Non-Repudiable Proofs of Storage in Cloud

Hongyuan Wang, Liehuang Zhu, Yijia Lilong, and Chang Xu

School of Computer, Being Institute of Technology,
Beijing, China, 100081
{wanghongyuan, liehuangz, lilongyijia, xuchang }@bit.edu.cn

Abstract. With the widespread use of cloud computing and cloud storage, how to ensure the authenticity of data in remote storage has become a severe problem. Provable data possession (PDP) and Proof of Retrievability (POR) are techniques for a client to verify whether an untrusted server possesses the original data entirely, and many PDP and POR schemes have been proposed to resolve above issue so far. But driven by profits, a malicious client may accuse an honest server and deny the correct verification in many circumstances.
In this paper, we give a method to reform any private verification PDP/POR scheme into a non-repudiable PDP/POR scheme. And then we propose an instantiation, the Non-repudiable PDP (NRPDP) scheme of private verification, which can provide mutual proof to protect both the client and server. Based on homomorphic verifiable tags and commitment function, our solution allows both the client and the server to prove themselves and verify the other side respectively. To achieve better performance, we improve the NRPDP scheme to obtain an E-NRPDP scheme, which can save the storage cost of the server and reduce the I/O costs to raise efficiency.
We prove the security of NRPDP scheme in the random oracle model, and implement a prototype based on our NRPDP scheme. Then we utilize big size as much as 10G to evaluate the performance in a realistic cloud platform. The experiments show our scheme can be executed efficiently as the original PDP/POR scheme and guarantee non-repudiation efficaciously. Our method is universal and practical, which means that it can be applied in any private PDP/POR schemes to guarantee non-repudiation.

Keywords: Non-repudiation, Provable Data Possession, Cloud storage, Commitment function

1 Introduction

Cloud computing is getting increasingly popular as it provides a low-cost, scalable, location-independent service and infrastructure. Cloud storage has become one of the most popular applications of cloud computing, for instance, the well-known Google Drive and Amazon S3. But whether the cloud storage providers (we also call them servers in the following sections) and the clients are all trusted is still a problem.

An honest client gives his data to a cloud storage provider. But the data may be corrupted because of internal or external reasons. Even in normal circumstances, the data may be damaged or lost by administration errors such as the physical implementation of storage evolves, updating system or equipment replacement. But taking reputation and money into consideration, the server doesn't want to take the responsibility. After that, the client cannot know the condition of data in time, and consequently it leads to loss. How to effectively and timely detect the integrity of data has been a popular problem.

Many PDP and POR schemes have been proposed to offer data possession guarantee for cloud storage applications, such as the best-known S-PDP scheme proposed by G. Ateniese et al.[4, 5] and Proofs of Retrievability (POR) defined and proposed by Juels and Kalisky[14, 15]. The previous schemes[21, 20, 7, 12, 31, 30, 19, 18, 11, 26] are all unidirectional proof and verification, which protect a client from a dishonest server. In the meantime, several public verification schemes[5, 13, 18, 25, 24, 23, 22]were proposed to achieve zero-storage on client. Private keys are not needed in a public verification scheme, so that anyone can verify a proof generated by the server, and it doesn't have the problem of repudiation. However, some public verification schemes need more time to generate and verify a proof, and ordinarily require more communication bandwidth, and some schemes[25, 24, 23, 22] based on Trusted Third Party (TTP) are based on stronger assumption. In addition, it may reveal the privacy of client since anyone can compute the proof and verification.

But in another case, driven by interests, a client may deny the valid proof and falsely accuse an honest server of modifying his data secretly. For instance, a cloud storage provider gives the promise that he can provide a low-cost storage which guarantees privacy and integrality, and he would pay a huge compensation to the victims if he breaks his promise, just as the cloud service provided by Beijing New Web, who has paid real compensation to the victims. In that case, some malicious clients who want to get the compensation may accuse that the data stored in the cloud has been corrupted, even though the data is intact. Towards the schemes which are not based on the third party and only can guarantee private verification, the server cannot merely make the client's information public, which becomes an issue.

There already exist many optimistic approaches to keep non-repudiation or resolve disputation in other areas, such as Optimistic Fair Exchange protocols [2, 3] and Non-repudiation protocols [27, 28]. These protocols all require a TTP involve in the execution. When a disputation occurs, the TTP can translate the commutative messages into a public message which can be recognized by all participants. The audit schemes based on third party can be used in public verification, though this approach can reduce the burden of clients, only encrypted files can be applied and the TTP must maintain a long-term information state. In a PDP/POR system, this approach undoubtedly will increase the cost of computation and storage services.

To sum up, we aim to construct a storage system to obtain the following guarantees:

Authenticated and retrievable storage: The client wishes to verify that the data stored on the server is correct and integrated. The server should keep the client's data well-preserved, if the data has been changed, the client could be aware of it timely and easily.
Mutual proof not based on the third party: In the absence of a third party, both the clients and servers must demonstrate by themselves that they are all trusted.
Non-repudiation: If the server keeps the client's data well-preserved and gives the client the correct proof of data possession, the client cannot repudiate it. If the client falsely accuses an honest server, the server could expose some information to generate a proof to overturn the false accusation and prove his innocence.
Efficient and practical scheme: We need a scheme that can achieve above goals without having the client retrieve the data from the server or having the server access the entire file. In the case of big data, we need the scheme perform efficiently, which means that we need to minimize the computing cost, communication cost and local storage cost.

1.1 Contributions

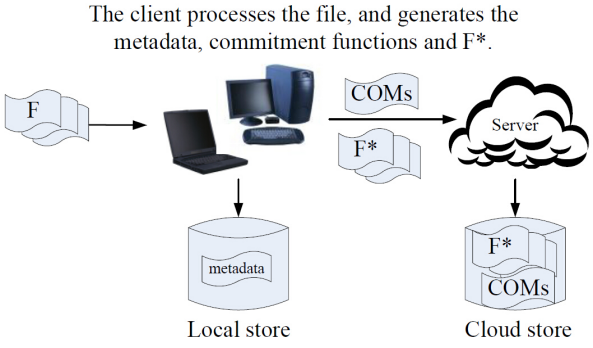
In this paper, we give a method to reform any private verification PDP/POR scheme into a non-repudiable PDP/POR scheme and propose the first NRPDP scheme, which requires mutual proofs between a server and a client to protect both of them. We use RSA scheme and homomorphic verifiable tags to construct the PDP scheme, and the commitment function (COM) to achieve non-repudiation. To achieve better performance, we improve the NRPDP scheme to abtain an E-NRPDP scheme, which can save the storage cost of the server and reduce the I/O costs, and consequently raise efficiency of the whole scheme.
We analyze the security of the NRPDP scheme and implement a prototype to evaluate the performance in a realistic cloud platform. We utilize big data as much as 10G to measure the performance of both PDP schemes and POR schemes .The experiments show that the improved PDP and POR schemes can be executed efficiently and guarantee non-repudiation efficaciously. In addition, we take S-PDP scheme as an instance and test the optimal parameter of NRPDP scheme. From these experiments, we can see that the NRPDP scheme is efficient and practical by the method of commitment functions.

1.2 Paper Organization

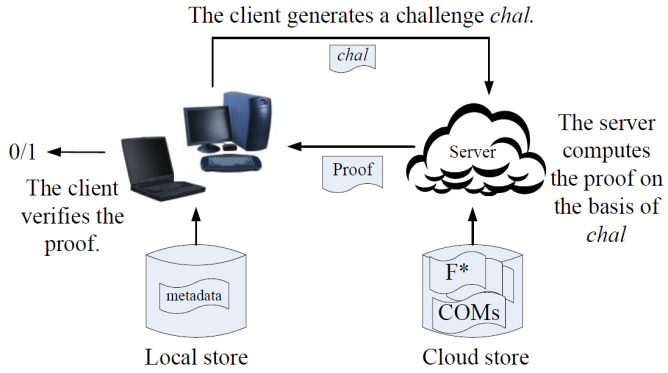
We give an overview of our idea in section 2. In section 3, we describe the preliminaries that will be used in our scheme. Then, we introduce the definition and concrete scheme of NRPDP which is our main construction in section 4, followed by the formal proof of security of our scheme in section 5. Then we support our theoretical claims by implementing and evaluating the NRPDP scheme in section 6. Finally, we roughly view the related work in section 7 and the conclusion in section 8.

2 Overview of Our Idea

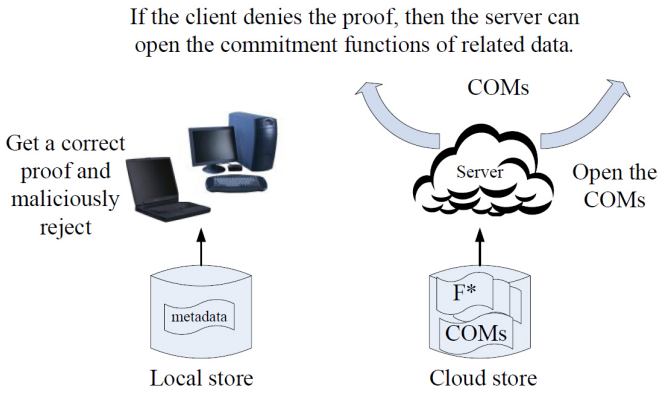
A NRPDP scheme requires mutual proofs between a server and a client so that it can protect both of them. Our construction is based on homomorphic verifiable tags (which construct from RSA scheme) and commitment function, which is a tool to prove the validity of a value at specific times.



(a) Pre-process



(b) Challenge and verification



(c) Public COM verification

Figure 1: the NRPDP scheme.

As Figure 1 shows, in the phase of pre-process, the client preprocesses the file and generates the metadata, commitment functions (COMs) and the processed data. Then the client sends the COMs and data to the server and keeps the metadata locally.
In the phase of challenge, the client gives a challenge *chal* to the server, and the server generates a proof based on the *chal* and sends it to the client. The client verifies whether the proof is valid based on the local metadata.
There are two cases:
Case 1: if the client accepts the proof generated by the server, it means that the server can guarantee the possession of the client's data of the time.
Case 2: if the client does not accept the proof, there are two possibilities, the dishonest client or the incompetent server. On this occasion, the server has two choices, opening the commitment functions or being responsible for dereliction of duty. If the client is dishonest and repudiates the valid proof, the server can open the commitment functions which can prove that the data is well-preserved.
The detailed construction is presented in section 4.

3 PRELIMINARIES

3.1 The Commitment Function

We shall use the Pedersen commitment function [17] as follows.
Let G_q be a group of prime order q in which computing the discrete logarithm function is intractable, and let g and h be elements of G_q where it is hard to compute $\log_g h \bmod q$.
Pedersen commitment function: Let $s \in Z_q$, the commitment $COM(s,t)$ to s, using the help value t which is random selected from Z_q, is $COM(s,t) = g^s h^t \bmod q$.
It has been proved that the committer $COM(s,t)$ reveals no information about s.[17]
The commitment function has two properties: information-theoretically hiding and computationally binding.
Information-theoretically hiding:
Note that with the help value t which is a random choice, the committer $COM(s,t)$ is a random element of G_q. In view of this circumstance, it is hard to distinguish $COM(s,t)$ from a random element of G_q.
Computationally binding:
Since it is hard to compute the value of $\log_g h \bmod q$, it is impossible to find a pair $(s',t') \neq (s,t)$ such that $COM(s',t') \neq COM(s,t)$. Consequently, the committer $COM(s,t)$ can be opened only in one way to demonstrate the original value s.
But Pedersen commitment function is only a one time effort, because that s is revealed when the commitment is opened. Therefore Pedersen function is not suit for a system which is repetitively used.
Based on Pedersen commitment function, we can construct a reusable commitment function as follows.

Let $p_1 = 2p_1' + 1$ and $q_1 = 2q_1' + 1$ be safe primes and let $N_1 = p_1 q_1$ be the modulus of an RSA scheme. Let g_1 be the generator of the unique cyclic subgroup whose order is $p_1' q_1'$.
Let e_1 and d_1 be secret primes such that $e_1 d_1 = 1 \bmod p_1' q_1'$.
Then a construction of reusable commitment function is $COM(M) = (g_1^M h^{H_1(ID)})^{e_1} \bmod N_1, d_1$.
Intuitively, it is a ramification of the Pedersen commitment function [17]. ID is the unique identifier of message M.
This commitment can later be opened only in one way by revealing M and ID, so it is computationally binding.
When the verifier starts a challenge, the committer reveals d_1 which can be viewed as a trapdoor of the commitment to the verifier. Notice that d_1 does not reveal the information of e_1 which keeps the privacy of the commitment, therefore it is information-theoretically hiding.
Then the committer discloses the value of M and ID, the verifier computes whether $s = t$, where $s = g_1^M h^{H_1(ID)} \bmod N_1$ and $t = (COM(M))^{d_1} = (g_1^M h^{H_1(ID)})^{e_1 d_1} \bmod N_1 = g_1^M h^{H_1(ID)} \bmod N_1$.
Notice that above commitment function not only has the properties of information-theoretically hiding and computationally binding, but also has the property of reusability by the reason of privacy of e_1.

3.2 Homomorphic Verifiable Tags (HVTs)

The concept of Homomorphic Verifiable Tags is mentioned by G. Ateniese et al.[5], and we will use them to generate proof of possession in our scheme.
A homomorphism is a map $f : G_1 \to G_2$ such that $f(g \circ g') = f(g) \bullet f(g')$ between two groups G_1, G_2 , where $\circ$ is the operation in G_1 and $\bullet$ is the operation in G_2.
Given a file block m, let T_m be its homomorphic verifiable tag. Given two tags T_{m_i} and T_{m_j} of messages m_i and m_j, anyone can combine them into a tag $T_{m_i+m_j}$, which is corresponding to the sum of messages $m_i + m_j$. These tags will be stored together with the file F on the server, and act as verification metadata for the file blocks. They have the properties consist of unforgeability, homomorphism and blockless verification.
A construction of homomorphic verifiable tags is as follows.
Let g be the generator of a group whose order is N.
Let $T_{m_i} = g^{m_i} \bmod N$ and $T_{m_j} = g^{m_j} \bmod N$.
Then $T_{m_i+m_j} = T_{m_i} \cdot T_{m_j} = g^{m_i} \cdot g^{m_j} \bmod N = g^{m_i+m_j} \bmod N$.
We will utilize this homomorphic property in the following section.

4 MAIN CONSTRUCTION

4.1 Definitions

We begin with the definition of NRPDP as follows.

Definition 1: (Non-repudiation PDP scheme) A NRPDP scheme is a collection of five polynomial-time algorithms (**KeyGen**, **PreGen**, **ProofGen**, **VerifProof**, **VerifCOM**) as follows:

KeyGen$(1^\lambda) \rightarrow (pk, sk)$ is a probabilistic algorithm run by the client to generate the public key and private key. It takes a security parameter λ as input, and returns a pair of keys (pk, sk).

PreGen$(pk, sk, m) \rightarrow (T_m, COM_m)$ is an algorithm run by the client to generate the metadata which used to verify the proof. It takes the public key pk, the private key sk and a file block m as input, and returns the corresponding verification tags T_m and commitment function COM_m.

ProofGen$(pk, F, chal, \Sigma) \rightarrow \rho$ is an algorithm run by the server to generate a proof of possession. The input consists of a public key pk, an ordered collection F of blocks, a challenge $chal$ and an ordered collection Σ of tags which generated in **PreGen** to verify the possession proof. It returns a proof ρ for the blocks that determined by the challenge $chal$.

VerifProof$(pk, F, chal, \rho) \rightarrow \{1, 0\}$ is an algorithm run by client to verify a proof of possession. It takes the public key pk, the private key sk, a challenge $chal$ and a proof of possession ρ as input, and returns whether ρ is a correct proof for the blocks determined by $chal$.

VerifCOM$(\{M\}, \{I\}, c, \{COM\}) \rightarrow \{1, 0\}$ is an algorithm run by anyone to verify that the server exactly possesses the data of client. It takes the indices of the chosen blocks $\{I\}$ and corresponding commitment functions $\{COM\}$ as input, and returns whether the $\{COM\}$ passes validation.

A NRPDP system is constructed from a NRPDP scheme in three phases:
Pre-process: the client C runs **KeyGen** and **PreGen**, and stores (pk, sk)locally. Then C sends pk, F, COM and Σ to the server S to store and deletes F, COM and Σ from its local storage.
Challenge and verification: C generates a challenge $chal$ and sends it to S. Then S runs **ProofGen** and sends the proof ρ to C. Finally, C checks the validity of the proof ρ by running **VerifProof**.
Public COM verification: if C denies that S does possess the data fully intact, S can open the commitment functions to the public, and let anyone verify the possession guarantee of data.
The second phase can be executed many times in order to insure whether S still possesses the file blocks. To prove that the client is dishonest, the third phase can be executed many times as well.

4.2 A construction of NRPDP

Intuitively, we improve the S-PDP scheme (which is proposed by G. Ateniese et al. and is regarded as a notable landmark in this field.) to generate a NRPDP scheme.
We start by introducing some notations used in our schemes. Let $p = 2p' + 1$ and $q = 2q' + 1$ be safe primes and let $N = pq$ be the modulus of an RSA scheme. Let g be the generator of the unique cyclic subgroup QR_N of Z_N^* whose order is $p'q'$.

Analogously, we choose another RSA scheme for the commitment function. Let $p_1 = 2p_1' + 1$ and $q_1 = 2q_1' + 1$ be safe primes and let $N_1 = p_1 q_1$ be the modulus. Let g_1 be the generator of the unique cyclic subgroup whose order is $p_1' q_1'$.
Break a file into n blocks $m_1, ..., m_n$. Then the tag of each file block m_i is T_i, $i \in \{1, 2, ..., n\}$.

NRPDP overview:

In the phase of pre-process, the client computes the tag of each block. Each tag has a parameter i, which is a unique identifier of each block m_i. This binds a tag on a block, and prevents to utilize a tag to generate a proof for a different block which covers same information. To obtain non-repudiation, the client computes commitment function COM_i of each block m_i. The secret key e_1 is used to prevent these $COMs$ from modification of the server. These tags and $COMs$ are stored on the server together with the original file, while the client only stores a small, constant amount of data, which is irrelevant to the file size. In the phase of challenge, the client requests the proof of possession of c blocks, which are randomly chosen using a pseudo-random permutation with a fresh key by server. To ensure that the server cannot reuse values from previous challenges, the client uses the value $g_x = g^x \bmod N$. According to the *chal* sent by client, the server generates and sends to the client a proof ρ which consists two portions: T and u. The value T contains the information of requested tags, while u is obtained by the requested file blocks. When receives the proof, the client can remove some auxiliary parameters such as $H^{a_z}_{k_{hash}}(i_z)$ by using the public and private keys, and then verify the validity of the proof by checking whether a certain relation holds between T and u.
In the phase of COM verification, if the client denies a valid proof generated by an honest server, the server can request the indices of the error blocks (or all the challenged blocks), and reveal the blocks and related $COMs$ to the public. Notice that e_1 is a secret key so that the server cannot modify or forge these $COMs$ at all. Anyone can compute the values s, t based on the public information and check whether the certain relation holds between them.

NRPDP in detail:

Let $\lambda, \partial, \delta$ be security parameters. Let H and H_1 be a full-domain hash function and f be a pseudo-random function, and let π be a pseudo-random permutation:
$H : \{0,1\}^{\lambda} \times \{0,1\}^* \to QR_N$
$H_1 : \{0,1\}^{\log_2(n)} \to \{0,1\}^{\log_2(n)}$
$f : \{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \to \{0,1\}^{\partial}$
$\pi : \{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \to \{0,1\}^{\log_2(n)}$

KeyGen$(1^{\lambda}) \to (pk, sk)$:
Choose a random number $x \xleftarrow{R} Z_p$ and compute $g_x \leftarrow g^x \bmod N$. Let e and d be secret primes such that $ed = 1 \bmod p'q'$, and let e_1 and d_1 be secret primes such that $e_1 d_1 = 1 \bmod p_1' q_1'$.Then choose a hash key $k_{hash} \xleftarrow{R} \kappa_{hash}$.
Then let $sk = (e, e_1, d, x, k_{hash})$ and $pk = (N, N_1, d_1, g, g_1, g_x, h)$.

PreGen$(pk, sk, m_i, i) \to (T_i, COM_i)$:
For each i, $0 < i \leq n$, compute the tag of m_i :

$T_i = (H_{k_{hash}}(i) \cdot g^{m_i})^d \bmod N$.
For each i, $0 < i \leq n$, compute commitment function: $COM_i = COM(m_i) = (g_1^{m_i} h^{H_1(i)})^{e_1} \bmod N_1$ for each block.
Output $(F, \{T_i, COM_i\}_{i=1}^n)$.

ProofGen$(pk, F, chal, \Sigma) \rightarrow \rho$:
The challenge is $chal = (c, k_1, k_2, g_x)$.
For $0 < z \leq c$:
Compute the index of each sampled block: $i_z = \pi_{k_1}(z)$.
Compute the relevant coefficient: $a_z = f_{k_2}(z)$.
Compute $T = T_{i_1}^{a_1} \cdot ... \cdot T_{i_c}^{a_c} =$
$(h_{k_{hash}}^{a_1}(i_1) \cdot ... \cdot h_{k_{hash}}^{a_c}(i_c) \cdot g^{a_1 m_{i_1} + ... + a_c m_{i_c}})^d \bmod N$
Compute $u = (g_x^{\sum_{z=1}^{c} a_z m_{i_z}}) \bmod N$.
Output $\rho = (T, u)$.

VerifProof$(pk, sk, chal, \rho) \rightarrow \{1, 0\}$:
Let $sk = (e, x, k_{hash})$ and $chal = (c, k_1, k_2, g_x)$.
For $0 < z \leq c$:
Compute the index of each sampled block: $i_z = \pi_{k_1}(z)$.
Compute the relevant coefficient: $a_z = f_{k_2}(z)$.
Parse the response of server to obtain u and T.
Compute: $\tau = \frac{T^e}{\prod_{z=1}^{c} H_{k_{hash}}^{a_z}(i_z)} \bmod N$.
If $\tau^x = u$, then output 1. Otherwise output 0.

VerifCOM$(\{m_i\}, \{i\}, c, \{COM_i\}) \rightarrow \{1, 0\}$:
Let $pk = (N_1, d_1, g_1, h)$.
For each $i \in \{i_1, ..., i_c\}$ given in advance (the indices of challenged blocks or appointed blocks by the client), reveal the related commitment functions and file blocks to the public as follows.
$COM_i = COM(m_i) = (g_1^{m_i} h^{H_1(i)})^{e_1} \bmod N_1$, d_1 and m_i.
Then the verifier can check the validity of arbitrary block and its commitment.
For $i \in \{i_1, ..., i_c\}$, the verifier compute:
$s_i = g_1^{m_i} h^{H_1(i)} \bmod N_1$,
$t_i = (COM_i)^{d_1} = (g_1^{m_i} h^{H_1(i)})^{e_1 d_1} \bmod N_1 = g_1^{m_i} h^{H_1(i)} \bmod N_1$.
If $s_i = t_i$, the block m_i and its commitment are matching.
If all the pair of blocks and $COMs$ are completely matching, output 1, otherwise output 0.

Remark:
1. Notice that revealing of d_1 cannot expose e_1 or $COMs$, which means anyone who owns d_1 cannot compute a valid COM of any block m_i ; therefore our scheme can be used infinite number of times theoretically.
2. The index i is a global variable, therefore we can record the unified index in the case of multiple files. For instance, the indices of file blocks are 1 to 100 in file 1, and then in file 2, the indices of blocks are beginning with 101. From the view of a server, all the files constitute one big file for a server to sample and compute; while for a client, there are various files in cloud storage.
3. The phase of COM verification can be applied in any PDP and POR schemes to guarantee the property of non-repudiation, because the security of the com-

mitment scheme is proved independently. To achieve better performance and minimize the storage cost and communication cost, we adjust the parameter of some PDP and POR schemes. The detailed description is in section 6.

4.3 A more efficient scheme of NRPDP

In this chapter, we propose a more efficient variant of NRPDP, which is named E-NRPDP.

Obviously, NRPDP scheme increases storage cost than original scheme for the reason of commitment functions. For instance, if we denote the file block size as 4KB and the size of modulus N as 1024 bits, then each file block of 4KB corresponds with a verification tag of 1024 bits and a commitment function of 1024 bits. Thus the increased storage on the server is about 3.125% of original file size in the original PDP scheme, and 6.25% in the NRPDP scheme. It caused more storage burden on the server and additional cost of the client. On the other hand, the increased storage also influences the I/O costs, and as a consequence, the efficiency becomes blow.

Intuitively, the simplest method of improving efficiency and reducing the extra storage is increasing the size of file block. But this method could increase the computational cost of server because the server must compute $u = (g_x^{\sum_{z=1}^{c} a_z m_{i_z}}) \bmod N$ without awaring of $\phi(N)$. More importantly than that, with the increasing of block size, the granularity of positioning error is becoming larger. In other words, a good PDP scheme should ensure that the client can know which file block has been corrupted or lost. Errors can be positioning timely and losses can be avoided effectually. If the file block is in large size, the client only know that there are errors in this block but can not point out the precise location of errors. This becomes troubles of clients in a real world application.

In conclusion, we need an efficient NRPDP scheme with small granularity.

In E-NRPDP, we combine l small file blocks into a file chunks. In order to distinguish each block, we need l random coefficients. The detailed improvement is as follows:

1. In the phase of **KeyGen**, choose a PRF key $k_v \xleftarrow{R} \kappa_{prf}$.
Then $sk = (e, e_1, d, x, k_v, k_hash)$ and $pk = (N, N_1, d_1, g, g_1, g_x, h)$.

2. In the phase of **PreGen**, notice that each file chunk $m_{<i>}$ has l blocks $\{m_{i,j}\}_{0<j\leq l}$. Then all the file blocks can be presented as a matrix: $\{m_{i,j}\}_{\substack{0<i\leq n \\ 0<j\leq l}}$.
For each j, $0 < j \leq l$, compute the value of random: $v_j = f_{k_v}(j)$.
For each i, $0 < i \leq n$, compute the tag of $\{m_{i,j}\}_{0<j\leq l}$:
$T_i = (H_{k_{hash}}(i) \cdot g^{\sum_{j=1}^{l} v_j m_{i,j}})^d \bmod N$.
For each i, $0 < i \leq n$, compute commitment function $COM_i = COM(m_{<i>}) = (g_1^{m_{<i>}} h^i)^{e_1} \bmod N_1$ for each chunk.
Output $(F, \{T_i, COM_i\}_{i=1}^n)$.

3. In the phase of **ProofGen**, compute $T = T_{i_1}^{a_1} \cdot ... \cdot T_{i_c}^{a_c} =$
$(H_{k_{hash}}^{a_1}(i_1) \cdot ... \cdot H_{k_{hash}}^{a_c}(i_c) \cdot g^{a_1 \sum_{j=1}^{l} v_j m_{i_1,j} + ... + a_c \sum_{j=1}^{l} v_j m_{i_c,j}})^d$
$\bmod N$.

For each $i \in \{i_1, i_2, ..., i_c\}$, compute $u_j = (g_x^{\sum_{z=1}^{c} a_z m_{i_z,j}}) \bmod N$ for $0 < j \leq l$.
Output $\rho = (T, \{u_j\}_{j=1}^{l})$.
4. In the phase of **VerifProof**, for each j, $0 < j \leq l$, compute the value of random: $v_j = f_{k_v}(j)$.
Parse the response of server to obtain $u_1, ..., u_l$ and T.
Compute: $\sigma = \prod_{j=1}^{l} {u_j}^{v_j} \bmod N$.
Compute: $\tau = \frac{T^e}{\prod_{z=1}^{c} H_{k_{hash}}^{a_z}(i_z)} \bmod N$.
5. In the phase of **VerifCOM**, the file block m_i is replaced by file chunk $m_{<i>}$.

Except the improved portion, the rest of the scheme executes as same as the NRPDP scheme.
The E-NRPDP scheme reduces the storage cost of the server, such that it saves the I/O costs and achieves better performance with security guarantee of small granularity. E-NRPDP scheme can give possession guarantees of each block $\{m_{i,j}\}_{\substack{0<i\leq n \\ 0<j\leq l}}$, and pinpoints which block is corrupted efficiently.
The E-NRPDP scheme is a simple deformation of NRPDP scheme, so the security analysis is analogous to NRPDP. In section 6, we will give the detailed security analysis of NRPDP and omit that of E-NRPDP.
Therefore, E-NRPDP is efficient and practical in a real world application.

5 SECURITY

5.1 Security model

We state the security model with two games which capture the data possession property and non-repudiation property respectively.
Data possession game and Commitment game are used to state the security of the NRPDP system and certify the data possession and commitment property respectively. Intuitively, the data commitment game means that the adversary cannot forge or modify a generated commitment function even though it has the corresponding files with related identity of possessor.At the same time,the data possession game means that an adversary can successfully construct a valid proof if and only if it can access all the file blocks corresponding to the challenge.
Game 1: Non-repudiation Game
Setup: The adversary is given input 1^n, and outputs a pair of data m_0, m_1 of the same length.
Challenge: The challenger runs **PreGen**$(pk, sk, m) \rightarrow$
$\{COM\}$ and outputs b_0, b_1 where $\{COM(m_0)\} = b_0$ and $\{COM(m_1)\} = b_1$.
A random bit $r \leftarrow \{1, 0\}$ is chosen, and b_r and pk are given to the adversary.
Decision: The adversary runs **VerifCOM**$(c, \{COM\}) \rightarrow \{1, 0\}$ and output 1 if b_r is the correct commitment of m_1, otherwise output 0.
The commitment function is valid if and only if
$\Pr[PrivK_A^{COM}(n, b_0) = 1] \leq negl(n)$ and
$|\Pr[PrivK_A^{COM}(n, b_1) = 1] - 1| \leq negl(n)$. The probability that the adversary guesses the right chosen is

$\Pr[PrivK_A^{COM}(n, b_r) = r] = \frac{1}{2} \cdot \Pr[PrivK_A^{COM}(n, b_1) = 1] + \frac{1}{2} \cdot \Pr[PrivK_A^{COM}(n, b_0) = 0]$.

Remark: A data commitment game guarantees that a commitment function is computationally binding, which means that the commitment $COM(s, t)$ can be opened only in one way to demonstrate the original value s, which means $|\Pr[PrivK_A^{COM}(n, b_0) = 1] + \Pr[PrivK_A^{COM}(n, b_1) = 0]| \leq negl(n)$, and the probability that the adversary wins the game is negligible.

Game 2: Data Possession Game

Setup: The challenger runs $\mathbf{KeyGen}(1^\lambda) \rightarrow (pk, sk)$, sends pk to the adversary and keeps sk secret.

Query: The adversary queries adaptively: it selects a file block m and sends it to the challenger, and then the challenger runs $\mathbf{PreGen}(pk, sk, m_i) \rightarrow T_i$ to compute the tag and sends it to the adversary. These queries can be carried out many times. Then the adversary stores all the blocks $F = (m_1, ..., m_n)$ together with the tags $\Sigma = (T_1, ..., T_n)$.

Challenge: The challenger generates a challenge *chal* and requests the adversary a proof of possession for the blocks $m_1, ..., m_c$, where $1 \leq c \leq n$.

Forge: The adversary computes a proof ρ determined by *chal* and returns ρ.

If $\mathbf{VerifProof}(pk, sk, chal, \rho) \rightarrow$ "1", then the adversary wins the game.

Remark: A NRPDP system guarantees data possession if for any probabilistic polynomial-time adversary A and arbitrary set of files, the probability that A wins the game is negligibly close to the probability that the challenger can extract those file blocks by an extractor ε.

It means that if the adversary could win the game, then ε could extract the sampled file blocks by executing **GenProof** repeatedly.

5.2 Security analysis

Our scheme is based on the DHK assumption which was introduced by Dent A. W. [10] and derived from the DLP [16]. Analogously, the KEA1-r assumption which is used in many PDP schemes (especially in S-PDP scheme of G. Ateniese et al. [5]) is a ramification of KEA1 introduced by Damgard[9].

DLP (Discrete Logarithm Problem): Let G be a group and $g \in G$, let $\langle g \rangle$ be the cyclic subgroup generated by g. The discrete logarithm problem for the group G can be stated as:

Give $g \in G$ and $a \in \langle g \rangle$, determine if there exists an integer x such that $g^x = a$, and if so, find such an x.

Notice that the formulation of problem is hard to solve.

Afterwards, KEA1 has been shown to hold in generic groups by Dent A. W. [25]. Their assumption, named DHK, is used in our scheme.

DHK (Diffie-Hellman Knowledge): The DHK assumption states that for each polynomial-time attacker A, there exists a polynomial-time extractor A^* that takes the group elements (g, g^x, B, C) and the random coins $R[A]$ used by A as input, and outputs an element $r \in \{1, 2, ..., n\}$ such that $B = g^x$ and $C = g^{xr}$ (if such an r exists).

As previously mentioned, we needn't have to restrict g to be a generator of some specific groups.

Theorem 1: under the RSA and DHK assumptions, the NRPDP scheme guarantees dada possession and non-repudiation in the random oracle model.

Proof:
We respectively prove the non-repudiation property and data possession property by utilizing two games before-mentioned.
1) Proof of non-repudiation
As mentioned earlier, we utilize reduction method to prove non-repudiation of NRPDP.
We assume there exists an adversary A^* that wins the Data Commitment Game of the NRPDP scheme, and then we can construct an adversary A to break the Pedersen commitment function.
Setup:
A^* is given input 1^n, and outputs a pair of data m_0, m_1 of the same length.
Then A^* gives m_0 and m_1 to A.
Challenge:
A runs **PreGen**$(pk, sk, m) \rightarrow \{COM_m\}$ and outputs b_0, b_1 where $\{COM(m_0)\} = b_0$ and $\{COM(m_1)\} = b_1$.
Then A chooses a random bit $r \leftarrow \{1, 0\}$, and gives b_r and pk to A^*.
Forge:
A^* runs **VerifCOM**$(c, \{COM\}) \rightarrow \{1, 0\}$ and outputs 1 or 0.
If the probability $|\Pr[PrivK_A^{COM}(n, b_0) = 1] - 1| \leq negl(n)$ or
$|\Pr[PrivK_A^{COM}(n, b_1) = 0] - 1| \leq negl(n)$, then the adversary wins, in other words, A^* can forge a data m_0 that can commit another data m_1.
By Pedersen commitment function, A can find $m_1, m_2 \in Z_q$ satisfies $m_1 \neq m_2$ and $COM(m_1, ID_1) = COM(m_2, ID_2)$. Obviously, $ID_1 \neq ID_2 \bmod q$ and $\log_{g_1} h = \frac{m_1 - m_2}{ID_1 - ID_2} \bmod N_1$.
By the assumption of DLP, $\log_g h$ cannot be found except with negligible probability in $|N_1|$. Thus, it fulfills the property of non-repudiation.
2) Proof of data possession
We assume there exists an adversary A^* that wins the Data Possession Game of the NRPDP scheme, then we can construct an adversary to break RSA or inter factoring problem by using A^*.Denote the hash function $H(*)$ as random oracle.
Under the DHK assumption, it is easy to reduce the security of NRPDP scheme to the security of the RSA problem (i.e. inter factoring problem) and the discrete logarithm problem.
The reduction of proof is analogous to the security analysis of S-PDP [5], so the detailed process of proof will not be reiterated here.
Notice that the interaction between A and A^* are indistinguishable from interaction between A^* and an honest challenger in the Data Possession Game.

6 IMPLEMENTATION AND EVALUATION

All experiments were conducted on an Intel 2.3GHz Corei7 3610QM system with an 8MB DDR3 1333MHz Cache. Algorithms use the crypto library of OpenSSL version 0.9.8b with a modulus N of size 1024 bits. Experiments measure disk I/O performance with storing files on an hdfs on a Samsung 840 Pro (MZ-7PD128BW) 120GB SSD. All experimental results represent the mean of 20 trials because those results varied little across each attempt.
In essence, NRPDP is an improved version of S-PDP [3], which is regarded as "state-of-the-art" in this field. Therefore, we adopt sampling method as S-PDP, which achieves detecting misbehavior with high probability.
Denote n as the total number of file blocks on the server S, t as the number of the modified or deleted blocks and c as the number of challenged blocks chosen by client C.
Let X be the number of file blocks which are chosen by C and match the file blocks modified or deleted. Then P_X is the probability that at least one modified or deleted file blocks are chosen by C, i.e. the probability that C detects the misbehavior of S.
Then we have that: $P_X = P\{X \geq 1\} = 1 - P\{X = 0\} = 1 - \frac{n-t}{n} \cdot \frac{n-1-t}{n-1} \cdot \frac{n-2-t}{n-2} \cdot \ldots \cdot \frac{n-c+1-t}{n-c+1}$.
Finally, we can get:$1 - \left(\frac{n-t}{n}\right)^c \leq P_X \leq 1 - \left(\frac{n-c+1-t}{n-c+1}\right)^c$.
Data shows that the client can detect the misbehavior of server by asking proof of constant number of blocks with a high probability. Denote $t = 1\%$ of n, and P_X is at least 99%, then the number of challenged blocks is 460.

6.1 The performance of COM generation and verification

In NRPDP scheme, each block is corresponding to a COM. In the phase of pre-process, the client exponentiates data which has been reduced modulo $\phi(N)$ to generate the COM for each file block. The calculation of generating an independent COM is relatively small and has nothing to do with the file block size because the existence of $\phi(N)$.
Figure 2 shows the time of generating $COMs$. As we can see, the time of generation is linearly related to the file size. With the increasing of file size, the number of file blocks and corresponding $COMs$ is increasing, therefore the time of generating $COMs$ is linearly increased. On the other hand, for a same size of file, with the increasing of block size, the number of file blocks is decreased linearly, and consequently the time of generating $COMs$ is reduced.
Though the time of generation is growing linearly related to the file size, it is one-time calculation and reusable.
On the other side, we randomly choose 460 blocks to generate proof in each challenge, and then if the client denies the proof and gives the indices of the error blocks, the server will reveal $COMs$ of these blocks. In experiments, we adopt the worst-case performance to verify $COMs$ each time, this means we measure the time of verification of 460 blocks and $COMs$.

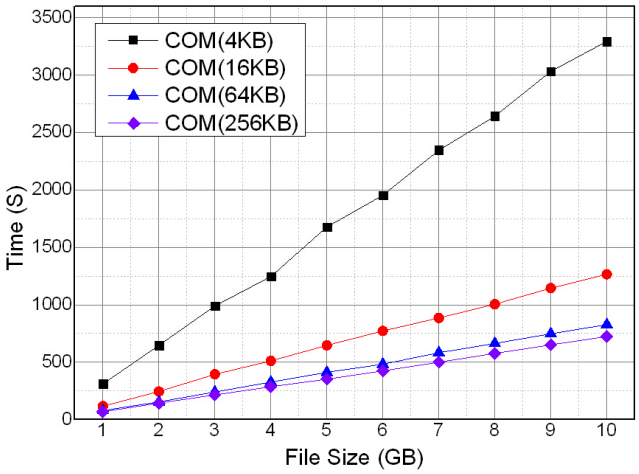


Figure 2: Performance of generating COM.

Figure 3 shows the time of verifying $COMs$. As we can see, the time of verification is almost constant with the same block size.

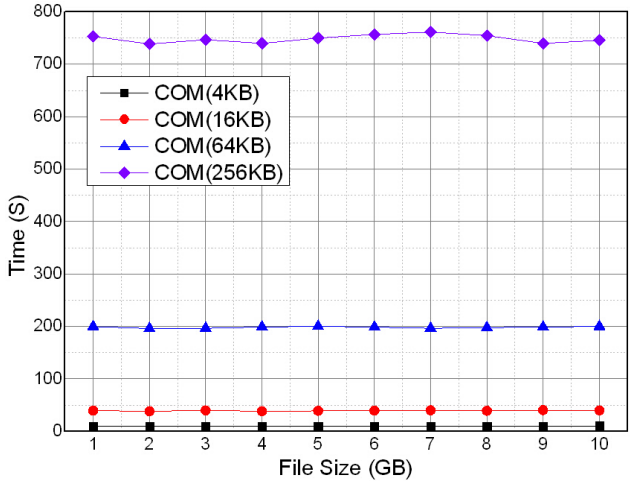


Figure 3: Performance of COM verification.

In the phase of the verification of $COMs$, the server doesn't have $\phi(N)$, and thus it has to exponentiate the whole data block which can be quite time consuming. With the increasing of block size, the augment of calculation is unavoidable, and therefore the time cost is increasing. For the case of same block size, the time of each verification is constant because that the calculation is steadfast (i.e. the calculation of verification of 460 $COMs$) and has no connection with the file size.

On the whole, it is efficient and practical enough for public verification in the case of chosen appropriate parameters.

6.2 The comparation of various PDP and POR schemes

The comparation of the performance of various PDP and POR schemes is as table 1.

Table 1. the comparation of the performance of various PDP and POR schemes

Scheme name	E-/S-PDP[5]	POR[14]	CPOR[18]	CPDP[30]	OPOR[1]	NRPDP	E-NRPDP
Data possession	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Support sampling	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Pre-processing	$O(n)$	$O(n)$	$O(t \log n)$	$O(t \log n)$	$2O(n)$	$O(n)$	$O(t \log n)$
Generating a proof	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$2O(c)$	$O(c)$	$O(c)$
Verifying	$O(1)$	$O(c)$	$O(1)$	$O(1)$	$O(1)$ [1]	$O(1)$	$O(1)$
Communication	$O(1)$	$O(c)$	$O(l)$	$O(l)$	$2O(l)$	$O(1)$	$O(l)$
Client storage	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Mutual proof	No	No	No	No	No	Yes	Yes
Non-repudiation	No	No	No	No	Yes	Yes	Yes

Note: n is the total number of file blocks or chunks, l is the number of blocks in each chunk (i.e. the number of sectors of each block in [30] and [18]), c is the number of sampling blocks or chunks, t is a parameter determined by l.

We transform both a PDP scheme and a POR scheme into non-repudiable schemes, and then measure and compare the performance of the original schemes and the improved schemes.

We choose to improve, implement and evaluate two representative schemes which are S-PDP scheme (which is regarded as a notable landmark in this field.), and CPOR scheme (compact proof of retrievability, which is the first compact and provable secure proof of Retrievability scheme.).

(1) the comparation of PDP schemes

In the phase of pre-process, the client generates the metadata for generating and verifying proof of possession. In this experiment, we measure the time of the pre-process of S-PDP scheme and NRPDP scheme, i.e. the time of generating keys and verification tags.

Notice that we only measure the time of keys and tags generation, which include the time of I/O and storing data to disk, but except transferring data to the server.

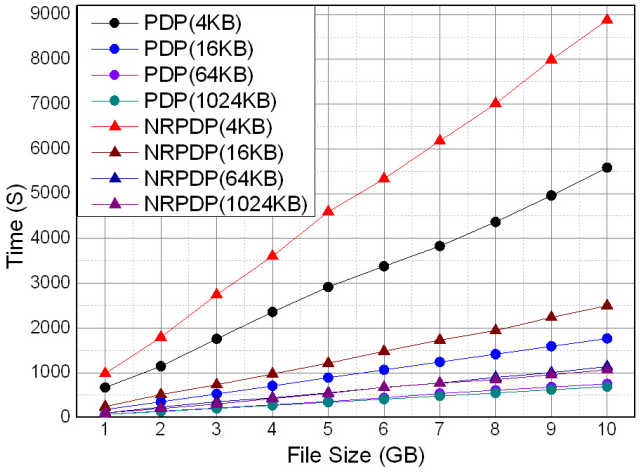
Figure 4 shows the comparation of performance of original PDP scheme and NRPDP scheme.

As shown in Figure 4(a), with the increase of file size, the time of pre-process is growing linearly. That is because both the original PDP and NRPDP performs an exponentiation on each block, which increases with file size.

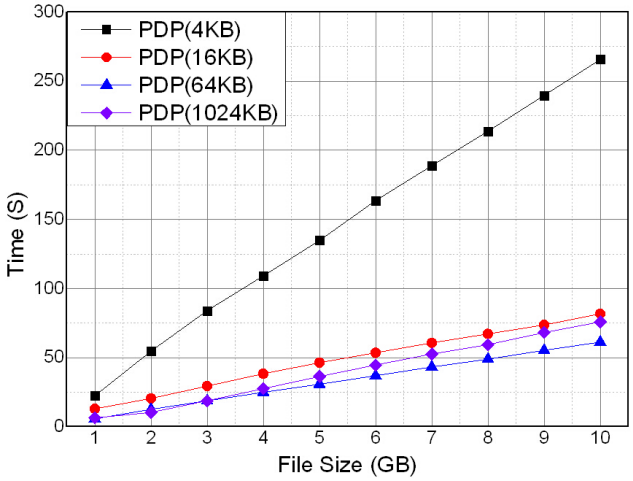
On the other hand, the time cost of pre-process is incremental than the original PDP scheme with same parameters. This is because that generating $COMs$ can

[1] In some particular cases, for instance, the verification cannot be passed or the auditor is dishonest, the complexity value is $2O(c \cdot n)$[1].

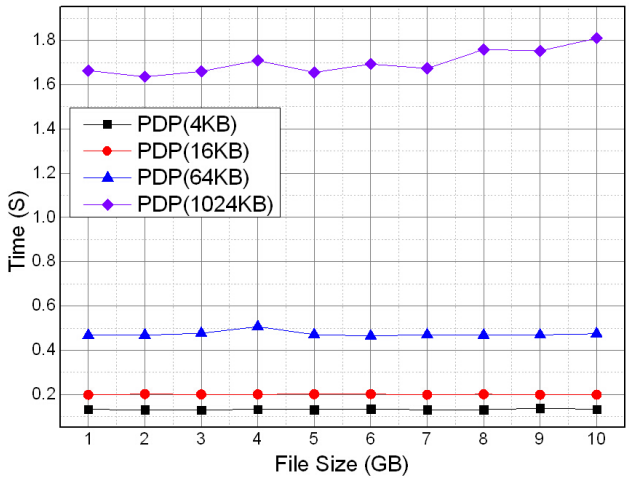
be slightly time consuming. Despite all this, the time cost of generating $COMs$ is smaller than that of generating Tags. And further more, all these computations in the pre-process are one-time calculation and can be reusable infinitely in the theory. Therefore, the improvement of NRPDP scheme is acceptable and practical in a real world application.



(a) The pre-process of original PDP scheme and NRPDP scheme



(b) The challenge and verification of PDP schemes with I/O



(c) The challenge and verification of PDP schemes without I/O

Figure 4: Performance of original PDP scheme and NRPDP scheme.

Contrast with the time of generating verification tags, the time of generating keys is relatively little. In the process of generating tags and $COMs$, the exponent calculation is the most time-consuming. In addition, the client has access to $\phi(N)$, so it can reduce the file blocks modulo $\phi(N)$ before exponent calculation. Furthermore, the security of the scheme depends on the privacy of $\phi(N)$ which is unavailable to the server. Therefore each performance cost comprises a modulus and an exponentiation.

In the phase of challenge and verification, the client generates a challenge *chal* and sends it to the server, and then the server generates a proof of possessing file blocks determined by the *chal*. Then the client verifies the proof of possession given by the server. In this experiment, we compare the time of challenge and verification with a same proof of possession of both two schemes. Exactly speaking, both these PDP schemes have selfsame phase of data possession challenge and verification.

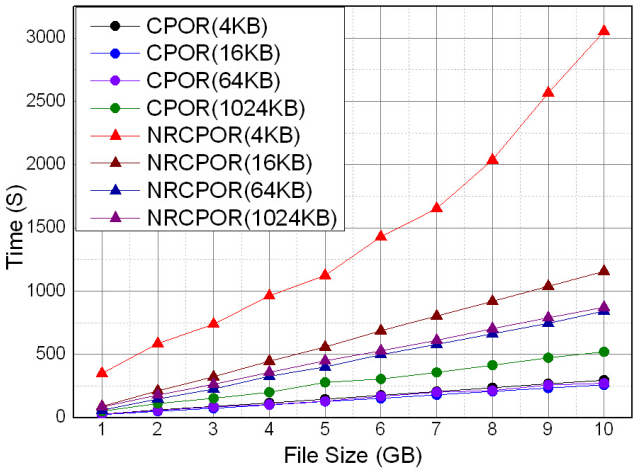
As Figure 4(b) shows, the time cost is linear increased with the increasing of file size. This is due to the increased time of retrieval, read and load data, which can be collectively called I/O cost. Apparently, the I/O cost is increased when the number of file blocks is increased. When we abandon the I/O cost as Figure 4(c), we can see that the time of generating and verifying proof is close to a constant time, this is because that both the original PDP scheme and NRPDP scheme utilize the sampling method which is efficient and secure with high probability.

(2) the comparation of POR schemes

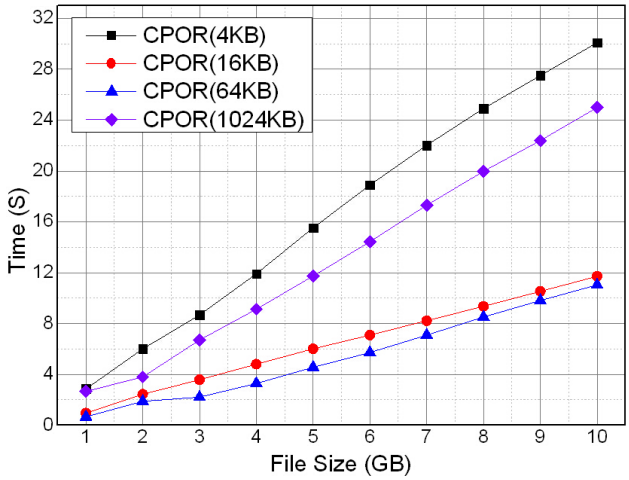
Similar with the PDP scheme, the original POR scheme generates Tags in the phase of pre-process, and generates and verifies a proof in the phase of challenge and verification. Intuitively, the NRPOR scheme combines the original POR scheme and the commitment functions. The performance of both original scheme and improved NRPOR scheme is as figure 5.

In the phase of pre-process of original CPOR scheme, generating Tags is using

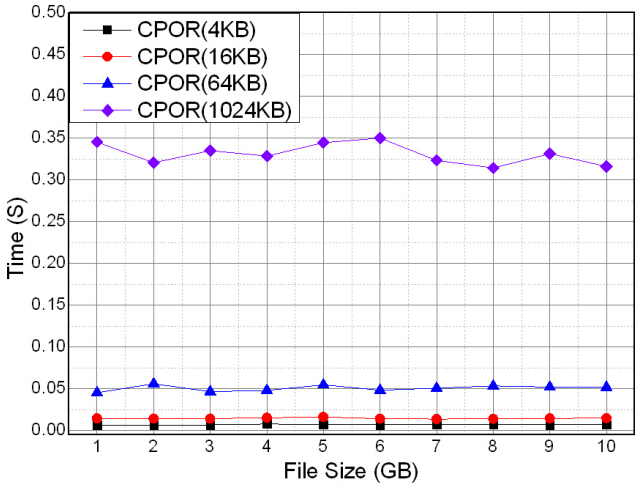
the multiplication and addition operations. Therefore, with the augment of the block size, the time of calculation is becoming longer. But on the other hand, for a same file, the bigger the block size is, the smaller the number of blocks is, and the number of Tags need to compute is smaller, which leads to the smaller I/O costs, as a consequence, the calculation time is smaller. Both of the two cases will trade off an optimal value of block size. Therefore, as we can see from the figure 5 (a), the time cost of different block size is alike, while that of bigger block size or smaller block size is fairly big. We only consider the influence resulted by commitment functions, and don't discuss the optimal value of block size in this chapter.



(a) The pre-process of CPOR scheme and NRCPOR scheme



(b) The challenge and verification of POR schemes with I/O



(c) The challenge and verification of POR schemes without I/O

Figure 5: Performance of pre-process of CPOR scheme and NRCPOR scheme.

The CPOR scheme utilizes multiplication operation and addition operation to generate tags, therefore, the calculation cost can be saved greatly than exponent operation. As a consequence, we can see from figure 5 (a), the time cost of pre-process will increase after adding commitment functions. But the phase of pre-process is one-time calculation and reusable; as a result, the time cost is acceptable.

Analogically, figure 5(b) and 5(c) show the performance of data possession challenge and verification, which has nothing to do with non-repudiation, i.e. commitment functions do not influence the property of data possession or retrievability. In figure 5(b), the time cost is linear increased with the increasing of file size which is due to the I/O cost.And in Figure 4(c), the time of generating and verifying proof is close to a constant time without the I/O cost.

6.3 The storage cost and bandwidth

1) Storage cost:

NRPDP needs more storage cost than S-PDP scheme. In S-PDP, each file block of 4KB corresponds with a verification tag of 1024 bits; this means there are additional 3.125% of the whole data on the server, i.e. the total storage in cloud increased by 3.125%. Though in NRPDP, the increased data capacity is double, but E-NRPDP has settled this problem well. In E-NRPDP, each file chuck of corresponds with a tag of 1024 bits and a commitment function of 1024 bits. Let $l = 9$, and then the total storage in cloud increased by 0.694%.

The storage cost on client is $O(1)$, which is connected with the security parameters.

2) Client-server bandwidth:

Both in NRPDP and E-NRPDP, the required bandwidth is also $O(1)$, this because the *chal* and the proof are all constant (almost 1KB). In the phase of

verifying $COMs$, though the server needs a larger bandwidth (about 1.8MB) to reveal the file chunks and related $COMs$, this rarely happens in a normal cloud storage system. Above all, the bandwidth is entirely reasonable and acceptable to a practical PDP system.

6.4 The parameter optimization of NRPDP

In this chapter, we implement the NRPDP scheme of this paper and compare the performance of them.

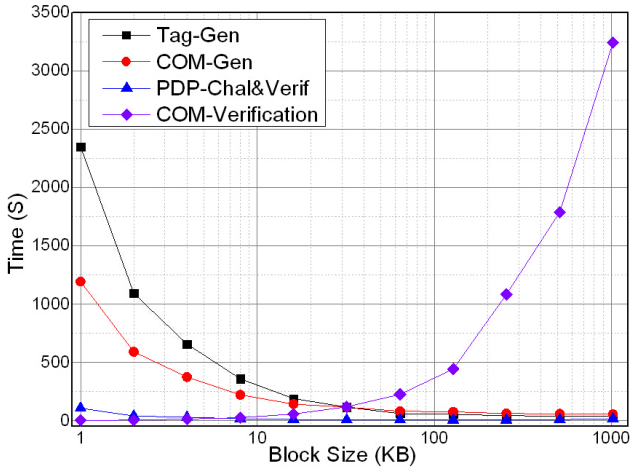


Figure 6: the trade-off of block size with 1GB file

First of all, we should determine the value of the block size. In pre-process, NRPDP exponentiates data which has been reduced modulo $\phi(N)$ to generate both the Tag and the COM, but in the phase of challenge and the verification of $COMs$, the server doesn't have $\phi(N)$ so that it has to exponentiate the whole data block which can be quite time consuming. In addition, when file size is fairly large, we have to take the I/O costs into consideration.

All these reasons cause a natural tradeoff time between pre-process, challenge and COM verification by varying the value of block size.

Figure 6 shows the trade-off caused by block size, and indicates that the best balance occurs at block size of 8-128KB.

Obviously, in the phase of pre-process, with the increasing of block size, the number of file blocks is decreasing, and result in the reduction of time cost (because that the client has the modulo $\phi(N)$). In the meanwhile, the time of COM verification is increasing because that the increased exponent leads to large amount of calculation.

We also can see from the figure, the time of challenge and verification of PDP is fairly small when compared with other phases, and even so, the optimal block size is also between 10KB and 100KB. With the increase of the block size, the consuming time is increasing because that the server doesn't have $\phi(N)$, but at the same time, the I/O costs are decreased for the reduction of the number of file blocks (i.e. for a same file, if the block size is bigger, then the number of file

blocks is smaller). Therefore, the phase of challenge and verification of PDP can be much time consuming when the block size is too big or too small.
Above all, we choose a block size of 32KB to get a better result for NRPDP.
As we can see from figure 7, the time of pre-process is linear increased and as previously mentioned, this phase is one-time calculation and reusable. Therefore, the result is acceptable. On the other hand, the time of both PDP verification and COM verification are constant time cost without I/O cost.
In the actual NRPDP system, the server guarantees the data possession in most instances, which needs fairly little time. Only when the dishonest client denies a valid proof, then server can reveal the $COMs$ and the public will verify these $COMs$. We adopt the worst case, i.e. we reveal and verify 460 blocks and $COMs$ each time. And as we see from figure 7, the time of $COMs$ verification is less than 100 seconds, i.e. the verification of $COMs$ is efficient and practical in real-world applications.

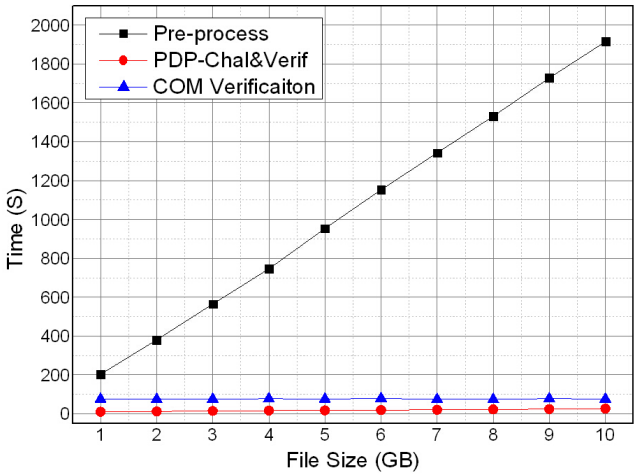


Figure 7: the performance of NRPDP with block size of 32KB

7 RELATED WORK

In 2007, G. Ateniese et al.[5, 4] defined the first sampling model for PDP without having the server retrieving and accessing the entire file. In their schemes, the server provides probabilistic proof with different levels of PDP guarantees. They utilized exponent structure to construct the homomorphic verification tags and used the RSA scheme to keep the tags privacy. After generating a proof, they verified it with the secret key of RSA.
The RSA method has the property of homomorphism, and can be used to construct the detection mechanism of integrity. The simplified algorithm is as fol-

lows:

Pre-process:
1.Choose two large prime p and q, and compute$N = pq$. Generate key pair: $pk = (N, g)$ and $sk = (e, d)$.
2.Let m_i be the file block, and compute Tag: $T_i = (g^{m_i})^d \bmod N$ where g is the generator of QR_N .
3.Send m_i and T_i to the server.
Challenge and Verification:
1.The client gives a challenge to the server.
2.The server chooses file blocks $m_{i_k}(1 \leq k \leq c)$ and Tags T_{i_k}based on the challenge.
3.The server compute $T = \prod_{k=1}^{c} T_{i_k}$ and $\rho = g^{\sum_{k=1}^{c} m_{i_k}} \bmod N$, and sends them to the client.
4.The client computes $\tau = T^e$ and verify whether $\tau = \rho$.

This method is regarded as one of the notable landmarks in this filed, and many subsequent schemes utilize the RSA method, which has the drawback of exponential calculation.

Juels and Kaliski[14, 15]introduced the notion of proof of retrievability (POR) in 2007, which allows a client to retrieve a file that was previously stored on a server. The POR scheme uses sentinels hidden among regular file blocks to detect data modification so that it can only be applied to encrypted files and only perform a limited number of queries, which equal to the number of sentinels. The simplified processes are as follows:

Pre-process:
1.The client encodes the file with error correction code, and then inserts the sentinels into the encoded file.
2.Record the sentinels and send the file to the server.
Challenge and Verification:
1.The client gives the challenge which includes positions of sentinels to the server.
2.The server returns the sentinels of the corresponding position based on the challenge.
3.The client compares the sentinels with local records.

This approach can only perform a limited number of detections because of the finite number of sentinels. Therefore, majority of later POR schemes abandoned this approach.

In 2008, H. Shacham and B. Waters[18] proposed a new notion of Compact POR (CPOR), which utilize symmetric cryptography and homomorphic properties to combine multiple authenticator values into a small one and minimize the communication cost. It utilizes exponential calculation for verification tags so that it increases computation cost. In the meantime, G. Ateniese et al. constructed a scalable and efficient PDP [6] based on symmetric cryptography which supports dynamic operations. But the fatal drawback of this scheme is limited number of verifications. Then In 2009, Y. Dodis et al. [11]formally proved the security of a variant of scheme proposed by Juels and Kaliski, and built the

first unbounded-use POR scheme which doesn't rely on Random Oracles and the first bounded-use scheme with information-theoretic security. It is a theoretical scheme and has not been actually implemented. In the same year, C.C. Erway et al.[12] proposed a dynamic PDP (DPDP) model which is based on the above model to support provable updates to stored data. To meet requirements of both static and dynamic storage. Q. Zheng et al.[26]introduced a notion of fair and dynamic POR (FDPOR), which assures that a server cannot illegally manipulate the data of clients in a dynamic POR scheme. It based on range-based 2-3 trees and an incremental signature scheme. They mentioned the issue of dishonest client, but didn't take effective measures to solve it. B. Chen et al. [8] proposed an efficient RDC model that relies on spot checking. In 2013, Shi E. et al.[19] proposed a practical dynamic POR scheme, which is efficient and uses less bandwidth than the constructions of Stefanov E. et al. [21, 20]and Cash D. et al.[7].

there are also many approaches based on bilinear pairing for public verification which can achieve zero-storage on client. Though the public verification can guarantee the non-repudiation property, The drawback of this methods is that the operation of bilinear pairing is very time-consuming, so that the schemes of public public verification are always not efficient enough. In addition, public verification cannot guarantee privacy because anyone can obtain and verify the data of other clients.

C. Hanser et al. [13] proposed the first simultaneous privately and publicly verifiable PDP protocol based on bilinear pairing and elliptic curve (EC), which uses the same pre-process and metadata to achieve two kinds of verifiability. The drawback is still the extra storage cost and exponential calculation on both server and client. Y. Zhu et al. [31, 30, 29]presented a cooperative PDP (CPDP) scheme based on bilinear pairing, homomorphic verifiable response and hash index hierarchy to support scalability of service and data migration in hybrid cloud.

Another approach for public verification is utilizing TTP to represent the cloud client to verify the possession of data stored in the cloud. It supports the public verification and usually applies on encrypted data. Cong Wang et al. presented some public auditing schemes[25, 24, 23, 22] based on TTP to achieve public verification. Though it could avoid the repudiation issue, it needs the stronger assumption of TTP and usually causes extra cost of client.

Recently, a new notion of Outsourced Proofs of Retrievability (OPOR) [1]is proposed in CCS'2014. The OPOR scheme, which is named Fortress, utilizes an external party to conduct a POR scheme and interact with the server on behalf of the client. Unlike other public verification schemes which are based on TTP, the auditor of Fortress is untrusted. Therefore, the Fortress scheme can protect all these three parties synchronously. But the drawbacks are obviously. On one hand, the Fortress conducts two POR scheme in parallel, so that all the computation costs and communication costs are double. On the other hand, once the verification is unacceptable, the client has to inspect both the auditor and

server, which is very costly. In addition, in the process of verifying auditor, a "forensic" analysis is needed, which is also a strong assumption.

8 Conclusions

In this work, we present the first NRPDP scheme based on homomorphic verifiable tags and commitment function, which focuses on the problem of mutual proof and verification in the circumstance that both the server and client are dishonest. The NRPDP scheme has the properties of non-repudiation and allows both the client and the server prove themselves and verify the other side respectively to protect both of them. In addition, we also propose an efficient NRPDP scheme to achieve better performance.
The performance measurement indicates that our schemes can guarantee data possession with non-repudiation.The method of commitment function we used to achieve non-repudiation is universal and practical, and can be used in many PDP and POR schemes to guarantee the property of non-repudiation. We take S-PDP scheme as an instance and test the optimal parameter of NRPDP scheme. From these experiments, we can see that the NRPDP scheme is efficient and practical by the method of commitment functions.
Under the RSA and DHK assumptions, The NRPDP scheme is secure in the random oracle model. Experiments show that our scheme is efficient and practical, and offers an efficient probabilistic possession guarantee and effective non-repudiation and verification.

9 Acknowledgments

The authors would like to thank Dr. Yuan Zhang and Professor Rui Xue for their comments on improving this paper. The paper is supported National Science Foundation of China under Grant No. 61100172 and No. 61272512, National 863 Plans Projects No. 2013AA01A214, Program for New Century Excellent Talents in University (NCET-12-0046), and Beijing Natural Science Foundation No. 4121001.

References

1. F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter. Outsourced proofs of retrievability. In *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*, pages 831–843. ACM, 2014.
2. N. Asokan, V. Shoup, and M. Waidner. Asynchronous protocols for optimistic fair exchange. In *Security and Privacy, 1998. Proceedings. 1998 IEEE Symposium on*, pages 86–99. IEEE, 1998.
3. N. Asokan, V. Shoup, and M. Waidner. Optimistic fair exchange of digital signatures. In *Advances in CryptologyEUROCRYPT'98*, pages 591–606. Springer, 1998.

4. G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song. Remote data checking using provable data possession. *ACM Transactions on Information and System Security (TISSEC)*, 14(1):12, 2011.
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*, pages 598–609. ACM, 2007.
6. G. Ateniese, R. Di Pietro, L. V. Mancini, and G. Tsudik. Scalable and efficient provable data possession. In *Proceedings of the 4th international conference on Security and privacy in communication netowrks*, page 9. ACM, 2008.
7. D. Cash, A. Küpçü, and D. Wichs. Dynamic proofs of retrievability via oblivious ram. In *Advances in Cryptology–EUROCRYPT 2013*, pages 279–295. Springer, 2013.
8. B. Chen and R. Curtmola. Robust dynamic provable data possession. In *Distributed Computing Systems Workshops (ICDCSW), 2012 32nd International Conference on*, pages 515–525. IEEE, 2012.
9. I. Damgard. Towards practical public key systems secure against chosen ciphertext attacks. In *Advances in Cryptology - Crypto91*, pages 445–456, 1992.
10. A. W. Dent. The hardness of the dhk problem in the generic group model. *IACR Cryptology ePrint Archive*, 2006:156, 2006.
11. Y. Dodis, S. Vadhan, and D. Wichs. Proofs of retrievability via hardness amplification. In *Theory of Cryptography*, pages 109–127. Springer, 2009.
12. 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*, pages 213–222. ACM, 2009.
13. C. Hanser and D. Slamanig. Efficient simultaneous privately and publicly verifiable robust provable data possession from elliptic curves. *IACR Cryptology ePrint Archive*, 2013:392, 2013.
14. 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*, pages 584–597. ACM, 2007.
15. A. Juels, B. S. Kaliski Jr, K. D. Bowers, and A. M. Oprea. Proof of retrievability for archived files, Feb. 19 2013. US Patent 8,381,062.
16. K. S. McCurley. The discrete logarithm problem. In *Proc. of Symp. in Applied Math*, volume 42, pages 49–74, 1990.
17. T. P. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *Advances in CryptologyCRYPTO91*, pages 129–140. Springer, 1992.
18. H. Shacham and B. Waters. Compact proofs of retrievability. In *Advances in Cryptology-ASIACRYPT 2008*, pages 90–107. Springer, 2008.
19. E. Shi, E. Stefanov, and C. Papamanthou. Practical dynamic proofs of retrievability. In *Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security*, pages 325–336. ACM, 2013.
20. E. Stefanov, E. Shi, and D. Song. Towards practical oblivious ram. *arXiv preprint arXiv:1106.3652*, 2011.
21. E. Stefanov, M. van Dijk, A. Juels, and A. Oprea. Iris: A scalable cloud file system with efficient integrity checks. In *Proceedings of the 28th Annual Computer Security Applications Conference*, pages 229–238. ACM, 2012.
22. C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for secure cloud storage. *Computers, IEEE Transactions on*, 62(2):362–375, 2013.

23. C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou. Toward secure and dependable storage services in cloud computing. *Services Computing, IEEE Transactions on*, 5(2):220–232, 2012.
24. C. Wang, Q. Wang, K. Ren, and W. Lou. Privacy-preserving public auditing for data storage security in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9. Ieee, 2010.
25. Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou. Enabling public verifiability and data dynamics for storage security in cloud computing. In *Computer Security–ESORICS 2009*, pages 355–370. Springer, 2009.
26. Q. Zheng and S. Xu. Fair and dynamic proofs of retrievability. In *Proceedings of the first ACM conference on Data and application security and privacy*, pages 237–248. ACM, 2011.
27. J. Zhou and D. Gollman. A fair non-repudiation protocol. In *2012 IEEE Symposium on Security and Privacy*, pages 0055–0055. IEEE Computer Society, 1996.
28. J. Zhou and D. Gollmann. Observations on non-repudiation. In *Advances in CryptologyASIACRYPT'96*, pages 133–144. Springer, 1996.
29. Y. Zhu, G.-J. Ahn, H. Hu, S. S. Yau, H. G. An, and C.-J. Hu. Dynamic audit services for outsourced storages in clouds. *Services Computing, IEEE Transactions on*, 6(2):227–238, 2013.
30. Y. Zhu, H. Hu, G.-J. Ahn, and M. Yu. Cooperative provable data possession for integrity verification in multicloud storage. *Parallel and Distributed Systems, IEEE Transactions on*, 23(12):2231–2244, 2012.
31. Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau. Efficient provable data possession for hybrid clouds. In *Proceedings of the 17th ACM conference on Computer and communications security*, pages 756–758. ACM, 2010.