Secure and Constant Cost Public Cloud Storage Auditing with Deduplication

Jiawei Yuan
Department of Computer Science
University of Arkansas at Little Rock, USA
Email: jxyuan@ualr.edu

Shucheng Yu
Department of Computer Science
University of Arkansas at Little Rock, USA
Email: sxyu1@ualr.edu

***Abstract*—Data integrity and storage efficiency are two important requirements for cloud storage. Proof of Retrievability (POR) and Proof of Data Possession (PDP) techniques assure data integrity for cloud storage. Proof of Ownership (POW) improves storage efficiency by securely removing unnecessarily duplicated data on the storage server. However, trivial combination of the two techniques, in order to achieve both data integrity and storage efficiency, results in non-trivial duplication of metadata (i.e., authentication tags), which contradicts the objectives of POW. Recent attempts to this problem introduce tremendous computational and communication costs and have also been proven not secure. It calls for a new solution to support efficient and secure data integrity auditing with storage deduplication for cloud storage. In this paper we solve this open problem with a novel scheme based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. Our design allows deduplication of both files and their corresponding authentication tags. Data integrity auditing and storage deduplication are achieved simultaneously. Our proposed scheme is also characterized by constant realtime communication and computational cost on the user side. Public auditing and batch auditing are both supported. Hence, our proposed scheme outperforms existing POR and PDP schemes while providing the additional functionality of deduplication. We prove the security of our proposed scheme based on the Computational Diffie-Hellman problem, the Static Diffie-Hellman problem and the t-Strong Diffie-Hellman problem. Numerical analysis and experimental results on Amazon AWS show that our scheme is efficient and scalable.**

I. Introduction

Cloud storage has been increasingly prevalent because of its advantages [1]. Currently, commercial cloud storage services including Microsoft Skydrive, Amazon S3 and Google Cloud Storage have attracted millions of users. Cloud storage stands for not only the massive computing infrastructure but also the economics of scale. Under such a trend, it becomes urgent to assure the quality of data storage services which involves two frequent concerns from both cloud users and cloud service providers: data integrity and storage efficiency. On one hand, with the many data loss and corruption events reported for those best-known cloud service providers [2], [3], data owners, who are also cloud users, have the need to periodically audit the integrity of their outsourced data. On the other hand, for cloud service providers it is necessary to improve the efficiency of cloud storage to take advantage of the economics of scale. According to a recent survey by EMC [4], 75% of today's digital data are duplicated copies. To reduce the unnecessarily redundant copies, the cloud storage servers would deduplicate by keeping only one or few copies for each file and making a link to the file for every user who asks to store the file. Cloud users (i.e., data owners) shall always be able to verify the integrity of the file at any time. For storage efficiency, it is desirable to deduplicate both the file and the metadata (e.g., authentication tags) needed for data integrity check. Taking malicious or misbehavior users or cloud servers into consideration, the cloud server needs to verify that the user actually owns the file before creating a link to this file for him/her; the user also needs to confirm that the cloud actually has the file in its storage and audit the integrity of the file throughout its lifetime.

Related Work. Considering only integrity auditing for data outsourced to cloud servers, a number of POR schemes [5], [6], [7], [8] and PDP schemes [9], [10], [11], [12] have been proposed. Among those ref.[5] has the best performance which achieves public auditing at a constant communication cost. Similar to other POR or PDP schemes, users in ref.[5] still need to perform $O(k)$ multiplication and addition operations over the underlying field, where k is the number of checking data blocks. Batch auditing for multiple requests scenarios is not supported in ref.[5].

For secure storage deduplication, Halevi et al. [13] introduced the first POW scheme based on the Merkle hash tree. Pietro et al. [14] enhanced ref.[13] and proposed a secure POW scheme which reduces the computational cost to a constant number of pseudorandom function operations. Nevertheless, these POW schemes do not consider data integrity auditing.

To achieve both data integrity auditing and storage deduplication, one trivial solution is to directly combine an existing POR/PDP scheme with a POW scheme. This trivial solution, however, will result in a $O(W)$ storage overhead for each file, where W is the number of owners of this file. This is because the data owners, lacking mutual trust, need to separately store their own authentication tags in cloud for file integrity auditing. Since these tags are created for auditing the same file, storing $O(W)$ such copies represents

a type of duplication which contradicts the objective of POW for saving storage cost. For efficient proof of storage with deduplication (POSD), Zheng et al. [15] proposed a scheme aiming at providing both public data integrity auditing and secure storage deduplication. In ref.[15] the communication cost and computational cost on the user side are linear to the number of elements in each data block as well as the number of checking blocks during the integrity auditing process. With an increasing population of mobile users, who access cloud through mobile apps (e.g., iAWS, iCloud, etc.) and have constrained computational resources and bandwidth (e.g., mobile phones with limited data plan), such a communication and computational complexity could represent a barrier to accessing the cloud storage service. Preferably, computational cost and communication cost on the user side shall be constant. Moreover, ref.[15] has been proven not secure [16]. Specifically, by setting the elements in secret keys to some special values, a data owner who outsources data to the cloud server is able to use the server as a malware distribution platform. Therefore, it still calls for a new solution to support efficient and secure data integrity auditing with storage deduplication for cloud storage.

Our Contribution. In this paper, we solve this open problem and propose the first *P*ublic and *C*onstant cost storage integrity *A*uditing scheme with secure *D*eduplication (PCAD) based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. The proposed PCAD scheme is characterized by following salient properties: 1) PCAD is able to securely “deduplicate" the authentication tags by aggregating the tags of the same file from different owners, and hence make the storage overhead independent to the number of owners of the file; 2) the communication cost in our PCAD scheme is made constant thanks to our novel design of polynomial-based authentication tags and secure data aggregation; 3) the computational cost on cloud users is also constant because most computational tasks can be securely offloaded to the cloud server; 4) PCAD supports public auditing, i.e., the data integrity auditing operation can be securely performed by any third party other than the owner(s); 5) PCAD allows batch auditing, i.e., multiple auditing requests can be securely aggregated, which substantially reduces the auditing cost for simultaneous requests; 6) in PCAD data integrity auditing and secure deduplication operations can be performed without the help of existing owners. Noticeably, our PCAD outperforms existing POR and PDP schemes while providing an additional functionality of data deduplication. The main idea of our scheme can be summarized as follows: The data owner outsources the erasure-coded file to the cloud server together with the corresponding authentication tags. To audit the integrity of the outsourced file, a user (who may not be the owner) challenges the cloud with a challenging message. On receiving the message, the cloud generates the proof information based on the public key and sends it to the user. The user verifies the data integrity with the proof information, using our verification algorithm. In order to deduplicate data, when a user wants to upload a data file that already exists in the cloud, the cloud server executes a checking algorithm to see whether or not this user actually possesses the whole file. If the user passes the checking, he/she can directly use the file existed on the server without uploading it again. The security of our proposed scheme is proven under the Computational Diffie-Hellman (CDH) problem, the Static Diffie-Hellman problem and the t-Strong Diffie-Hellman (SDH) problem. Thorough analysis and experimental results on Amazon EC2 Cloud show that our scheme is efficient and scalable. Our main contributions can be summarized as below.

- We propose the first public and constant cost storage integrity auditing scheme with secure deduplication, which can also efficiently handle multiple auditing requests with batch operations.
- We formally prove the security of PCAD. The advantages of PCAD are validated by both numerical analysis and real experiments on Amazon AWS Cloud.
- Our design of polynomial based authentication tag can be used as an independent solution for other related applications, such as verifiable SQL search, encrypted key word search, etc.

The rest of this paper is organized as follows: In Section II, we introduce the models and goals of our scheme. Section III provides the construction and security proof of our scheme; Performance evaluations of our scheme are provided in Section IV; We conclude our paper in Section V.

II. Models and Goals

A. System Model

In this work, we consider a system consisting of four major entities: $Trust\ Authority\ (TA)$, $Data\ Owner$, $Cloud\ Server$ and $User$. The TA in our design is a party only responsible for generating part of the public keys for the system and will go off-line after the key generation. The TA will not participate in any other operations during integrity auditing and deduplication processes. The data owner has a number of data files and stores them on the cloud server together with the authentication tags. Each owner in our design will also generate its own secret keys and public keys for authentication tag generation and data integrity verification. A user to whom the owner shares the data files can access and check the integrity of data files using the public key. A user can also be a $Third\ Party\ Authority\ (TPA)$, who has capabilities/expertise and can periodically audit the integrity of data files being stored on the behalf of data owners. When a user wants to upload data files which are already stored in the cloud, the cloud server just create a link to this file, instead of storing another copy, for this user if the user has been proven a true owner of the file with our scheme. During the integrity auditing and deduplication processes, the user and the cloud server only use the public key and do not need any help from the data owner. While

cloud servers are always equipped with abundant computing resources, data owners and users may have constrained computational power or bandwidth (e.g., mobile phones with limited data plan).

B. Security Model

In our PCAD scheme we consider the following factors that may impact integrity of data stored on cloud servers: 1) attackers corrupting data stored on cloud servers; 2) attackers claiming the ownership of file stored on the cloud even if they do not possess the whole file; 3) hardware/software failures of cloud servers and operational errors of system administrator. The cloud server in our scheme is considered as selfish, which may potentially misbehave in order to save resources (e.g., deleting data stored on it). This assumption is consistent with the previous POSD scheme [15]. We also allow attackers, cloud servers and some data owners to collude with each other in order to pass the integrity verification of valid users.

C. Design Goals

To securely and efficiently verify integrity of the shared data on cloud with deduplication, our PCAD scheme should achieve the following properties at the same time:

- Efficiency: the communication cost and computational cost for users to verify the integrity of data stored on cloud should be constant.
- Functionality: Public data integrity verification and deduplication should be supported at the same time without introducing functionally duplicated authentication tags.
- Correctness: The proposed scheme should accept all valid secret keys and public keys, all valid authentication tags, all valid proof information generated based on valid public keys and all valid data blocks.
- Soundness: Any polynomial-time adversary cannot forge proof information based on modified data and pass the verification algorithm in our scheme; any polynomial-time adversary without the whole data file cannot pass the ownership checking process; any polynomial-time adversary who collude with other data owners and cloud servers cannot forge proof information and pass the integrity verification process.

III. Construction of PCAD

A. Preliminaries and Notation

Bilinear Map: Let G and G_1 be two multiplicative cyclic groups of the same prime order q. A bilinear map is a map that for all $g, h \in G$ and $x, y \overset{R}{\leftarrow} Z_q^*$, $e(g^x, h^y) = e(g, h)^{xy}$. For a bilinear map, there exists a computable algorithm that can compute e efficiently and $e(g, g) \neq 1$.

Notation: Let $H(\cdot)$ be the one-way hash function, G be a multiplicative cyclic group of prime order q, g be the generator of G and $u \overset{R}{\leftarrow} G$. F' is the erasure coded file to be outsourced and is split into n blocks, each of which has s elements: $\{m_{ij}\}, 1 \leq i \leq n, 0 \leq j \leq s-1$. $f_{\vec{a}(x)}$ is denoted as a polynomial with coefficient vector $\vec{a} = (a_0, a_1, \cdots, a_{s-1})$.

B. Our Construction

In this section, we describe the construction of our PCAD scheme as below.

KeyGen: The TA chooses a random number $\alpha \overset{R}{\leftarrow} Z_q^*$ and generates the public keys for the system as $\{g^{\alpha^j}\}_{j=0}^{s+1}$. α is the master key of the system only known to the TA. The TA will go off-line after publishing its public keys. Given a security parameter λ, the data owner generates a signing key-pair $((spk, ssk) \overset{R}{\leftarrow} Sign())$ [17]. The owner also chooses a random number $\epsilon \overset{R}{\leftarrow} Z_q^*$ and computes $\kappa \leftarrow g^{\epsilon}, \nu = g^{\alpha\epsilon}$. Then the public key, secret key and master key are:

$$PK = \{q, \kappa, \nu, spk, u, \{g^{\alpha^j}\}_{j=0}^{s+1}\}$$
$$SK = \{\epsilon, ssk\} \quad MK = \{\alpha\}$$

Setup: To outsource a file F, the data owner first obtains F' by applying erasure code (e.g., Reed-Solomon code [18]), where F' consists of n data blocks and each block has s elements: $\{m_{ij}\}, 1 \leq i \leq n, 0 \leq j \leq s-1$. The owner then randomly chooses a file name $name \in Z_q^*$ and generates the file tag τ under ssk as $\tau \leftarrow name||n||Sign_{ssk}(name||n)$. For each data block $m_i, 1 \leq i \leq n$, the owner produces an authentication tag as:

$$\sigma_i = (u^{H(name||i)} \cdot \prod_{j=0}^{s-1} g^{m_{ij}\alpha^{j+2}})^{\epsilon} = (u^{H(name||i)} \cdot g^{f_{\vec{\beta_i}}(\alpha)})^{\epsilon}$$

where $\vec{\beta_i} = \{0, 0, m_{i,0}, m_{i,1}, \cdots, m_{i,s-1}\}$. The data owner stores F', file tag τ and corresponding authentication tags σ_i on the cloud server.

Challenge: To verify the integrity of F', a user first gets the file tag τ from the cloud server and verifies the signature on τ with ssk. If the signature is not valid, the user rejects and halts; otherwise, the user recovers file name $name$ and n. Then the user randomly chooses a $k-$ elements subset K of $[1, n]$ and two random numbers $r \overset{R}{\leftarrow} Z_q^*$. Finally, the user produces the challenging message $CM = \{K, r\}$ and sends it to the cloud server.

Prove: Based on the challenging message $CM = \{K, r\}$, the cloud server firsts generates $\{p_i = r^i \ mod \ q\}, i \in K$. The cloud then generates $y = f_{\vec{A}}(r)$, where $\vec{A} = \{0, 0, \sum_{i \in K} p_i m_{i,0}, \cdots, \sum_{i \in K} p_i m_{i,s-1}\}$. As polynomials $f(x) \in Z[x]$ have the algebraic property that $(x-r)$ perfectly divides the polynomial $f(x) - f(r), r \overset{R}{\leftarrow} Z_p^*$. The server divides the polynomial $f_{\vec{A}}(x) - f_{\vec{A}}(r)$ with $(x - r)$ using polynomial long division, and denotes the coefficients vector of the resulting quotient polynomial as $\vec{w} = (w_0, w_1, \cdots, w_{s+1})$, that is, $f_{\vec{w}}(x) \equiv \frac{f_{\vec{A}}(x) - f_{\vec{A}}(r)}{x - r}$. The

cloud server generates

$$\psi = \prod_{j=2}^{s+1} (g^{\alpha^j})^{w_j} = g^{f_{\vec{w}}(\alpha)}$$

The cloud server finally computes $\sigma = \prod_{i \in K} \sigma_i^{p_i}$ and sends the proof information $Prf = \{\sigma, \psi, y\}$ to the user.

Verify: On receiving the Prf, the user first computes $\vartheta = \sum_{i \in K} p_i H(name||i)$ and $\eta = u^{\vartheta}$. Based on η, the user verifies the integrity of F' together with $Prf = \{\sigma, \psi, y\}$ as:

$$e(\eta, \kappa) \cdot e(\psi, \nu \cdot \kappa^{-r}) \stackrel{?}{=} e(\sigma, g) \cdot e(\kappa^{-y}, g) \quad (1)$$

If Eq.1 holds, then the user outputs $AuditRst$ as accept; otherwise, outputs $AuditRst$ as reject.

Deduplication: In the $Deduplication$ algorithm, a user claims that he has a file F' and wants to store it on the cloud server, where F' is an existing file on the server. To check whether or not the user actually owns the whole F', the cloud server randomly chooses a $d-$ elements subset D of $[1, n]$ (we discuss the size of D in Section III-E) and sends D to the user. On receiving the set D, the user responds with the corresponding data block $m_i, i \in D$. The cloud server computes

$$\sigma' = \prod_{i \in D} \sigma_i \quad \eta' = \prod_{i \in D} u^{H(name||i)}$$
$$\psi' = e(\prod_{j=2}^{s+1} (g^{\alpha^j})^{B_j}, \kappa) = e(g^{f_{\vec{B}}(\alpha)}, \kappa)$$

where $\vec{B} = (0, 0, \sum_{i \in D} m_{i,0}, \cdots, \sum_{i \in D} m_{i,s-1})$. Then the cloud server checks the integrity of uploaded data blocks $m_i, i \in D$ as:

$$e(\eta', \kappa) \cdot \psi' \stackrel{?}{=} e(\sigma', g) \quad (2)$$

If Eq.2 holds, the cloud server trusts that the user has the whole F'.

Correctness: We analyze the correctness of our construction based on Eq.1 and Eq.2 as:
Eq.1:

$$\begin{aligned} & e(\eta, \kappa) \cdot e(\psi, \nu \cdot \kappa^{-r}) \quad (3) \\ &= e(u, g)^{\epsilon(\sum_{i \in K} p_i H(name||i))} \cdot e(g^{f_{\vec{w}}(\alpha)}, g^{\epsilon(\alpha - r)}) \\ &= e(u, g)^{\epsilon(\sum_{i \in K} p_i H(name||i))} \cdot e(g, g)^{\frac{f_{\vec{A}}(\alpha) - f_{\vec{A}}(r)}{\alpha - r} \cdot \epsilon(\alpha - r)} \\ &= e(u^{\epsilon(\sum_{i \in K} p_i H(name||i))} \cdot g^{\epsilon f_{\vec{A}}(\alpha)}, g) \cdot e(\kappa^{-y}, g) \\ &= e(\sigma, g) \cdot e(\kappa^{-y}, g) \end{aligned}$$

Eq.2

$$\begin{aligned} & e(\sigma', g) \quad (4) \\ &= e(u^{\epsilon(\sum_{i \in D} H(name||i))} \cdot g^{\epsilon f_{\vec{B}}(\alpha)}, g) \\ &= e(u, g)^{\epsilon(\sum_{i \in D} H(name||i))} \cdot e(g, g)^{\epsilon f_{\vec{B}}(\alpha)} \\ &= e(\eta', \kappa) \cdot \psi' \end{aligned}$$

The correctness of our scheme is obvious by Eq.3 and Eq.4.

C. Auditing After Deduplication

In this section, we describe the auditing of file owned by multiple owners after the deduplication process and show how to aggregate authentication tags for the same file.

After the deduplication process for F' on the cloud server, the user who passes ownership checking also becomes a new owner of F'. We define this new owner as $owner_w$, where $1 \leq w \leq W$, $owner_0$ as the original owner who uploads F' and W is the total number of $owner_w$. Since owners have no mutual trust in each other, they need to assure the integrity of F' separately. Specifically, after deduplication, a new owner $owner_w, w \neq 0$ runs the $KeyGen$ algorithm and generates the public key and the private key as:

$$PK_w = \{q, \kappa_w, \nu_w, spk_w, u, \{g^{\alpha^j}\}_{j=0}^{s+1}\}$$
$$SK_w = \{\epsilon_w, ssk_w\}$$

where $\kappa_w = g^{\epsilon_w}, \nu_w = g^{\epsilon_w \alpha}$, $\epsilon_w \stackrel{R}{\leftarrow} Z_q^*$. Then, by running the $Setup$ algorithm, the $owner_w$ generates the file tag τ_w as $\tau_w \leftarrow name||n||Sign_{ssk_w}(name||n)$ and authentication tags for each block m_i in F' as

$$\begin{aligned} \sigma_{wi} &= (u^{H(name||i)} \cdot \prod_{j=0}^{s-1} g^{m_{ij} \alpha^{j+2}})^{\epsilon_w} \quad (5) \\ &= (u^{H(name||i)} \cdot g^{f_{\vec{\beta}_i}(\alpha)})^{\epsilon_w} \end{aligned}$$

where $\vec{\beta}_i = \{0, 0, m_{i,0}, m_{i,1}, \cdots, m_{i,s-1}\}$. The file tag τ_w and corresponding authentication tags σ_{wi} are outsourced to the cloud server.

Instead of storing all the tags from different $owner_w$ for the same F' separately, the cloud server aggregates tags for each data block as:

$$\sigma_i = \prod_{w=0}^{W} \sigma_{wi} = u^{H(name||i) \sum_{w=0}^{W} \epsilon_w} \cdot g^{f_{\vec{\beta}_i}(\alpha) \cdot \sum_{w=0}^{W} \epsilon_w}$$

When a user helps an owner, say $owner_t, t \in [0, W]$, to audit the integrity of F', it runs the $Challenge$ algorithm to generate the challenging message $CM = \{K, r\}$ and sends it to the cloud server. The server runs the $Prove$ algorithm to generate $\sigma = \prod_{i \in K} \sigma_i$,$\psi$ and y. After that, the server outputs $\kappa' = \prod \kappa_w, w \in W/t$ and $\nu' = \prod \nu_w, w \in W/t$. The proof information is $Prf = \{\sigma, \psi, y, \kappa', \nu'\}$. To verify the integrity of F', the user runs the $Verify$ algorithm and checks

$$e(\eta, \kappa) \cdot e(\psi, \nu' \cdot \nu_t \cdot \kappa^{-r}) \stackrel{?}{=} e(\sigma, g) \cdot e(\kappa^{-y}, g) \quad (6)$$

Where $\kappa = \kappa' \cdot \kappa_t$ If Eq.6 holds, then the user outputs $AuditRst$ as accept; otherwise, outputs $AuditRst$ as reject.

Correctness: The correctness of our auditing construction after deduplication can be easily verified by expending Eq.6 similar as the correctness of Eq.1, due to the space limitation, we will not give details here.

D. Batch Auditing

As the TPA has expertise and capabilities that many data owners not have, it can help owners to audit the integrity of their stored files on the cloud periodically. However, when multiple owners delegate their integrity auditing requests at the same time, it is inefficient for the TPA to process these requests one by one. Specifically, given L integrity auditing requests for L different encoded files $F_l' = \{m_{li,j}\}, 1 \leq l \leq L, 1 \leq i \leq n_l, 0 \leq j \leq s_l - 1$ from T different owners (some files may from the same owner), it is desirable for the TPA to handle these requests in batch to reduce both communication cost and computational cost, where n_l is the number of data blocks in encoded file F_l' and s_l ($s_l < s$) is the number of elements in each data block. For this purpose, we design the batch auditing algorithm based on our single request construction as below.

T data owners run $KeyGen$ algorithm separately. The public keys and private keys are

$$PK_t = \{q, \kappa_t, spk_t, u, \{g^{\alpha^j}\}_{j=0}^{s+1}\}$$
$$SK_t = \{\epsilon_t, ssk_t\}, 1 \leq t \leq T$$

where $\kappa_t = g^{\epsilon_t}, \nu_t = g^{\epsilon_t \alpha}$, $\epsilon_t \overset{R}{\leftarrow} Z_q^*$. To audit the integrity of these L files, the TPA runs $Challenge$ algorithm and sends the challenging message $CM = \{r, K\}$ to the cloud server.

On receiving the CM, the cloud server first runs $Prove$ algorithm for L files and generates $Prf_l = \{\psi_l, \sigma_l, y_l\}, 1 \leq l \leq L$ and aggregates the files from same data owner as

$$\psi_t = \prod \psi_{tl} = g^{f_{\vec{w_{tl}}}(\alpha)} \quad y_t = \sum y_{tl}$$

The authentication tags of all files will be aggregated as $\sigma = \prod_{l=1}^{L} \sigma_l$, where $\sigma_l = \prod_{i \in K} \sigma_{li}$. The final proof information $Prf = \{\{\psi_t\}, \sigma, \{y_t\}\}$ is sent to the TPA.

Based on the received Prf, the TPA first runs $Verify$ to get $\eta_t = u^{\sum_{l \in t} \sum_{i \in K} p_i H(name_l || tli)}$ for each files from the same owner. Then the TPA checks the integrity of these L files together as

$$\prod_{t=1}^{T} (e(\eta_t, \kappa_t) \cdot e(\psi_t, \nu_t \cdot \kappa_t^{-r})) \overset{?}{=} e(\sigma, g) \prod_{t=1}^{T} e(\kappa_t^{-y_t}, g) \quad (7)$$

If Eq.7 holds, the TPA outputs $AuditRst$ as accept; otherwise, outputs $AuditRst$ as reject. *Correctness:*

The correctness of our batch auditing construction can be easily verified by expending Eq.7 similar as the correctness of Eq.1, due to the space limitation, we will not give details here.

E. Discussion

In this section, we discuss the error detection probability of our PCAD scheme, the selection of set K in our $Challenge$ algorithm and the selection of set D in our $Deduplication$ algorithm. As we mentioned in our $Setup$ algorithm, we adopt Reed-Solomon code to encode the outsourcing file. For an ℓ Reed-Solomon encoded file ($0 < \ell < 1$), the original file can be recovered from any ℓ fraction of encoded data blocks. Thus, if an ℓ Reed-Solomon encoded file cannot be recovered, the probability of getting an uncorrupted encoded data block will be less than ℓ. In this case, when a user randomly chooses k independently encoded data blocks to challenge, the probability that all these blocks are uncorrupted is less than ℓ^k. When we set $\ell = 0.98$ as previous POR schemes [5], [19] do, the user can achieve at least 99.999% error detection probability when he challenges 600 data blocks for an encoded file.

With regard to the $Deduplication$ algorithm, the cloud server can choose a small set D to check whether the user actually owns the file for storage. Specifically, suppose the user missing 1% blocks of the file F', as proved in [9], the cloud server can have 99% or 95% confidence to detect that the user does not owning the whole file F' only by challenging 460 blocks or 300 blocks (total blocks in F' is larger than 460).

F. Security Proof

In this section, we give the assumptions used in our scheme and then prove the security of the scheme based on these assumptions.

Definition III.1. ***Computational Diffie-Hellman (CDH) Problem*** *[20]*

Let $x, y \overset{R}{\leftarrow} Z_p^*$. Given (g, g^x, g^y), it is computationally intractable to compute the value of g^{xy}, where G is a cyclic group of order q and g is a generator of G.

Definition III.2. ***Static Diffie-Hellman Problem*** *[21]*

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

Definition III.3. ***t-Strong Diffie-Hellman (t-SDH) Problem*** *[17]*

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

Theorem III.4. *If $g^{f_{\vec{A}}(\alpha)}$ can be forged by an existed probabilistic polynomial time adversary Adv, we can construct an algorithm B to efficiently compute the solution to the t-SDH problem based on the Adv.*

Using the similar idea of ref.[22], we prove Theorem.III.4 as:

Proof: Suppose there exists a probabilistic polynomial time adversity Adv that can generate $f_{\vec{A_1}}(\alpha)$ such that $g^{f_{\vec{A_1}}(\alpha)} = g^{f_{\vec{A}}(\alpha)}$, where $f_{\vec{A}}(x)$ and $f_{\vec{A_1}}(x)$ are known to the Adv. The Adv can construct another polynomial $f_{\vec{A_2}}(x) = f_{\vec{A}}(x) - f_{\vec{A_1}}(x)$. Therefore, $g^{f_{\vec{A_2}}(\alpha)} = g^{f_{\vec{A}}(\alpha)} / g^{f_{\vec{A_1}}(\alpha)} = g^{f_{\vec{A}}(\alpha) - f_{\vec{A_1}}(\alpha)} \in Z_q[x]$. As $f_{\vec{A_1}}(\alpha) = f_{\vec{A}}(\alpha)$ and $f_{\vec{A_2}}(\alpha) = 0$, α becomes a root of polynomial $f_{\vec{A_2}}(x)$. By factoring $f_{\vec{A_2}}(x)$[23], B can find α and easily find a number c to get

$(c, g^{\frac{1}{\alpha+c}})$ as solution to the instance of the t-SDH problem given by the system parameters. ■

Theorem III.5. *If there exists a probabilistic polynomial time adversary Adv that can pass the verification in our proposed scheme with invalid proof information Prf', where $Prf' \neq Prf$ and Prf is the honest proof information, we can construct an algorithm B that uses Adv to solve the CDH problem, the Static Diffie-Hellman problem or the t-SDH problem.*

Proof: Suppose a probabilistic polynomial time adversity Adv can generate a $Prf' = (\psi', \sigma', y'), (\psi', \sigma', y') \neq (\psi, \sigma, y)$ and pass the verification in our proposed scheme, we can get the following two equations:

$$e(\eta, \kappa) \cdot e(\psi, \nu \cdot \kappa^{-r}) \stackrel{?}{=} e(\sigma, g) \cdot e(\kappa^{-y}, g) \quad (8)$$

$$e(\eta, \kappa) \cdot e(\psi', \nu \cdot \kappa^{-r}) \stackrel{?}{=} e(\sigma', g) \cdot e(\kappa^{-y'}, g) \quad (9)$$

Dividing Eq.8 with Eq.9, we obtain:

$$\frac{e(\psi, \nu \cdot \kappa^{-r})}{e(\psi', \nu \cdot \kappa^{-r})} = \frac{e(\sigma, g)}{e(\sigma', g)} \cdot e(\kappa^{(y'-y)}, g) \quad (10)$$

Now we do a case analysis for Prf'.
Case 1: $\sigma \neq \sigma'$. We rewrite Eq.10 as

$$\begin{aligned} e(\sigma, g) &= \frac{e(\psi, \nu \cdot \kappa^{-r})}{e(\psi', \nu \cdot \kappa^{-r})} \cdot e(\kappa^{(y-y')}, g) \cdot e(\sigma', g) \\ e(\eta, \kappa) &= \frac{e(\kappa^{-y'}, g) \cdot e(\sigma', g)}{e(\psi', \nu \cdot \kappa^{-r})} \\ e(\eta, \kappa) &= \frac{e(g^{-y'}, \kappa) \cdot e(\sigma'^{\frac{1}{\epsilon}}, \kappa)}{e(\psi'^{(\alpha-r)}, \kappa)} \end{aligned} \quad (11)$$

where $\eta = u^{\sum_{i \in K} p_i H(name||i)}$ is known to the Adv. We denote ψ' as $g^{\theta'}$, η as g^{ρ}, σ' as $g^{\pi'}$, based on Eq.11 we can get

$$\begin{aligned} e(g^{\rho}, \kappa) &= \frac{e(g^{-y'}, \kappa) \cdot e(g^{\frac{\pi'}{\epsilon}}, \kappa)}{e(g^{\theta'(\alpha-r)}, \kappa)} \\ \rho &= -y' + \frac{\pi'}{\epsilon} - \theta'(\alpha - r) \\ (\theta'(\alpha - r) + \rho)\epsilon &= -y'\epsilon + \pi' \end{aligned} \quad (12)$$

In this case, the Adv can output $g^{(\theta'(\alpha-r)+\rho)\epsilon} = \kappa^{-y'} \cdot \sigma'$. Now, if the Adv knows the value of θ', it can get $(\nu \cdot \kappa^{-r})^{\theta'} \cdot \eta^{\epsilon} = \kappa^{-y'} \cdot \sigma'$. That is, given g and g^{ϵ}, where ϵ is unknown, the Adv solve the Static Diffie-Hellman problem instance with $u^{\epsilon} = (\frac{\kappa^{-y'} \cdot \sigma'}{(\nu \cdot \kappa^{-r})^{\theta'}})^{\frac{1}{\sum_{i \in K} p_i H(name||i)}}$. If the Adv does not know the value of θ', the TA can give the Adv $\psi'^{(\alpha-r)} \cdot \eta = g^{(\theta'(\alpha-r)+\rho)}$ and $\kappa = g^{\epsilon}$, where ϵ and $(\theta'(\alpha - r) + \rho)$ are not known to the Adv, the Adv solve the CDH problem instance with $\kappa^{-y'} \cdot \sigma'$. Therefore, $\sigma' = \sigma$.
Case 2: $y \neq y'$. Here, we denote ψ as g^{θ} and ψ' as $g^{\theta'}$. If $y' \neq y$, the Adv divides Eq.8 with Eq.9 with $\sigma' = \sigma$ and outputs

$$\begin{aligned} \left(\frac{e(\psi, \kappa)}{e(\psi', \kappa)}\right)^{(\alpha'-r)} &= \frac{e(g, \kappa)^{-y}}{e(g, \kappa)^{-y'}} \\ \theta(\alpha' - r) + y &= \theta'(\alpha' - r) + y' \\ \frac{(\theta - \theta')}{y' - y} &= \frac{1}{\alpha' - r} \end{aligned} \quad (13)$$

Note that the operations in Eq.13 are modular operations with module q. In this case, the Adv can compute

$$(\frac{\psi}{\psi'})^{\frac{1}{y'-y}} = g^{\frac{1}{\alpha'-r}} \quad (14)$$

and outputs $(-r, g^{\frac{1}{\alpha-r}})$ as a solution for t-SDH problem by given system parameters. Therefore, $y' = y$.
Case 3: $\psi \neq \psi'$. The Adv divides Eq.8 with Eq.9 with $\sigma' = \sigma$, $y' = y$ and outputs

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

As $\psi' \neq \psi$, the Adv can infer $\alpha' = r$ based on Eq.15. In this case, the Adv can also output $(r, g^{\frac{1}{\alpha'+r}})$ as a solution of the t-SDH problem by given the system parameters. Therefore, $\psi' = \psi$. In addition, as we proved in Theorem III.4, $\psi = g^{f_{\vec{A}}(\alpha)}$ cannot be forged. That is, when the Adv output $\psi' = \psi$, it have to computed based on actual data blocks according to our PCAD scheme.

Based on the our above analysis, we proved that there is no Adv that can use invalid proof information and pass the verification in our PCAD scheme with non-negligible probability. Theorem.III.5 is proved. ■

Now, we prove the security of the data integrity verification after deduplication.

Theorem III.6. *If there exists a probabilistic polynomial time adversary Adv that can collude with $W-1$ owners and pass $owner_t$'s integrity verification after deduplication with invalid proof information Prf', where $Prf' \neq Prf$ and Prf is the honest proof information, we can construct an algorithm B that uses Adv to solve the Static Diffie-Hellman problem, the CDH problem or the t-SDH problem.*

Proof: As the proof of $Theorem$ III.6 is similar to the proof of $Theorem$ III.5, we just give the idea and key differences here due to the space limitation. We leave the detail proof in the full version of this paper.

Suppose a probabilistic polynomial time adversity Adv can generate a Prf' and pass the integrity auditing, we can divide the two equations generated based on Prf and Prf' as in the proof of $Theorem$ III.5. The difference between $Theorem$ III.5 and $Theorem$ III.6 is the known knowledge of the Adv. In this scenario, the Adv can get $\epsilon_w, w \in W/t$ besides the information it can get in $Theorem$ III.5. However, the information in original authentication tags of $owner_t$ remains the same as in $Theorem$ III.5. To prove the security of $Theorem$ III.6, we can do case analysis of $\sigma' \neq \sigma$, $y' \neq y$ and $\psi' \neq \psi$ first similar to $Theorem$ III.5.

Then we can do further analysis for $\kappa'' \neq \kappa'$ and $\nu'' \neq \nu'$. ■

IV. Performance Evaluation

A. Numerical Analysis

In this section, we numerically analyze our PCAD scheme and compare it with ref.[15], [5], [14]. For simplicity, in the rest of this paper, we use MUL and EXP[1] to denote the complexity of one multiplication operation and one exponentiation operation on Group G respectively. $Pairing$ is a bilinear pairing operation.

1) Communication: In our PCAD scheme, the communication cost of the auditing process is caused by the challenging message $CM = \{K, r\}$ and the proof information $Prf = \{\psi, \sigma, y\}$. The CM consists of a set K with k block ids and a random number r. As we discussed in Section III-E, the user can randomly challenge $k = 600$ data blocks to assure at least 99.999% error detection probability. If an error detection probability a fixed parameter, the size of K can be considered as constant and the complexity of challenging message CM is $O(1)$. The proof information is composed a polynomial y and two group elements ψ and σ. Therefore, the total communication complexity of auditing process in our PCAD scheme is also $O(1)$. In the $Deduplication$ process of our scheme, the user only needs to send d encoded data blocks to the cloud server to prove that it actually owns the whole file. As we discussed in Section III-E, the cloud server only needs challenging 300 blocks or 460 blocks to achieve 95% or 99% confidence whether the user actually owns the whole data file. Therefore, the size of D can be bounded and the total communication complexity of the $Deduplication$ process in our scheme is $O(1)$.

Now, we compare our PCAD scheme with existing schemes [15], [5], [14] and show the result in Table.1. In ref.[15], the $Auditing$ process requires the cloud server to send k authentication tags of the challenging blocks and s aggregated data blocks to the user, where s is the number of elements in an encoded block. Thus, its communication complexity during the $Auditing$ process is $O(s + k)$. To perform the $Deduplication$ process, the user needs to sends $2s$ aggregated data blocks to the cloud server and thus introduces the communication complexity as $O(s)$. Differently, the aggregation of communication information in our design enables our scheme to achieve $O(1)$ communication complexity for both $Auditing$ and $Deduplication$ processes. The POR schemes proposed by Yuan et al. [5] achieves constant communication complexity for the $Auditing$ process same as our PCAD scheme. However, their scheme does not support the $Deduplication$ process and batch auditing, and introduces much higher computational cost on the user side (Discuss later in Section IV-A2). Considering the deduplication process only, ref.[14] also requires $O(1)$ communication cost, but their scheme cannot support the data integrity auditing.

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

2) Computation: As shown in Section III-B, our PCAD scheme consists of 6 algorithms: $KeyGen$, $Setup$, $Challenge$, $Prove$, $Verify$ and $Deduplication$. Among these algorithms, $KeyGen$ and $Setup$ are prepossessing procedures, which are performed by the data owner off-line. To produce authentication tags for a encoded file with n blocks, each of which has s elements, the data owner needs $(s+2)n$ EXP and sn MUL operations. Note that the cost in the prepossessing of our scheme is one-time cost for the data owner. After these prepossessing procedures, the data owner can go off-line. During the data integrity auditing process, the user performs $Challenge$ algorithm to generate the challenging message CM by choosing a constant number of random numbers with negligible cost. On receiving the CM, the cloud server needs $(k + s - 1)$ MUL and $(s+k)$ EXP operations to produce the proof information. To verify the integrity of the auditing file, the user performs 3 EXP, 3 MUL and 4 Pairing operations. Therefore, the computational cost for the user to audit the data integrity of a single file is $O(1)$MUL+$O(1)$EXP+$O(1)$Pairing. To perform the $Deduplication$ algorithm in our scheme, no computation cost is required for the user. The cloud server needs to perform $O(s + d)$MUL+$O(s)$EXP+$O(1)$Pairing.

We now compare our PCAD scheme with existing schemes [15], [5], [14] and summarize the result in Table.1. In ref.[15], the data integrity auditing process costs a user $O(ks)$MUL+$O(k)$EXP operations, and the deduplication processes introduces $O(sk)$MUL computational complexity to the user, where k is the number of challenging blocks and s is the number of element in a data block. Differently, by outsourcing most computational tasks of both auditing and deduplication processes to the cloud server, our PCAD scheme achieves constant computational cost on users and thus significantly outperforms ref.[15]. Compared with ref.[5] that only supports data integrity auditing, our PCAD scheme reduces the computational complexity on the user from $O(k)$MUL+$O(k)$EXP+$O(1)$Pairing to $O(1)$MUL+$O(1)$EXP+$O(1)$Pairing as shown in Table.1. Considering only the deduplication process, ref.[14] requires $O(1)$PRF operation on the user side that is comparable to our PCAD scheme, where PRF is one pseudorandom function operation.

3) Auditing After Deduplication: In this section, we discuss the storage overhead saved by aggregation of authentication tags in our proposed scheme. Suppose W owners $owner_w, 1 \leq w \leq W$ pass the deduplication checking of the file F' existed on the cloud server. As these owners have no mutual trust with each other, each $owner_w$ needs to store n authentication tags on the cloud server separately for future public integrity auditing of F', where n is the number of encoded data blocks in F'. If the cloud server directly store these authentication tags, a $O(Wn)$ storage overhead complexity is introduced to it. Differently, by aggregating the tags for the same data block, the cloud server in our scheme can reduce the storage overhead complexity to $O(n)$. With regard to the computational complexity and

	Ref.[5] (POR)	Ref.[14] (POW)	Ref.[15] (POSD)	Our PCAD
Public Auditing	Yes	No	Yes	Yes
Deduplication	No	Yes	Yes	Yes
Secure	Yes	Yes	No	Yes
Batch Auditing	No	No	No	Yes
Prepossessing	$O(sn)$MUL+$O(sn)$EXP	$O(n)$PRF	$O(sn)$MUL+$O(n)$EXP	$O(sn)$MUL+$O(sn)$EXP
Auditing Comp.Cost (Cloud)	$O(s+k)$MUL+$O(s+k)$EXP	N/A	$O(ks)$MUL+$O(k)$EXP	$O(k+s)$MUL+$O(s+k)$EXP
Auditing Comp.Cost (User)	$O(k)$MUL+$O(k)$EXP +$O(1)$Pairing	N/A	$O(ks)$MUL+$O(k)$EXP	$O(1)$MUL+$O(1)$EXP +$O(1)$Pairing
Auditing Comm.Cost	$O(1)$	N/A	$O(s+k)$	$O(1)$
Deduplication Comp.Cost (Cloud)	N/A	$O(n)$PRF	$O(sk)$MUL+$O(k)$EXP	$O(s+d)$MUL+$O(s)$EXP +$O(1)$Pairing
Deduplication Comp.Cost (User)	N/A	$O(1)$PRF	$O(sk)$MUL	0
Deduplication Comm.Cost	N/A	$O(1)$	$O(s)$	$O(1)$

Table.1 Complexity Summary: in this table, n is number of encoded blocks for the file, s is the number of elements in each block and k is number of blocks selected for challenging; PRF is one pseudorandom function operation, EXP and MUL are one multiplication operation and one exponentiation operation on Group G respectively

communication complexity on an auditing user, it remains the same as the constant level of auditing before deduplication, i.e., $O(1)$MUL+$O(1)$EXP+$O(1)$Pairing computational complexity and $O(1)$ communication complexity.

4) Batch Auditing: In this section, we discuss the communication cost and computational cost saved by our batch auditing design for multiple requests scenarios. Suppose a TPA is hired by T data owners to help them audit the integrity of L outsourced files on the cloud server periodically. If the TPA processes these L auditing requests one by one, it needs $3L$ EXP, $3L$ MUL and $4L$ Pairing operations for computation, and $2L$ group elements, L random numbers and L polynomials for communication. With our batch auditing design, the cloud server can aggregate L σ_i into one group element and use one random number, one polynomial instead of L ones. Thus, compared with processing requests sequentially, our batch auditing design can help the TPA and the cloud server to save about 50% communication cost. From the perspective of computational cost, our batch auditing design enables the TPA to reduce number of Pairing operations from $4L$ to $3L$, which is much more expensive compared with MUL and EXP operations. Therefore, about 25% computational tasks are saved for the TPA with our batch auditing design. Assume $c\%$ files are from same data owners, our batch auditing design can save additional $c\%$ communication cost and $c\%$ computational cost.

B. Experimental Result

To show that our proposed PCAD scheme is efficient and scalable, we conducted experiments on Amazon EC2 Cloud Platform using JAVA with Java Pairing-Based Cryptography library (jPBC) [24]. The machine we used for the TPA is a laptop running Mint Linux 13 with 2.50GHz Intel i5-2520M CPU and 8GB memory. For the cloud server, we utilize nodes that run Red Hat Enterprise Linux 6.3 with 8 Cores CPU and 16GB memory. We set the security parameter $\lambda = 160$, which achieves 1024-bits RAS equivalent security on Group. All experimental results represent the mean of 10 trials.

To verify our PCAD scheme's constant communication cost and computational cost on the user side, we vary the number of data blocks stored on the cloud server from 1000 to 10000. As shown in Fig.1 (a), the computational cost of users for performing an integrity auditing task almost keeps around 420 ms when the number of data blocks in the auditing files increases. With regard to the communication cost, it also remains stable as about 622 Bytes when the number of data blocks in the auditing files increases as shown in Fig.1 (b). Note that, although we do not perform experiment on more large files, it is easy to obtain that both computational cost and communication cost of our scheme are constant from the analysis in Section IV-A2 and Section IV-A1.

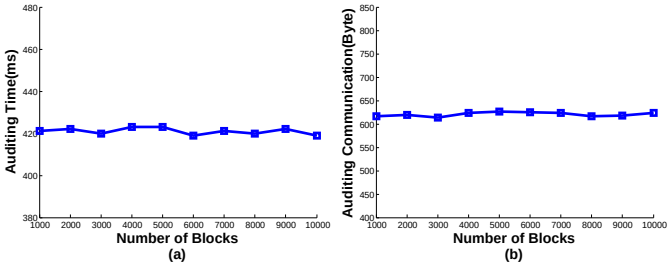

Fig. 1. (a) Auditing Time on Users (b) Auditing Communication Cost on Users

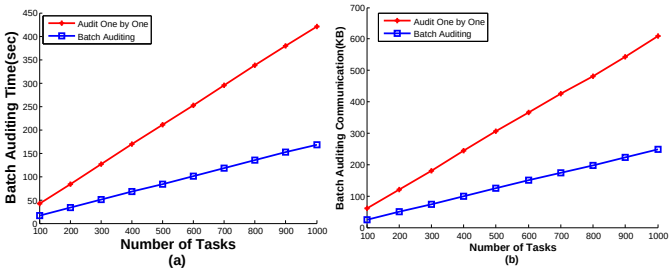

Fig. 2. (a) Auditing Time on TPA (b) Auditing Communication Cost on TPA

To show the benefits of our batch auditing design for multiple auditing tasks scenarios, we change the number of tasks a TPA needs to perform from 100 to 1000. Among

#Tasks = 500	Average Comp.Cost	Average Comm.Cost
Auditing One by One	423 ms	626 Bytes
Batch Auditing	181 ms	259 Bytes

Table.2. Average Computational Cost and Communication Cost for Batch Auditing and Single Auditing

these files, 20% files are from same data owners. As we demonstrated above, the number of data blocks in each file does not influence the performance of our scheme, we set the number of data blocks to 5000 in each auditing task. Compared with performing these auditing tasks one by one, Fig.2 (a) shows that the TPA can save about 55% auditing time with batch auditing. From the perspective of communication cost, Fig.2 (b) shows our batch auditing saves about 60% bandwidth for the TPA. Considering the average cost per task, which is computed by dividing total auditing time and total auditing bandwidth cost by the number of tasks respectively, Table.2 shows that our batch reduce the computation cost per task on the TPA from 423 ms to 181 ms and the bandwidth cost per task from 626 Bytes to 259 Bytes.

V. Conclusion

To securely fulfill the two important requirements of cloud storage: data integrity and storage efficiency, a number of schemes have been proposed based on the concepts of POR, PDP, POW and POSD. However, most existing schemes only focus on one aspect, because trivial combination of existing POR/PDP schemes with POW schemes can contradict the objects of POW. The only one that simultaneously emphasized both aspects based on the concept of POSD suffers from tremendous computation and computational costs and has been proven not secure. In this work, we filled the gap between POR and POW and proposed a constant cost scheme that achieves secure public data integrity auditing and storage deduplication at the same time. Our proposed scheme enables the deduplication of both files and their corresponding authentication tags. In addition, we extend our design to support batch integrity auditing, and thus substantially save computational cost and communication cost for multiple requests scenarios. The security of our PCAD scheme is proved based on the CDH problem, the Static Diffie-Hellman problem and the t-SDH problem. We validate the efficiency and scalability of our scheme through numerical analysis and experimental results on Amazon EC2 Cloud. Our proposed polynomial based authentication tag can also be used as an independent solution for other related applications, such as verifiable SQL search, encrypted key word search, etc.

References

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

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

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

[4] J. Gantz and D. Reinsel, "The digital universe decade - are you ready?" *http://www.emc.com/collateral/analyst-reports/idc-digital-universe-are-you-ready.pdf*, May 2010.

[5] J. Yuan and S. Yu, "Proofs of retrievability with public verifiability and constant communication cost in cloud," *Proceedings of the ACM ASIACCS-SCC'13*, 2013.

[6] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security*, ser. ASIACRYPT '08, Berlin, Heidelberg, May 2008, pp. 90–107.

[7] 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*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 584–597.

[8] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Proceedings of the 6th Theory of Cryptography Conference on Theory of Cryptography*, ser. TCC '09, Berlin, Heidelberg, 2009, pp. 109–127.

[9] 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*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 598–609.

[10] 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*, ser. SecureComm '08. New York, NY, USA: ACM, 2008.

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

[12] Q. Wang, C. Wang, K. Ren, W. Lou, and J. Li, "Enabling public auditability and data dynamics for storage security in cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 22, no. 5, pp. 847–859, 2011.

[13] S. Halevi, D. Harnik, B. Pinkas, and A. Shulman-Peleg, "Proofs of ownership in remote storage systems," in *Proceedings of the 18th ACM conference on Computer and communications security*, ser. CCS '11. New York, NY, USA: ACM, 2011, pp. 491–500.

[14] R. Di Pietro and A. Sorniotti, "Boosting efficiency and security in proof of ownership for deduplication," in *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ser. ASIACCS '12. New York, NY, USA: ACM, 2012, pp. 81–82.

[15] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, ser. CODASPY '12. New York, NY, USA: ACM, 2012, pp. 1–12.

[16] K. K. Youngjoo Shin, Junbeom Hur, "Security weakness in the proof of storage with deduplication," Cryptology ePrint Archive, Report 2012/554, 2012, http://eprint.iacr.org/.

[17] D. Boneh and X. Boyen, "Short signatures without random oracles," in *EUROCRYPT*, 2004, pp. 56–73.

[18] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journal of the Society for Industrial and Applied Mathematics*, vol. 8, no. 2, pp. 300–304, 1960.

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

[20] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Sep. 1976.

[21] D. R. L. Brown and R. P. Gallant, "The static diffie-hellman problem," Cryptology ePrint Archive, Report 2004/306, 2004, http://eprint.iacr.org/.

[22] A. Kate, G. M. Zaverucha, and I. Goldberg, "Constant-size commitments to polynomials and their applications," in *ASIACRYPT*, 2010, pp. 177–194.

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

[24] jPBC, "http://gas.dia.unisa.it/projects/jpbc/."