_i\cdot v_i}, H_1'\Big) \cdot e\Big(\prod_{i\in I_2} G_2^{r_i'\cdot v_i}, H_1'\Big) \cdot \Big(e\Big(\prod_{j=1}^{s} u_j^{\mu_j'}, H_2\Big) \cdot \pi'^{-1}\Big)^{\phi} \\
&= e\Big(\prod_{i\in I} G^{r_i\cdot v_i}, H_1'\Big) \cdot e\Big(\prod_{i\in I_2} G_2^{r_i'\cdot v_i}, H_1'\Big) \cdot \Big(e\Big(\sigma', h)^{\phi} \cdot e(\prod_{i\in I} G^{r_i\cdot v_i}, H_1'\Big)^{-\phi}\Big) \\
&= e(\sigma'^{\phi}, h) \cdot e(G_2^{\sum_{i\in I_2} r_i' v_i} \cdot G^{(1-\phi)\sum_{i\in I} r_i v_i}, H_1').
\end{aligned}$$

We have the equations $e(\sigma'' \cdot \sigma'^{-\phi}, h) = e(G_2^{\sum_{i\in I_2} r_i'\cdot v_i} \cdot G^{(1-\phi)\sum_{i\in I} r_i v_i}, H_1')$, $H_1' = h^{a\gamma}$, and $G_1 = G^a$, thus eq. (5) holds. Furthermore, we have

$$\Pr[\mathcal{M}(CDH(G, G^a, G^b)) = G^{ab}] \geqslant \Pr[\langle P^*(F, \{\sigma^*\}), \mathcal{M}\rangle(pk, \psi) = 1] \geqslant 1/p(\kappa).$$

It follows that $\mathcal{M}$ can solve the given $\epsilon$-CDH challenge with advantage at least $\epsilon$, as required. This completes the proof of Theorem.

**Lemma 2.** The verification protocol $Proof(P, V)$ is a computational zero-knowledge system in our IPOR scheme.

*Proof.* For the protocol $Proof(P, V)$, we construct a machine $S^*$, which is called a simulator for the interaction between $V$ and $P$. Given the public key $pk = (g, h, H_1, H_2)$, for a file $F$, a public verification information $\psi = (\xi^{(1)}, u_1, \ldots, u_s, \chi)$, and a index set $I$ ($t = |I|$), the simulator $S^*(pk, \psi)$ executes the following:

1. Chooses a random $\sigma' \in_R \mathbb{G}$ and computes $e(\sigma', h)$.
2. Chooses $t$ random coefficients $\{v_i\}_{i\in I} \in_R \mathbb{Z}_p^t$ and a random $\gamma \in_R \mathbb{Z}_p$ to compute $H_1' \leftarrow H_1^{\gamma}$ and $A_1 \leftarrow e(\prod_{i\in I} H_{\xi^{(1)}}(\chi_i)^{v_i}, H_1')$.
3. Chooses $s$ random $\{\mu_i\} \in_R \mathbb{Z}_p^s$ to $A_2 \leftarrow e(\prod_{j=1}^{s} u_j^{\mu_j}, H_2)$.
4. Calculates $\pi \leftarrow A_1 \cdot A_2 \cdot e(\sigma', h)^{-1}$.
5. Outputs $S^*(pk, \psi) = (C, Q, \theta) = ((H_1', \pi), \{(i, v_i)\}_{i=1}^{t}, (\sigma', \mu))$.

It is obvious that the output of simulator $S^*(pk, \psi)$ is an available verification for eq. (3). Let $\langle P(F, \sigma), V^*\rangle(pk, \psi) = ((\overline{H_1'}, \overline{\pi}), \{(i, \overline{v_i})_{i=1}^{t}\}, (\overline{\sigma'}, \overline{\mu}))$ denote the output of the interactive machine $V^*$ after interacting with the interactive machine $P$ on common input $(pk, \psi)$. In fact, every pair of variables

**Table 1** The storage/communication and computation overheads in our IPOR scheme

| Algorithm | | Computation overheads | Communication overheads |
|---|---|---|---|
| KeyGen | | $2[E]$ | $2l_0$ |
| TagGen | | $(2n+s)[E]$ | $nsl_0 + nl_1$ |
| Protocol | Commitment | $[B] + (s+1)[E]$ | $l_2 + l_T$ |
| | Challenge | | $2tl_0$ |
| | Response | $t[E]$ | $sl_0 + l_1$ |
| | Verification | $3[B] + (t+s)[E]$ | |

is identically distributed in two ensembles, for example, $\overline{H_1'}, \{(i, \overline{v_i})\}$ and $H_1', \{(i, v_i)\}$ are identically distributed due to the fact that the variables $\gamma, \{v_i\} \in_R \mathbb{Z}_p$, as well as $(\overline{\sigma'}, \overline{\mu})$ and $(\sigma', \mu)$ are identically distributed since $\sigma' \in_R \mathbb{G}$, $\lambda_j \in_R \mathbb{Z}_p$ and $u_j \leftarrow \lambda_j + \gamma \sum_{i \in I} v_i \cdot m_{i,j}$ for $i \in [1, s]$. Two variables, $\overline{\pi}$ and $\pi$, are computational indistinguishable because the $\overline{\pi}$ is identically distributed in terms of the random choice of all $\lambda_i$ and the distribution of $\pi$ is decided on the randomized assignment of the above variables.

Hence, two ensembles, $S^*(pk, \psi)$ and $\langle P(F, \sigma), V^* \rangle (pk, \psi)$, are computationally indistinguishable, thus for every probabilistic polynomial-time algorithm $D$, for every polynomial $p(\cdot)$, and for all sufficiently large $\kappa$, it holds that

$$\left| \begin{array}{c} \Pr[D\,(pk, \psi, S^*(pk, \psi)) = 1] - \\ \Pr[D\,(pk, \psi, \langle P(F, \sigma), V^* \rangle (pk, \psi))] = 1 \end{array} \right| \leqslant 1/p(\kappa).$$

The fact that such simulators exist means that $V^*$ does not gain any knowledge from $P$ since the same output could be generated without any access to $P$. That is, the protocol $Proof(P, V)$ is zero-knowledge.

According to Lemmas 1 and 2, we have the following theorem:

**Theorem 1.** Under CDH assumption, our IPOR scheme is a zero-knowledge proof of retrievability in random oracle and rewindable extractor model.

## 6 Performances

We first analyze the computation cost of IPOR scheme. For the sake of clarity, Table 1 presents the results of our analyisis. In this table, we use $[E]$ to denote the computation cost of an exponent operation in $\mathbb{G}$, namely, $g^x$, where $x$ is a positive integer in $\mathbb{Z}_p$ and $g \in \mathbb{G}$ or $\mathbb{G}_T$. We neglect the computation cost of algebraic operations and simple modular arithmetic operations because they run fast enough [14]. The most complex operation is the computation of a bilinear map $e(\cdot, \cdot)$ between two elliptic points (denoted as $[B]$).

Secondly, we analyze the storage and communication costs of our schemes. We define the bilinear pairing taking the form $e : E(\mathbb{F}_{p^m}) \times E(\mathbb{F}_{p^{km}}) \to \mathbb{F}^*_{p^{km}}$ (we give here the definition from [15, 16]), where $p$ is a prime, $m$ is a positive integer, and $k$ is the embedding degree (or security multiplier). In this case, we utilize asymmetric pairing $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ to replace symmetric pairing in original schemes. Without loss of generality, let the security parameter $\kappa$ be 80-bits, we need the elliptic curve domain parameters over $\mathbb{F}_p$ with $|p| = 160$-bits and $m = 1$ in our experiments. This means that the length of integer is $l_0 = 2\kappa$ in $\mathbb{Z}_p$. Similarly, we have $l_1 = 4\kappa$ in $\mathbb{G}_1$, $l_2 = 24\kappa$ in $\mathbb{G}_2$, and $l_T = 24\kappa$ in $\mathbb{G}_T$ for the embedding degree $k = 6$. Based on these definitions, we describe storage or communication cost in Table 1. For a 1M bytes file and $s = 200$, the extra storage of tags is $250 \times 40 = 10$K bytes ($n = 250$) and the commitment and response overheads are $240 + 240 = 480$ bytes and $200 \times 20 + 40 \approx 4$K bytes, respectively. It is obvious that the communication overhead has a constant size in the commitment and response steps of verification protocol. Furthermore, given a file with $sz = n \cdot s$ sectors and the probability $\rho$ of sector corruption, the detection probability of our scheme has $P \geqslant 1 - (1-\rho)^{sz \cdot \omega}$, where $\omega$ denotes the sampling probability in the verification protocol.

## 7 Conclusions

In this paper, we addressed the construction of POR scheme on interactive proof systems. Based on an interactive zero-knowledge proof, we proposed an interactive POR (IPOR) scheme to support soundness property and zero-knowledge property. Our analysis showed that our schemes require a small, constant amount of overhead, which minimizes computation and communication complexity.

**Acknowledgements**

This work was supported by the National Natural Science Foundation of China (Grant No. 61003216), and the US National Science Foundation (Grant Nos. NSF-IIS-0900970, NSF-CNS-0831360). The authors gave thanks to the collaborators at Arizona State University: Dijiang Huang and Stephen S. Yau for discussing the research direction and the method for proofs, also to the intern student, Kainan Liu, at Peking University for verifying the scheme by C++ Language.

## References

1 Juels A, Kaliski-Jr B S. Pors: Proofs of retrievability for large files. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007. Alexandria: ACM, 2007. 584–597
2 Ateniese G, Burns R C, Curtmola R, et al. Provable data possession at untrusted stores. In: Proceedings of the 2007 ACM Conference on Computer and Communications Security, CCS 2007. Alexandria: ACM, 2007. 598–609
3 Bowers K D, Juels A, Oprea A. Proofs of retrievability: Theory and implementation. In: Proceedings of the 2009 ACM Workshop on Cloud Computing Security, CCSW 2009. Chicago: ACM, 2009. 43–54
4 Odis Y, Vadhan S P, Wichs D. Proofs of retrievability via hardness amplification. In: Reingold O, ed. Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009. Lecture Notes in Computer Science, vol. 5444. San Francisco: Springer-Verlag, 2009. 109–127
5 Wang Q, Wang C, Li J, et al. Enabling public verifiability and data dynamics for storage security in cloud computing. In: Proceedings of the 14th European Symposium on Research in Computer Security, ESORICS 2009. Saint-Malo: Springer-Verlag, 2009. 355–370
6 Shacham H, Waters B. Compact proofs of retrievability. In: Advances in Cryptology - ASIACRYPT 2008, 14th International Conference on the Theory and Application of Cryptology and Information Security. Melbourne: Springer-Verlag, 2008. 90–107
7 Goldreich O. Foundations of Cryptography: Basic Tools. Volume Basic Tools. Cambridge: Cambridge University Press, 2001
8 Christopher Erway C, Küpçü A, Papamanthou C, et al. Dynamic provable data possession. In: Proceedings of the 2009 ACM Conference on Computer and Communications Security, CCS 2009. Chicago: ACM, 2009. 213–222
9 Boneh D, Boyen X, Shacham H. Short group signatures. In: Proceedings of CRYPTO 2004, LNCS series. Santa Barbara: Springer-Verlag, 2004. 41–55
10 Bowers K D, Juels A, Oprea A. Hail: A high-availability and integrity layer for cloud storage. In: ACM Conference on Computer and Communications Security, CCS 2009. Chicago: ACM, 2009. 187–198
11 Boneh D, Franklin M. Identity-based encryption from the weil pairing. In: Advances in Cryptology (CRYPTO'2001), vol. 2139 of LNCS. Santa Barbara: Springer-Verlag, 2001. 213–229
12 Schnorr C P. Efficient signature generation by smart cards. J Cryptol, 1991, 4: 161–174
13 Cramer R, Damgård I D, MacKenzie P D. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In: Public Key Cryptography. Melbourne: Springer-Verlag, 2000. 354–373
14 Barreto P S L M, Galbraith S D, O'Eigeartaigh C, et al. Efficient pairing computation on supersingular abelian varieties. Des Codes Cryptogr, 2007, 42: 239–271
15 Beuchat J L, Brisebarre N, Detrey J, et al. Arithmetic operators for pairing-based cryptography. In: Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop. Vienna: Springer-Verlag, 2007. 239–255
16 Hu H G, Hu L, Feng D G. On a class of pseudorandom sequences from elliptic curves over finite fields. IEEE Trans Inf Theory, 2007, 53: 2598–2605