IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received April 9, 2019, accepted May 5, 2019, date of publication May 20, 2019, date of current version June 21, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2917919

Efficient Group Proof of Storage With Malicious-Member Distinction and Revocation

LIEHUANG ZHU[1], (Member, IEEE), HONGYUAN WANG[1], CHANG XU[1], KASHIF SHARIF[1], (Member, IEEE), AND RONGXING LU[2], (Senior Member, IEEE)

[1]Beijing Engineering Research Center of Massive Language Information Processing and Cloud Computing Application, School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[2]Faculty of Computer Science, University of New Brunswick, Fredericton, NB E3B 5A3, Canada

Corresponding author: Chang Xu (xuchang@bit.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61402037 and Grant 61272512, and in part by the National Key Research and Development Program under Grant 2016YFB0800301.

ABSTRACT Proof of Storage (POS) is a system utilized by a client to verify whether the original data is intact while being possessed by an untrusted server. In a grouping application, multiple members share and verify the same file, and the group manager is responsible for determining if the data has been manipulated based on the responses from group members. However, a malicious member may repudiate a correct proof; therefore, it is important to distinguish the honest members from malicious ones. To the best of our knowledge, none of the existing group-oriented schemes have solved this challenge efficiently and up to the desired satisfaction. In this paper, based on matrix calculation, pseudo-random functions, and commitment functions, we propose a new Group Proof of Storage with Malicious-Member Distinction and Revocation scheme (DR-GPOS). Specifically, in terms of functionality, DR-GPOS can distinguish and revoke the malicious members, as well as, guarantee the integrity and deduplication of the outsourced data. From a security perspective, DR-GPOS can also resist against selective attacks and the collusion attacks from the revoked members (e.g. forging proofs by colluding with the server). The security properties of the proposed schemes have also been formally proven in a standard model. We have further implemented it in a real-world (Baidu) cloud server, to evaluate the performance with large scale data ($>$ 10 G).

INDEX TERMS Malicious-member distinction, malicious-member revocation, deduplication, proof of storage, cloud storage.

I. INTRODUCTION

Cloud computing [1]–[3] in recent years has provided a low-cost and scalable services for clients and has been widely used in industrial and academic communities. The most fundamental and popular service is the cloud storage, which has been widely used, and provided by many vendors (e.g. Google Drive, Amazon S3, Dropbox, etc.).

However, there still exist a number of security issues, which lead to data loss in cloud storage. For instance, T-Mobile[1] suffered phone sales hit by sidekick loss. Afterwards, Google Gmail[2] has encountered large-scale data losses in 2011. Hence, these accidental or malicious breaches incur huge losses for the clients as well as the enterprises. Therefore, data integrity of cloud storage verification has become an important research issue.

In order to solve aforementioned challenge, a series of proof of storage (POS) schemes [4]–[7] have been proposed. In some actual applications, multiple users jointly manage and verify the same data. For instance, data of some internet companies (e.g. purchase records and browsing history) is exploding every day. Once the data is damaged, the company suffers serious losses. To solve this problem, some professional verifiers are required to check and preserve the data regularly. These verifiers can be considered as members of a group who manage and share the same data. The group manager concludes if the data is intact when being possessed based on the group members' responses.

If malicious members exist in the group, the malicious members should be identified and revoked. In a real world, malicious members may also collude with the cloud server to

[1]http://news.bbc.co.uk/2/hi/technology/8303952.stm
[2]http://www.redorbit.com/news/technology/2003817/

The associate editor coordinating the review of this manuscript and approving it for publication was Kuo-Hui Yeh.

forge valid proofs. Therefore, if different verification results are returned from group members, the dishonest one should be revoked.

Public verification schemes can avoid this trouble, since anyone can verify the data integrity. However, public schemes cannot be efficiently executed since they are generally constructed based on the bilinear pairing or third-party verification.

Therefore, an efficient POS system needs to be established to meet the following requirements:

1 **Achieve data integrity in group applications**: The scheme should prove the data possession efficiently and accurately in a group environment. Each group member can verify the proof independently.

2 **Distinguish malicious members from honest ones**: If different verification results are returned, the scheme can exactly distinguish the dishonest member from honest members.

3 **Revoke malicious members**: When a malicious member is identified, the scheme can revoke its access, which means that the secret key of a revoked member cannot be used to forge a valid tag or proof, even if the malicious member colludes with the server. Additionally, the revoked member cannot derive any secret key of other members.

4 **Resist selective opening attacks**: In the group, multiple members with different keys correspond to the same verification tag. For security reasons, if the secret keys of some members are leaked, the adversary (possibly a cloud server or a malicious third party) cannot use the information to forge a legitimate tag or proof.

5 **Support data deduplication**: For existing work, each member holds its own metadata (e.g. verification tags). If all the members with different secret keys share the same metadata (i.e. deduplication is achieved), the storage costs will be reduced. That is, if deduplication is captured, the metadata of the system should be only related to the total data size, and has no relation with the number of members.

Wang et al. proposed a GPDP scheme [8] to guarantee the properties of data deduplication, data integrity and resisting selective attacks, i.e. the above requirements 1, 4 and 5. In order to achieve all of the above properties, we will present a new DR-GPOS scheme which is based on matrix calculation, pseudo-random functions, and commitment functions. The four-fold concrete contributions of this paper are summarized as follows:

1. In terms of functionality, our proposed scheme can accurately distinguish and securely revoke the malicious member, as well as guarantee the integrity and deduplication of the outsourced data.
2. From a security perspective, our proposed scheme can resist against the selective attacks and the collusion attacks from revoked members (e.g. forging proof by colluding with server).
3. To verify its security, we define three security games to capture properties of member distinction, member revocation, and proof of storage respectively, and then give the formal security analysis in the standard model.
4. In order to evaluate the performance of the scheme, we implement a prototype of DR-GPOS scheme on Baidu cloud server, and utilize data size of 10G to measure the efficiency of DR-GPOS.

The rest of this paper is organized as follows: We give the overview of our idea in section 2, and the preliminaries in section 3. Then in section 4, the definition and concrete construction of DR-GPOS are introduced in detail. We further give the security analysis and performance evaluation in section 5 and section 6, respectively. Finally, the related work is introduced in section 7 and the conclusion is given in section 8.

II. OVERVIEW OF OUR IDEA

DR-GPOS scheme consists of four phases, as shown in Fig.1: a) preprocessing, b) challenge and verification, c) malicious-member distinction, and d) malicious-member access revocation.

In the pre-process phase, the trusted third party (TTP) decomposes the file and generates the verification tags (Tags), commitment functions (COMs), and the processed data (File). Afterwards, TTP forwards the data to the server, distributes the secret keys to each group member, and deletes all local file data.

In the challenge and verification phase, each member sends an independent challenge *chal* to the server, against which the server generates the proofs and returns it respectively to the members. The returned proofs are verified for their validity, by using the local metadata at each member location.

The third phase has two sub-cases:

Case 1: If all members accept their corresponding proofs returned by the server, then the server can guarantee the possession of the group's data at given time.

Case 2: In the scenario, where some members do not accept the returned proofs, then it can conclude that either some members are being dishonest, or the server has been compromised. In this situation, the server is required to open the commitment functions to prove its innocence and distinguish the dishonest members. The commitment functions can be verified by anyone, and the dishonest entities can be sought out publicly.

In the phase of malicious-member revocation, the dishonest members should be revoked. Note that the revoked members cannot use their keys to forge legal proofs and pass the verification as legitimate members, even if they collude with the server. In addition, the revoked keys cannot be used to derive any other valid secret keys.

III. PRELIMINARIES

In this section, we introduce homomorphic Macs, cross-authentication codes, commitment function, and the security assumptions, which serve as a basis of the proposed scheme.

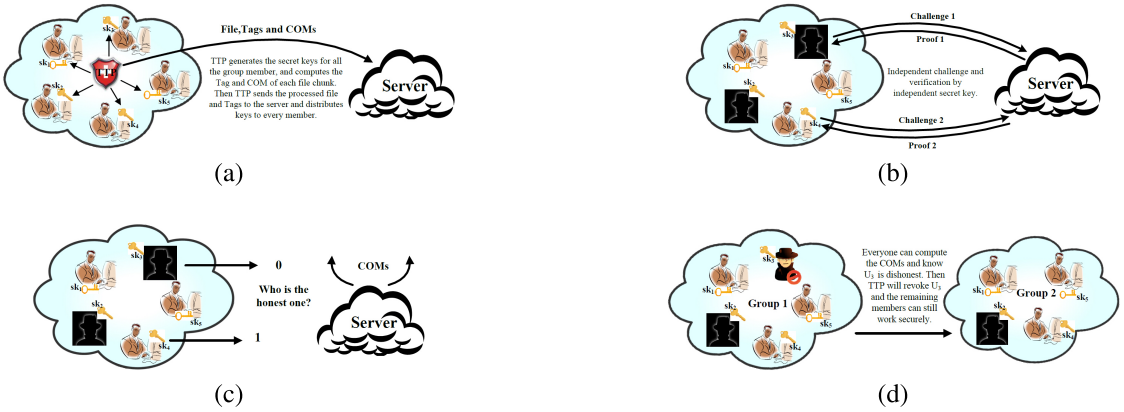


FIGURE 1. The DR-GPOS scheme. (a) The pre-process (b) The challenge and verification (c) The malicious-member distinction (d) The malicious-member revocation.

A. HOMOMORPHIC MACS

The definition of Homomorphic Message Authentication Code (HomMac) was initially proposed by Agrawal and Boneh [9]. We utilize boldfaced $\boldsymbol{v}$ to represent a vector.

Definition 1: A HomMac consists of four polynomial-time algorithms (*Gen*, **Sign**, *Combine*, *Verify*) as follows:

- $Gen(1^{\lambda}) \rightarrow (pk, sk)$ is a probabilistic algorithm, which creates the public-private key pair (pk, sk), based on the security parameter input λ.
- $Sign(sk, id, \boldsymbol{v}_i, i) \rightarrow t_i$ is an algorithm to generate a tag of each vector. It takes secret key sk, the vector space identifier id, vector $v_i \in F_q^s$ (s is the length of vector), and its index i as input, and returns the tag t_i of $\boldsymbol{v}_i$.
- $Combine((\boldsymbol{v}_1, t_1, \alpha_1), ..., (\boldsymbol{v}_c, t_c, \alpha_c), pk) \rightarrow (T, y)$ is an algorithm to generate a homomorphic Mac. It takes public key pk, c vectors $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_c \in F_q^s$, the related tags $t_1, ...t_c$, and c random constants $\alpha_1, \ldots, \alpha_c \in F_q$ as input. Then it computes $\boldsymbol{y} = \sum_{i=1}^{c} \alpha_i \boldsymbol{v}_i \in F_q^s$ and the corresponding tag T, and returns T and $\boldsymbol{y}$.
- $Verif(pk, sk, id, \boldsymbol{y}, T) \rightarrow \{1, 0\}$ is a deterministic algorithm to verify a homomorphic Mac. It takes the key pair (pk, sk), the vector space identifier id, vector $\boldsymbol{y} \in F_q^s$, and the corresponding T as input. If the verification is accepted, 1 is returned, otherwise 0.

B. CROSS-AUTHENTICATION CODES

1) DEFINITION OF CROSS-AUTHENTICATION CODES

The definition of Cross-Authentication Codes was given by Fehr et al. [10].

Definition 2: (L-Cross-Authentication Codes, *short:* L-XAC). Let XK be the key space and XT be the tag space. For $L \in N$: A L-XAC consists of three polynomial-time algorithms (**XGen**, **XAuth** and **XVer**) as follows:

The returned value of $\{0, 1\}$ shows if the input T is the correct tag (generated by *chal*), for one of the secret keys K_i.

- $XGen(1^k) \rightarrow K$ is a probabilistic algorithm to generate a uniform random key. Based on the security parameter k, the secret key K is returned.
- $XAuth(K_1, K_2, \ldots, K_L) \rightarrow T$ is an algorithm to generate the tags for verification. A set of keys $K_1, K_2, \ldots, K_L$ are given as input, and the corresponding tag $T \in$ XK is returned.
- $XVer(K_i, T) \rightarrow \{1, 0\}$ is an algorithm to verify a tag. The returned value of $\{0, 1\}$ shows if the input T is the correct tag (generated by *chal*), for one of the secret keys K_i.

The following properties are required:

1. **Correctness:**
   For all $i \in [L]$, the probability $fail_{XAC}(k)$ is negligible, where

$$fail_{XAC}(k) := \Pr[XVer(K_i, i, XAuth(K_1, K_2, \ldots, K_L)) \neq 1].$$

2. **Security Against Impersonation Attacks:**
   Intuitively, when the key K is not the generation key of the tag T', even if the whole tag space is accessed, it is difficult to find a T' which can be verified by K. The probability of $Adv_{XAC}^{imp}(k)$ is defined as:

$$Adv_{XAC}^{imp}(k) := \max_{i,T'} \Pr[XVer(K, i, T') = 1 | K \leftarrow XGen(1^k)] \leq negl(k)$$

   where the max is applied over all $i \in [L]$ and $T' \in$ XT.
3. **Security Against Substitution Attacks:**
   Intuitively, it is easy to verify the correctness of the tag T with the key K_i. But in the absence of K_i, even if the adversary has a correct tag T and all other keys $K_{\neq i}$, it is difficult to forge a legitimate label T' which can be verified by K_i. The probability of $Adv_{XAC}^{sub}(k)$ is defined as

$$\max_{i,K_{\neq i},F} \Pr\left[\begin{array}{c} T' \neq T \\ \wedge \\ XVer(K,i,T')=1 \end{array} \middle| \begin{array}{c} K_i \leftarrow XGen(1^k) \\ T := XAuth(K_1, K_2, \ldots, K_L) \\ T' \leftarrow F(T) \end{array}\right] \leq negl(k).$$

Notice that the max is applied over all $i \in [L]$, $K_{\neq i} = (K_j)_{j \neq i} \in \mathrm{XK}$, and all possible functions $F : \mathrm{XT} \to \mathrm{XT}$.

2) AN EXAMPLE OF L-XAC

Let $\mathrm{XK} = F^2$ and $\mathrm{XT} = F^L \cup \{\perp\}$, where F is a finite field of size q that depends on security parameter k (e.g. $q = 2^k$).

- **XGen:** generates a set of random keys $K_1 = (a_1, b_1), \ldots, K_L = (a_L, b_L) \in \mathrm{XK}$.
- **XAuth:** generates the authentication tag $T = (T_0, T_1, \ldots, T_{L-1}) \in F^L$ such that $p_T(a_i) = b_i$ for $i = 1, 2, \ldots, L$, where satisfy $p_T(x) = T_0 + T_1 x + \ldots + T_{L-1} x^{L-1} \in F[x]$. The linear equation $AT = B$ can be used to efficiently compute the value of T. It is important to note that $A \in F^{L \times L}$ is a Vandermonde matrix, and $B \in F^L$ is the column vector $[b_1, b_2, \ldots, b_L]$. Moreover, the $i-$th row is given by $1, a_i, a_i^2, \ldots, a_i^{L-1}$.
- **XVer:** verify the tag T and output 1 if and only if $T \neq \perp$ and $p_T(a_i) = b_i$, $i \in [L]$.

The work in [10] proves that L-XAC satisfies the conditions: $fail_{XAC}(k) \leq \frac{L(L-1)}{2q}$, $Adv^{imp}_{XAC}(k) \leq \frac{1}{q}$, and $Adv^{sub}_{XAC}(k) \leq 2 \cdot \frac{L-1}{q}$.

C. A REUSABLE COMMITMENT FUNCTION

The non-repudiation property in PDP/POR scheme has been achieved in Wang et al.'s scheme [7] by introducing a commitment scheme, which is reusable and binding. Based on Pedersen commitment function, the reusable commitment function is as follows.

Let $p = 2p' + 1$ and $q = 2q' + 1$ be safe primes and let $N = pq$ be the modulus of an RSA scheme. g is the generator of the unique cyclic subgroup whose order is $p'q'$. Let G_q be a group of prime order q in which computing the discrete logarithm function is intractable, and let h be elements of G_q where it is hard to compute $\log_g h \mod q$. Choose e and d, where e is a small prime and $ed = 1 \bmod \phi(N)$. A reusable commitment function ca be constructed as

$$COM(M) = (g^M h^{H(ID)})^e \bmod N$$

where H is a full-domain hash function. Notice that d can be viewed as a public trapdoor of the commitment.

Intuitively, it is a ramification of the Pedersen commitment function [11], therefore it is a perfect commitment scheme which is against an all-powerful receiver. Above reusable commitment function has two properties: information-theoretically hiding and computational binding.

1) INFORMATION-THEORETICALLY HIDING

- The committer $COM(M)$ is a random element of G_q, for any given a probabilistic polynomial time receiver. Hence, there is no distinction between a random element of G_q and $COM(M)$.
- it is possible to compute $\log_g h \bmod q$ by an all-powerful receiver. Hence, $\log_g COM \bmod q = M + H(ID) \log_g h$ can be obtained. Note that, it is impossible to determine the value of M with a random auxiliary value $H(ID)$.

2) COMPUTATIONAL BINDING

For a probabilistic polynomial time sender, it is difficult to compute $\log_g h \bmod q$, hence it is almost impossible to find a pair $(M', ID') \neq (M, ID)$ such that $COM(M', ID') \neq COM(M, ID)$. Consequently, the committer $COM(M)$ can be opened only in one way to demonstrate the original value M.

Remarks:

- ID is the unique identifier of M, and the hash function H is used to resist the forgery of multiple coefficient.
- d does not reveal the information of e, thus it ensures the commitment cannot be forged, which makes it reusable.
- When initiating a challenge, the committer discloses the value of M and ID, thus anyone can verify whether $s = t$ holds, where $s = g^M h^{H(ID)} \bmod N$ and $t = (COM(M))^d = (g^M h^{H(ID)})^{ed} \bmod N = g^M h^{H(ID)} \bmod N$.

D. SECURITY ASSUMPTIONS

1) DLP (DISCRETE LOGARITHM PROBLEM)

Let G be a group, $g \in G$, and $\langle g \rangle$ be the cyclic subgroup generated by g. Then the discrete logarithm problem on group G can be given as: Given $g \in G$ and $a \in \langle g \rangle$, determine if there exists an integer x that satisfy $g^x = a$, and find such an x. Notice that the problem is hard to solve in polynomial time.

2) PRF (PSEUDO-RANDOM FUNCTION)

Let I_k denote the set of all k-bit strings, and H_k be the set of all functions from I_k into I_k. Then the Pseudo-Random Function $F = F_k \subseteq H_k$ has the following properties:

- **Indexing:** Each function has a unique index k associated with it, thus randomly picking a function $f \in F_k$ is easy.
- **Polynomial time Computation:** Given an input x and a function $f \in F_k$, there exists a polynomial time algorithm to compute $f(x)$.
- **Pseudo-Randomness:** No probabilistic polynomial time algorithms can distinguish the functions in F_k and from the functions in H_k, as well as the value of $f(x)$ and a randomness in I_k.

IV. THE PROPOSED DR-GPOS SCHEME

The definition and detailed construction of DR-GPOS is given in the following section.

A. DEFINITION OF DR-GPOS

Definition 3: GPOS with Malicious-Member Distinction and Revocation (DR-GPOS). A DR-GPOS scheme is a collection of five polynomial-time algorithms (*KeyGen*, *PreGen*, *ProofGen*, *VerifProof* , *VerifCOM*). Each is defined as follows:

- $KeyGen(1^\lambda) \to (mK, K_l)$ is a probabilistic polynomial-time algorithm to generate keys. The input security parameter λ is used to generate the master key mK and the secret key K_l of each member.
- $PreGen(\{K\}_{l=1}^L, mK, m_i) \to (T_i, COM_i)$ is a polynomial-time algorithm which can generate the

metadata required to verify the proof. It uses the private keys $\{K\}_{l=1}^{L}$, the master key mK and a file block m_i as input, and returns the commitment function COM_i and the verification tags T_i.

- *ProofGen*$(File, chal, \Sigma) \rightarrow \rho$ is a polynomial-time algorithm to generate a proof of storage. The input consists of an ordered data collection *File*, a challenge *chal* and an ordered collection Σ of tags. It returns a proof ρ for the file data determined by *chal*.
- *VerifProof*$(K_l, chal, \rho) \rightarrow \{1, 0\}$ is a polynomial-time algorithm to verify a proof of storage. It takes secret key K_l of arbitrary member, a challenge *chal* and a proof ρ as input, and returns whether ρ is a correct proof for the data determined by *chal*.
- *VerifCOM*$(\{M\}, \{I\}, c, \{COM\}) \rightarrow \{1, 0\}$ is a polynomial-time algorithm to distinguish the dishonest member from honest members. It takes the chosen chunks $\{M\}$, the corresponding indices $\{I\}$ and commitment functions $\{COM\}$ as input, and returns whether each commitment is valid.

B. DETAILED CONSTRUCTION

The algorithm fist divides the file into n chunks, such that $File = \{m_{<1>}, m_{<2>}, \ldots, m_{<n>}\}$, where each chunk has L blocks: $m_{<i>} = \{m_{i,1}, m_{i,2}, \ldots, m_{i,L}\}$. Here, the commitment and tag of each chunk $m_{<i>}$ can be give as COM_i and $T_{<i>}$, where $1 \le i \le n$.

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 whose order is $p'q'$. Let F be a finite field of size 2^{λ}, then the key space $\kappa = F^3$ and tag space $\chi = F^L \cup \{\perp\}$.

Let H be a full-domain hash function and f be a pseudo-random function, and let π be a pseudo-random permutation. λ is the security parameter.

$$H : \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\log_2(n)}.$$
$$f : \{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\lambda}.$$
$$\pi : \{0,1\}^{\lambda} \times \{0,1\}^{\log_2(n)} \rightarrow \{0,1\}^{\log_2(n)}.$$

- *KeyGen*$(1^{\lambda}) \rightarrow (msk, mpk, sk_l)$: Let e and d be secret primes such that $ed = 1 \bmod p'q'$. Let L be the maximum number of group members. Given the security parameter λ, the outputs are $msk = (p, q, e)$, $mpk = (d, N, g, h)$ and a set of secret keys: $sk_l = (a_l, b_l, c_l) \in \kappa$, $1 \le l \le L$.
- *PreGen*$(\{sk_l\}_{l=1}^{L}, msk, m_{<i>}) \rightarrow (T_{<i>}, COM_i)$: For each i, $1 \le i \le n$, compute the verification tag of each chunk $m_{<i>}$:

$$\begin{bmatrix} 1, c_1, c_1^2, \ldots, c_1^{L-1} \\ 1, c_2, a_2^2, \ldots, c_2^{L-1} \\ \vdots \\ 1, c_L, c_L^2, \ldots, c_L^{L-1} \end{bmatrix} \cdot \begin{bmatrix} m_{i,1} \\ m_{i,2} \\ \vdots \\ m_{i,L} \end{bmatrix} + \begin{bmatrix} f_{b_1}(i) \\ f_{b_2}(i) \\ \vdots \\ f_{b_L}(i) \end{bmatrix}$$

$$= \begin{bmatrix} 1, a_1, a_1^2, \ldots, a_1^{L-1} \\ 1, a_2, a_2^2, \ldots, a_2^{L-1} \\ \vdots \\ 1, a_L, a_L^2, \ldots, a_L^{L-1} \end{bmatrix} \cdot \begin{bmatrix} T_{i,1} \\ T_{i,2} \\ \vdots \\ T_{i,L}. \end{bmatrix}$$

The tag $T_{<i>} = (T_{i,1}, T_{i,2}, \ldots, T_{i,L})$ can be computed by solving linear matrix equation group with L unknowns. For each i, $1 \le i \le n$, compute the commitment function for each chunk $m_{<i>}$: $COM_i = COM(m_{<i>}) = (g^{m_{<i>}} h^{H(i)})^e \bmod N$. Output $\{m_{<i>}, T_{<i>}, COM_i\}_{1 \le i \le n}$.

- *ProofGen*$(\{m_{<i>}\}_{1 \le i \le n}, \{T_{<i>}\}_{1 \le i \le n}, chal) \rightarrow \rho$ The challenge is $chal = (c, k_1, k_2)$, where c is the number of challenged chunks and k_1, k_2 are fresh random keys. For $1 \le z \le c$: 1. Compute the index of each sampled block: $i_z = \pi_{k_1}(z)$. 2. Compute the relevant coefficient: $v_z = f_{k_2}(z)$. Compute:

$$\begin{cases} \tau_1 = v_1 T_{i_1,1} + v_2 T_{i_2,1} + \ldots + v_c T_{i_c,1} \\ \tau_2 = v_1 T_{i_1,2} + v_2 T_{i_2,2} + \ldots + v_c T_{i_c,2} \\ \vdots \\ \tau_L = v_1 T_{i_1,L} + v_2 T_{i_2,L} + \ldots + v_c T_{i_c,L} \end{cases} \quad (1)$$

and

$$\begin{cases} \omega_1 = v_1 m_{i_1,1} + v_2 m_{i_2,1} + \ldots + v_c m_{i_c,1} \\ \omega_2 = v_1 m_{i_1,2} + v_2 m_{i_2,2} + \ldots + v_c m_{i_c,2} \\ \vdots \\ \omega_L = v_1 m_{i_1,L} + v_2 m_{i_2,L} + \ldots + v_c m_{i_c,L} \end{cases} \quad (2)$$

Output vector $(\tau_1, \ldots, \tau_L)$ and $(\omega_1, \ldots, \omega_L)$.

- *VerifProof*$(\rho, chal, sk_l) \rightarrow \{1, 0\}$ Let $sk_l = (a_l, b_l, c_l)$ and $chal = (c, k_1, k_2)$. For $1 \le z \le c$: 1. Compute the index of each sampled block: $i_z = \pi_{k_1}(z)$. 2. Compute the relevant coefficient: $v_z = f_{k_2}(z)$. Parse the proof to obtain τ and ω. Compute: $\tau = \tau_1 + a_l \tau_2 + \ldots + a_l^{L-1} \tau_L$, $\omega = \omega_1 + c_l \omega_2 + \ldots + c_l^{L-1} \omega_L$ and $\sigma = v_1 f_{b_l}(i_1) + \ldots + v_c f_{b_l}(i_c)$. If $\tau = \sigma + \omega$, then output 1. Otherwise output 0.
- *VerifCOM*$(\{m_{<i>}\}, \{i\}, c, \{COM_i\}) \rightarrow \{1, 0\}$: Let $mpk = (N, d, g, h)$. For each $i \in \{i_1, \ldots, i_c\}$ given in advance, the related commitment functions and file chunks can be revealed to public as following:
COM_i, d, $m_{<i>}$ where $COM_i = COM(m_{<i>}) = (g^{m_{<i>}} h^{H(i)})^e \bmod N$.
Following this, the validity of arbitrary chunks and its commitment can be verified.
For $i \in \{i_1, \ldots, i_c\}$, the verifier computes:
1. $s_i = g^{m_{<i>}} h^{H(i)} \bmod N$. 2. $t_i = (COM_{<i>})^d = (g^{m_{<i>}} h^{H(i)})^{ed} \bmod N = g^{m_{<i>}} h^{H(i)} \bmod N$. If $s_i = t_i$, the chunk $m_{<i>}$ and its commitment are matching.
Output 1, if and only if all pairs of chunks and commitments match completely. Otherwise, return 0.

C. THE DR-GPOS PROTOCOL

A DR-GPOS protocol is designed in four phases:

1) PRE-PROCESS

The TTP runs *KeyGen* and *PreGen*, and sends F, COM and Σ to the server S for storage. Then TTP distributes the secret key K_i to each group member. Following this, F, COM and Σ are removed from local storage.

2) CHALLENGE AND VERIFICATION

Every member U_i can independently generate a challenge *chal* and sends it to S. Then S executes *ProofGen* and sends the proof ρ to U_i. Finally, U_i checks the validity of ρ by executing *VerifProof*.

3) MALICIOUS-MEMBER DISTINCTION

When multiple members have different decisions, S is required to open the commitment functions to the public, hence, everyone can distinguish the dishonest member.

4) MALICIOUS-MEMBER REVOCATION

When a dishonest member is distinguished, TTP can revoke it, which means TTP removes the dishonest member from the group and informs all members that verification of the malicious one is invalid. Notice that the key of revoked member cannot be used to derive keys of other members or forge a valid tag and proof. The final three phases can be executed multiple times respectively to insure that both the server and group members are honest.

V. SECURITY ANALYSIS

In this section, we will introduce the security model and security proof of DR-GPOS scheme.

A. SECURITY MODEL

The security model is given by three games, which capture properties of member distinction, member revocation, and proof of storage. For simplicity, we utilize T_i and m_i to replace $T_{<i>}$ and $m_{<i>}$ respectively.

Game 1 *Member Distinction Game:*

- Setup: The adversary is provided with the security parameter 1^λ as input, and it outputs a pair of file chunks m_0, m_1 of the same length.
- Challenge: The challenger runs $PreGen(msk, m)$ and outputs b_0, b_1 where $COM(m_0) = b_0$ and $COM(m_1) = b_1$.
  Choose a random bit $r \leftarrow \{1, 0\}$, then give b_r and mpk to the adversary.
- Decision: The adversary runs $VerifCOM(mpk, b_r)$ and outputs 1 if b_r is the correct commitment of m_1, otherwise outputs 0.

The commitment function is valid (i.e. satisfy the property of binding) 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 of guessing the correct choice by is

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

Remarks: In Game 1 (which is proposed by Wang et al. [7]), we define the adversary's advantage as the absolute value of the probability of guessing the right choice minus 1/2. If the advantage of adversary is negligible, then the scheme is secure, which means:

$$\left| \frac{1}{2} \cdot \Pr[PrivK_A^{COM}(n, b_1) = 1] + \frac{1}{2} \cdot \Pr[PrivK_A^{COM}(n, b_0) = 0] - \frac{1}{2} \right| \leq negl(n).$$

Game 2 *Member Revocation Game:*

- Setup: The challenger performs $KeyGen(1^\lambda)$ to generate a set of keys $\{K_l\}_{l=1}^L$, and keeps them secret.
- Selective Revocation: The adversary queries keys adaptively, and the challenger sends these queried keys to the adversary and keeps other keys secret. Note that the number of queried keys must be less than L. After that, the adversary owns partial keys $\{K_{l_1}, \ldots, K_{l_r}\}$ which are the keys of revoked members, while the challenger preserves residual keys $\{K_l\}_{l=1}^L / \{K_{l_1}, \ldots, K_{l_r}\}$ privately.
- Forge: The adversary can obtain files $File = (m_1, m_2, \ldots, m_n)$, tags $\Sigma = (T_1, T_2, \ldots, T_n)$ and some keys $\{K_{l_1}, \ldots, K_{l_r}\}$. Then the adversary runs **PreGen** and **ProofGen** to generate a tag T_r' and proof ρ. Note that ρ comprises of T_r', where $T_r' \neq T_r$, and both of them are tags of chunk m_r.

If $VerifProof(K', chal, \rho) \rightarrow 1$, where $K' \in \{K_l\}_{l=1}^L / \{K_{l_1}, \ldots, K_{l_r}\}$, then the adversary wins the game. The probability that adversary wins the game is negligible if the scheme is against selective and collusion attacks. The probability must satisfy:

$$\Pr\left[\begin{array}{l|c} VerifProof(K', chal, \rho) \rightarrow 1 & PreGen(\{K_{l_1}, \ldots, K_{l_r}\}, m_r) \rightarrow T_r' \\ K' \in \{K_l\}_{l=1}^L / \{K_{l_1}, \ldots, K_{l_r}\} & PreGen(\{K_l\}_{l=1}^L, m_r) \rightarrow T_r \\ ProofGen(File, T_r') \rightarrow \rho & T_r' \neq T_r \end{array}\right] \leq negl(\lambda).$$

Game 3 *Storage Proof Game:*

- Setup: The challenger runs $KeyGen(1^\lambda)$ to generate a set of keys $\{K_l\}_{l=1}^L$, and keeps them secret.
- Query: The adversary can query adaptively. Select chunk m_i of a file and provide it to the challenger. It then executes $PreGen(\{K_l\}_{l=1}^L, m_i)$, and computes T_i which is then provided to the adversary. By executing these queries multiple times, all these chunks $F = (m_1, \ldots, m_n)$ can be stored by the adversary, together with tags $\Sigma = (T_1, \ldots, T_n)$.
- Challenge: The challenger generates a challenge *chal* and requests a proof of chunks $m_{i_1}, \ldots, m_{i_c}$, where $1 \leq i_c \leq n \wedge 1 \leq c \leq n$.
- Forge: The adversary computes a proof ρ determined by *chal* and returns ρ.

If $VerifProof(K', chal, \rho) \rightarrow 1$, where $K' \in \{K_l\}_{l=1}^L$, then the adversary wins the game.

The probability that the adversary wins the game is negligibly close to the probability that the adversary extracts those challenged file chunks. Intuitively, a verification tag T and proof ρ cannot be computed unless the adversary already possesses all the challenged chunks.

B. SECURITY PROOF

The security of our scheme is proven based on provable security theory. The modern security proofs take the reductionist approach rely on an assumption about the hardness of some mathematical problem in order to prove their security. Therefore, in this paper, we reduce breaking the proposed scheme to solving an underlying hard problem. DR-GPOS scheme is based on the security assumptions of DLP [12] and secure PRF [13].

Theorem 1: If f is a secure PRF and TTP is trusted, then under the DLP assumption, the DR-GPOS scheme can accurately distinguish a malicious member, guarantee secure revocation of the malicious member, and data possession in standard model.

Proof.

1) CORRECTNESS

For $1 \le l \le L$ and $(a_l, b_l, c_l) \leftarrow KeyGen(1^\lambda)$, the probability of failure is given as:

$$fail_{GPOS}(\lambda) := \Pr[VerifProof(K_l, chal, ProofGen \times(\{m_{<i>}\}_{1\le i\le n}, \{T_{<i>}\}_{1\le i\le n}, chal)) \neq 1].$$

Since f is a secure PRF, then the calculation of tags can be expressed as $AT = X$, where A is the matrix of a, T is the matrix of Tag, and X is a random matrix. By the assumption of L-XAC, we can obtain: $fail_{GPOS}(\lambda) \le \frac{L(L-1)}{2q}$.

2) SOUNDNESS

For $1 \le l \le L$ and arbitrary invalid proof $\rho' \notin \{ProofGen(File, \Sigma, chal)\}$, the probability that ρ' passes the verification is given as:

$$Adv^{inval}_{GPOS}(\lambda) := \max_l \Pr[VerifProof(K_l, chal, \rho') = 1 | K_l \leftarrow KeyGen(1^\lambda)].$$

As the tags which are used to generate a proof are random, hence we can conclude: $Adv^{inval}_{GPOS}(\lambda) \le \frac{1}{q^c}$, where c is the number of challenged chunks.

3) DISTINCTION OF MALICIOUS-MEMBER

We utilize the reduction method to prove the property of distinction, which is realized by using Game 1 of DR-GPOS. In order to to distinguish a malicious-member, it needs to be established that the verifier cannot lie and repudiate, i.e. the verification result must be bound to the original data and cannot be altered.

An adversary A can be constructed, if there exists an adversary A^* wins the Distinction Game, to break the Pedersen commitment function. Game 1 can be executed as following:

- Setup: A^* is given input security parameter 1^λ, and it outputs a pair of data chunks m_0, m_1 of the same length. A^* then gives m_0 and m_1 to A.
- Challenge: A runs $PreGen(msk, m_i)$ 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 mpk to A^*.
- Forge: A^* executes $VerifCOM(mpk, b_r)$ and outputs 1 or 0.

If the probability satisfies:

$$|\Pr[PrivK^{COM}_A(n, b_0) = 1] - 1| \le negl(n)$$

or

$$|\Pr[PrivK^{COM}_A(n, b_1) = 0] - 1| \le negl(n),$$

then the adversary wins. This means that, A^* can forge data m_0 that can commit another data m_1, for the reason that m_0 and m_1 are given by A^* in advance. By Pedersen commitment function, A can compute commitment which satisfies $COM(m_0, ID_0) = COM(m_1, ID_1)$ where $m_0, m_1 \in Z_q$ and $m_0 \neq m_1$. Obviously, $ID_0 \neq ID_1 \bmod N$ and $\log_g h = \frac{m_0 - m_1}{H(ID_0) - H(ID_1)} \bmod N$. By the assumption of DLP, $\log_g h \bmod N$ cannot be found except with negligible probability in $|N|$. Thus, it fulfills the property of non-repudiation, i.e. a malicious member can be distinguished if it lies or repudiates a valid decision.

4) REVOCATION OF MALICIOUS-MEMBER

Intuitively, secure revocation means that the operation of revocation does not affect the normal execution of system, and the revoked member cannot derive keys of others or forge a valid tag and proof by utilizing his secret key. Run Game 2 as follows:

- Setup: The challenger runs $KeyGen(1^\lambda)$ to generate a set of secret keys $\{K_l\}_{l=1}^L$, and keeps them private.
- Selective Revocation: After the adversary queries keys adaptively, the adversary owns partial keys $\{K_{l_1}, \ldots, K_{l_r}\}$, which are the keys of revoked members, while the challenger preserves residual keys $\{K_l\}_{l=1}^L / \{K_{l_1}, \ldots, K_{l_r}\}$ privately.
- Forge: The adversary uses the known information: $File = (m_1, m_2, \ldots, m_n)$, $\Sigma = (T_1, T_2, \ldots, T_n)$ and $\{K_{l_1}, \ldots, K_{l_r}\}$ to run **PreGen** and **ProofGen** and generate a tag T'_r and a proof ρ. Notice that ρ is consisted of T'_r, where $T'_r \neq T_r$, and both them are tags of chunk m_r.

By securing against substitution attacks of L-XAC, the probability of secure revocation $Adv^{rev}_{GPOS}(\lambda)$ is

$$\max_{File,\Sigma} \Pr\left[\begin{array}{l|c} VerifProof(K', chal, \rho) = 1 & T'_r \leftarrow PreGen(\{K_{l_1}, \ldots, K_{l_r}\}, m_r) \\ K' \leftarrow \{K_l\}_{l=1}^L / \{K_{l_1}, \ldots, K_{l_r}\} & T_r \leftarrow PreGen(\{K_l\}_{l=1}^L, m_r) \\ \rho \leftarrow ProofGen(File, T'_r) & T'_r \neq T_r \end{array}\right] = Adv^{sub}_{XAC}(\lambda) \le 2 \cdot \frac{L-1}{q}.$$

The above probability is negligible because $q = 2^\lambda$.

5) PROOF OF STORAGE

The parameters have been simplified for the proof as: set all v_z to 1, and the matrix is represented by capital letters. Then we utilize PRF $f(x)$ and a random value r to run Game 3 in real environment and ideal environment respectively.

TABLE 1. The comparison of the performance of PDP and POR schemes.

	E-/S-PDP [4]	CPOR [14]	POSD [15]	Oruta [16]	GPoS [6]	GPDP	NRPDP	DR-GPOS
Data possession	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Private verification	Yes	Yes	No	No	Yes	Yes	Yes	Yes
Applied to a group	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Support deduplication	No	No	Yes	No	No	Yes	No	Yes
Same key in pre-processing	No	No	No	No	No	No	No	No
Same key in verification	No	No	No	Yes	No	No	No	No
Secure Member-revocation	Yes	Yes	No	Yes	No	Yes	Yes	Yes
Pinpoint a dishonest member	No	No	Yes	Yes	No	No	No	Yes
Pre-processing	$O(L \cdot n)$	$O(L \cdot n)$	$O(L \cdot n)$	$O(n)$	$O(n)$	$O(n)$	$O(L \cdot n)$	$O(n)$
Generating a proof	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(c)$	$O(c)$
Verifying a proof	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$	$O(1)$
Communication cost	$O(1)$	$O(l)$	$O(l)$	$O(l)$	$O(l)$	$O(L)$	$O(l)$	$O(L)$
Storage cost	$O(n + w \cdot L)$	$O(n+w \cdot L)$	$O(n + w)$	$O(n + w \cdot L)$	$O(n + w \cdot L)$	$O(n + w)$	$O(n + w \cdot L)$	$O(n + w)$

Note: n is the total number of file blocks or chunks, c is the number of sampling blocks or chunks, w is the extra storage on the server such as verification tags and commitment functions, l is the number of blocks in each chunk, L is the number of members in a group, and $L = l$ in the DR-GPOS and GPDP scheme.

a: IN REAL ENVIRONMENT

- Setup: The challenger runs $KeyGen(1^\lambda)$ to generate a set of keys $\{K_l\}_{l=1}^{L}$, and keeps them secret.
- Query: The adversary can query adaptively. A file chunk m_i is selected and provided to the challenger, which then executes $PreGen(\{K_l\}_{l=1}^{L}, m_i)$ to compute the tag T_i as follows: compute $T = A^{-1}CM + A^{-1}B$, where M is the matrix of file blocks, A and C are the matrices of secret keys, and $B = [f_{b_1}(i), f_{b_2}(i), \ldots, f_{b_L}(i)]^T$. Note that $T_i = T^T$ where the vector T_i is the transpose of matrix T. The challenger then sends T_i to the adversary. This query process can be executed arbitrary times, after which, the adversary owns and stores all the file data $File = (m_1, \ldots, m_n)$ and the corresponding tags $\Sigma = (T_1, \ldots, T_n)$.
- Challenge: The challenger generates a challenge $chal$ and requests a proof of chunks $m_{i_1}, \ldots, m_{i_c}$, where $1 \leq i_c \leq n \wedge 1 \leq c \leq n$ from the adversary.
- Forge: The adversary computes a proof ρ determined by $chal$ and returns ρ to the challenger.

b: IN IDEAL ENVIRONMENT

We utilize the random number to replace the result of the pseudo random function, hence $B = [r_1, r_2, .., r_L]^T$. All the random numbers need be recorded, and will be used in verification. In an ideal environment, Game 3 is executed similar to real environments.

If the adversary passes the verification process when M was changed or forged, then that the adversary can successfully forge a T' such that $T' = A^{-1}CM' + A^{-1}B$, where M' is the changed M in the ideal environment. Due to the fact that A and C are matrixes of random secret keys and B is the random matrix, the inequation can be represented as follows: $A_{GPOS}^{ideal} = \Pr[T'|T' = A^{-1}CM' + A^{-1}B] = \Pr[T'|T' = R_1M' + R_2] \leq \frac{1}{2^{\lambda \cdot L}}$, where R_1 and R_2 are matrixes of random numbers. Based on the assumption: f is a secure PRF, therefore the adversary cannot distinguish whether the protocol runs in the ideal environment or the real environment. As a consequence, the probability that the adversary forge a valid tag is $A_{GPOS}^{real} \cong A_{GPOS}^{ideal} \leq \frac{1}{2^{\lambda \cdot L}}$.

VI. IMPLEMENTATION AND EVALUATION

The evaluation has been done using Baidu Cloud Servers, where all experimental data is stored. The server is configured with 16GB memory and 4-core processors with multi-threading support. Algorithms are implemented using OpenSSL version 0.9.8b with a modulus N of size 1024 bits on Red Hat Enterprise Linux AS release 4. Disk I/O performance is measured with Samsung 840 Pro (MZ-7PD128BW) 120GB Solid state Disk. In order to minimize the error margin, all experimental results are obtained by repeated testing and comparison.

In DR-GPOS scheme, the sampling method used is uniform with the classic S-PDP scheme [4]. From the inequation:

$$1 - \left(\frac{n-t}{n}\right)^c \leq P_X \leq 1 - \left(\frac{n-c+1-t}{n-c+1}\right)^c,$$

it is easy to see that the data integrity can be detected with a high probability by asking proof of constant number of file chunks. Denote $t = 1\%$ of n, and P_X is at least 99%, then the number of challenged blocks is 460.

In order to standardise the performance of experiments, the size of file data chunk/block is 256KB in the absence of special instructions.

A. THE COMPARISON OF VARIOUS PDP AND POR SCHEMES

The comparison of various schemes is shown in Table 1. We can see from the table that DR-GPOS is the only one that can accurately distinguish a malicious member from a private verification group, as well as support secure revocation of malicious-members and data deduplication.

Note that, when S-PDP and CPOR are applied to groups, each group member runs S-PDP or CPOR scheme by using their secret keys. Therefore, each member is independent

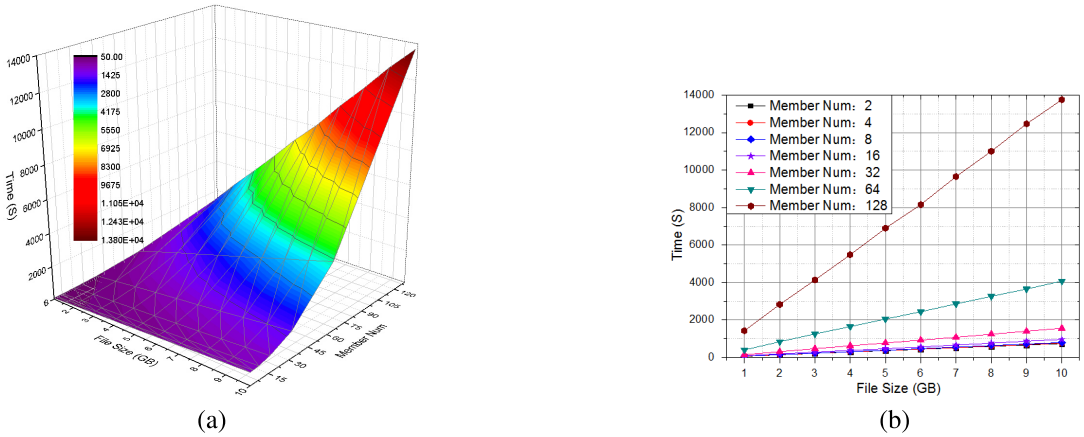


FIGURE 2. The efficiency evaluation of pre-process of DR-GPOS with different member number. (a) The performance in 3D surface (b) The performance in 2D line.

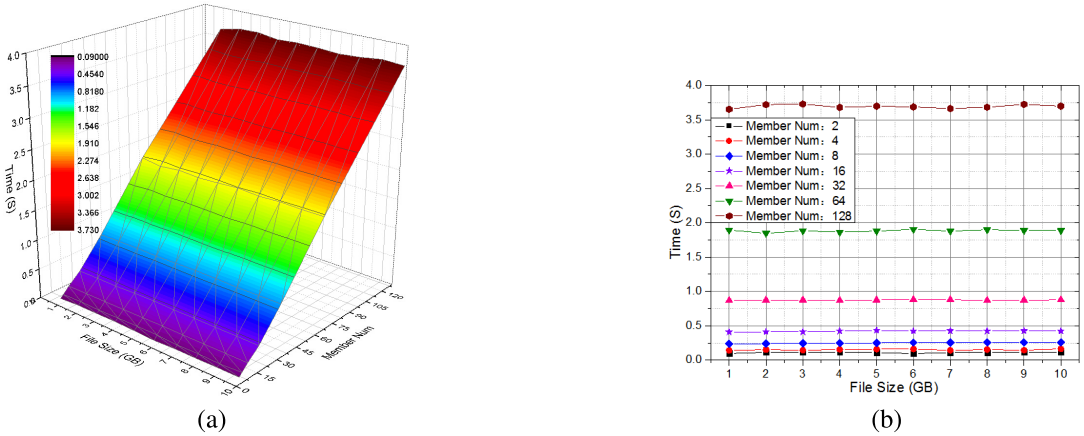


FIGURE 3. The efficiency evaluation of challenge and verification with varying member size. (a) The performance in 3D surface (b) The performance in 2D line.

and the selective opening attacks (i.e. a malicious member colludes with the server.) are avoided. Hence, the system can support secure revocation of malicious members.

B. THE EFFICIENCY EVALUATION OF DR-GPOS

Fig. 2 and Fig. 3 show the performance of DR-GPOS with different number of group members. To improve understanding, they are presented in 3D and 2D format respectively.

In the pre-process phase, TTP utilizes all the secret keys $\{K_l\}_{l=1}^{L}$ and master secret key *msk* to preprocess the file data, and generates verification tags and commitment functions. Then TTP sends the file data, tags and commitments to the server, distributes the secret keys to each member respectively through a secure channel, and finally deletes all the local storage except the master secret key. In experiments, we measure the computing cost of pre-process, which includes the time of generating keys, tags and commitments, as well as the time of necessary I/O, but without the time of data transmission.

We can see from Fig. 2 that the computing time is linearly increased with the increase of the file size. For fixed sized file chunks, the total file size is greater, hence individual file chunks are bigger, which leads to larger computation costs. Moreover, with the increase in members, the computing time rises. Moreover, with the increase in members, the computing time rises. This is due to the fact that generation of tags requires secret keys of all members. Hence, as the members grow, the computation costs grow with more keys. Though the computing cost is fairly large when the total file size or the number of group member is large, the computation of pre-process is a one-time effort, and the results can be used repeatedly. Thus, the experimental results are acceptable.

In the phase of challenge and verification, each group member U_k can start a challenge *chal* to the server, and the server generates a proof ρ based on *chal* and sends it to U_k, finally, U_k utilizes its secret key K_k to verify ρ. In the process of evaluation, we measure the computation time of proof generation, proof verification, and necessary I/O time. Analogously, we ignore the time of transmission and the I/O time of data seeking and comparison.

As Fig. 3 shows, for the fixed file chunks size and group member number, the time of challenge and verification is constant, irrespective of the file size. Thus, the number of challenged chunks is fixed at 460 in experiments. Therefore, the computing time of generating and verifying one proof

is not dependent on the total file size. However, with the increase in members, the number of file blocks (each file chunk includes L blocks.) rises, accordingly, the computing cost increases. We can observe from the figure, when the number of group member is relatively small (i.e. under 20), the time of one challenge and verification remains under 0.5 seconds. When the number of members is relatively large (i.e. about 128), the time is about 3.7 seconds. This is reasonably efficient in a real word application.

C. THE EFFICIENCY EVALUATION OF DISTINCTION

The efficiency of distinction is directly relevant to the computational efficiency of commitment functions. Therefore, we measure the performance of commitment functions in this section.

In DR-GPOS scheme, each chunk corresponds to a commitment. In the phase of pre-processing, TTP utilizes *msk* to run **PreGen** and compute commitment, and sends them to the server. In the phase of public commitment verification, anyone can use the *mpk* to run **VerifCOM** and verify the commitment, and accordingly everyone can determine who is dishonest. In experiments, we only measure the time of commitment generation and verification, and ignore the time of information transmission.

Fig. 4 shows the performance of generating and verifying commitments. As we can observe, the generation time is linearly related to the total file size. This is due to the fact that the number of file chunks increases with the increase in file size and fixed chunk size, therefore the time of generation is linearly increased. On the other hand, for a file with the same size, with the increase in chunk size, the number of file chunks is decreased linearly, and consequently the time of generation is reduced. Though the generation time is growing linearly related to the file size, it is a one-time calculation and reusable.

In the phase of commitment verification, the number of challenged chunks can be given by the group members. In experiments, we adopt the worst-case performance, which means we measure the verification time of 460 chunks and commitments each time.

As we can see from Fig. 4, the time of verification is constant with the fixed chunk size. In this phase, not all verifiers have $\phi(N)$, and thus these verifiers have to exponentiate the whole data chunks which can be quite time consuming. With the increase in chunk size, the calculation cost is unavoidably incremental, and therefore the time cost also rises. In the meantime, for the same chunk size, the time of each verification is constant, because the number of chunks for each verification is fixed at 460. Thus, it has no relation with the total file size. Though the time cost of verification is more consuming than private verification, the commitment verification is infrequent, even if malicious members exist during each authentication phase, the computational time is acceptable.

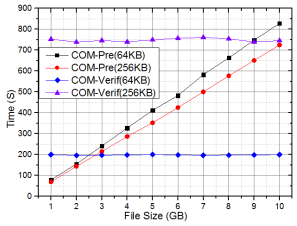


FIGURE 4. The performance of commitment generation and verification.

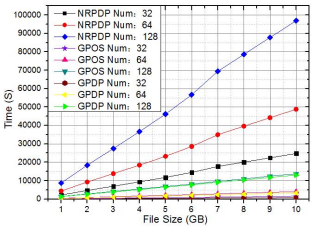


FIGURE 5. The comparison between group NRPDP and GPDP.

On the whole, it is efficient and practical enough for distinguishing the dishonest group members, so that appropriate parameters can be chosen.

D. THE COMPARISON WITH GROUP NRPDP AND GPDP

The performances of pre-process of DR-GPOS, group NRPDP and GPDP [8] are shown in **Fig. 5**. Similar to the previous analysis, when NRPDP is applied in group, each group member running NRPDP scheme use its own secret key.

It can be observed, for fixed total file size and chunk/block size (256KB), NRPDP is more time-consuming than DR-GPOS and GPDP in pre-processing phase. The reason is in NRPDP scheme, each tag generation needs an exponent arithmetic numerous multiplication steps, while in DR-GPOS and GPDP scheme, multiplication and addition operations are only needed to generate tags. However, the computational cost of DR-GPOS is slightly higher than GPDP because of the added computational cost of commitment functions. Even so, the increased computational cost is infinitesimally small than the original computation of pre-process.

On the other hand, the time is linearly related to the total file size in all three schemes. Analogously, the number of file chunks/blocks with fixed size is increasing with the growth of file size, therefore the time of generation linearly increases. Additionally, with the increase in the members, the computing time of all three rises. Even then, the DR-GPOS and GPDP are more efficient than NRPDP.

As for the verification time, the three private verification schemes have minor differences, while the efficiency of the three is similar.

E. THE COMPARISON BETWEEN PRIVATE VERIFICATION AND PUBLIC VERIFICATION

The benefit of public verification is the exposure of dishonest members. However, the public schemes tend to be more time-consuming. Fig. 6 shows a generalized comparison of the

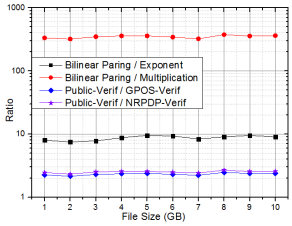


FIGURE 6. The comparison between public and private verification.

public verification using bilinear pairing and private verification of DR-GPOS.

In experiments, we utilize the the Pairing-Based Cryptography (PBC) library version 0.5.14, and use the modulus N of size 1024 bits.

It can be observed from Fig. 6 that the ratio of one bilinear pairing against one multiplication operation is approximately 350, regardless of file size. Moreover, the ratio against one exponent operation is approximately 9. Hence, the bilinear operations tend to be inefficient. For the verification performance, the number of challenged chunks is set to 460 in both public and private schemes. We only observe the verification on time. The ratio of onetime public verification against private is approximately 2.5. Although it will not affect much in single verification, but in real world applications, it will be done thousands of times, which will make it significant. We can conclude that public schemes using bilinear pairing are inefficient in real world systems.

F. THE STORAGE AND COMMUNICATION COST

1) STORAGE COST

The storage cost of DR-GPOS is only relevant to the total file size and security parameter, and is irrespective of the number of group members. For instance, each file chunk of 4KB corresponds with both a verification tag and a commitment of 1024 bits; this means the total storage in cloud increases by 6.25% (equal split of tags and commitments). It is acceptable because we can reduce the extra storage by enlarging the chunk size. Though it requires more extra storage than GPDP, it has more advantages compared to the most other schemes. The storage cost on each group member is $O(1)$, which is only related to the security parameters.

2) COMMUNICATION COST

In the phase of challenge and verification, the required bandwidth between client and server is $O(1)$, this is because the challenge and the proof are all constant (the *chal* is 384B, and the ρ is $256\times L$B). In the phase of commitment verification, the server requires a larger bandwidth (each commitment is 1024 bits, and each chunk is 256KB) to reveal the original data and related commitments. However, these circumstances rarely appear in normal cloud systems. The availability of high bandwidth also makes it acceptable in practical systems.

VII. RELATED WORK

In order to guarantee data integrity in cloud storage, many approaches and schemes have been proposed, e.g. [2], [17], [18]. Each type of schemes has its own advantages and disadvantages, and can solve different security problems. According to the definition in S-PDP scheme, only verifiers who hold the private key can verify the data integrity in a private scheme, while others cannot. Comparatively, the verification keys are public in a public verification scheme, and anyone can use these keys to verify the data. Generally speaking, private protocols are more efficient, but the verification results are only known by the verifiers. Meanwhile, the public schemes are usually inefficient because of the usage of bilinear pairing, but everyone can identify the verification results. We categorized the different types of schemes and described the properties the proposed scheme can satisfy.

A. PRIVATE VERIFICATION SCHEMES

Although private verification schemes are highly efficient, the data deduplication, malicious-member distinction and revocation cannot be guaranteed when the private schemes are applied in group application.

1) SCHEMES BASED ON RSA

Ateniese et al. [4], [19] proposed the first sampling model for PDP without requiring the server retrieval 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 use RSA scheme to keep the tags private. After generating a proof, they verified it with the secret key of RSA. The schemes are all based on RSA cryptography, which has the drawback of exponential calculations.

The RSA method has the property of homomorphism, and can be used to construct the detection mechanism of data integrity. The simplified algorithm is as follows:

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: $T_i = (g^{m_i})^d \mod N$ 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 $m_{i_k} (1 \le k \le c)$ and tags T_{i_k} based on the challenge.
3. The server compute $T = \prod_{k=1}^{c} T_{i_k}$ and $\rho = g^{\sum_{k=1}^{c} m_{i_k}} \mod N$, 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 such as [7], [20] utilize the RSA method.

2) SCHEMES BASED ON SENTINELS

Juels and Kaliski [5], [21] introduced the notion of proof of retrievability (POR), which is function-similar to PDP. The POR scheme uses sentinels hidden among regular file blocks to detect modified data, so that it can only be applied to encrypted files and only perform a limited number of queries, which equals to the number of sentinels.

The simplified process is 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 positions 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 have abandoned this approach.

3) SCHEMES BASED ON SYMMETRIC CRYPTOGRAPHY

Shacham and Waters [14] proposed a new notion of Compact POR (CPOR), which utilizes symmetric cryptography and homomorphic properties to combine multiple authenticator values into a small one and minimizes the communication cost. It utilizes exponential calculation for verification tags so that it increases computation cost. Dodis et al. [22] 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 RO (Random Oracle) and the first bounded-use scheme with information-theoretic security. It is a theoretical scheme and has not been implemented.

B. PUBLIC AUDITING SCHEMES

Public verification scheme can easily distinguish a malicious member. However, the public verification schemes are generally constructed based on the bilinear pairing or third-party (TP) verification, which makes the computational efficiency relatively low.

1) SCHEMES BASED ON BILINEAR PAIRING

This approach is generally used for public verification which can achieve zero-storage on client. The drawback is inefficient computation of bilinear pairing and no privacy because anyone can obtain and verify the data of other clients.

Hanser and Slamanig [23] proposed the first simultaneous private and public verification PDP scheme 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. Zhu et al. [24]–[26] 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. The drawback is that the operation of bilinear pairing is very time-consuming, thus these schemes of public verification are always not very efficient.

2) SCHEMES BASED ON TP

TP is utilized to represent the cloud client to verify the possession of data. It supports the public verification and usually applies on encrypted data. Wang et al. presented public auditing schemes [27]–[30] based on TP which usually require extra cost of client side.

In 2014, a new notion of Outsourced Proofs of Retrievability (OPOR) [31] was proposed. The OPOR scheme, which is named Fortress, utilizes an external party which is untrusted to conduct a POR scheme and interact with the server on behalf of the client. Though the Fortress scheme can protect all these three parties synchronously, the drawbacks are obvious. 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 inefficient.

C. GROUP AUDITING SCHEMES

In some special scenarios, different clients in a group may share a same file, and the system needs more than one client to verify this file data.

Wang et al. [16], [32] proposed the privacy-preserving schemes with private and public auditing respectively. In their mechanism, the identity of data signer can be kept private to the auditors. Soon afterwards, Wang et al. [33] introduced a new public auditing scheme which can efficiently revoke a user, and the revoked user can not utilize his secret key to forge a valid proof. However, the shortcoming is taking no account of collusion. Wang et al. [6] proposed a notion of Group-oriented Proofs of Storage (GPOS), which can limit the communication bandwidth in a fixed size. But all these group schemes do not involve the issue of deduplication. To solve this problem, Wang et al. [8] proposed a group PDP (GPDP) scheme which can efficiently guarantee data possession with deduplication, as well as against selective opening attacks of a malicious party. Nevertheless, how to accurately and efficiently distinguish and revoke a malicious member in a private verification group is still a unresolved problem.

VIII. CONCLUSION

In this paper, we give a DR-GPOS scheme, which is based on on matrix calculation, pseudo-random function, and commitment function. When malicious members are involved in a group and repudiate a valid proof, DR-GPOS can efficiently distinguish the malicious one and securely revoke its access.

In terms of functionality, DR-GPOS can accurately distinguish the dishonest members in a group, as well as guarantee the integrity and deduplication of the outsourced data. From

a security perspective, DR-GPOS can support the revocation of a dishonest member and resist the attacks from revoked members (e.g. forge proof by colluding with server).

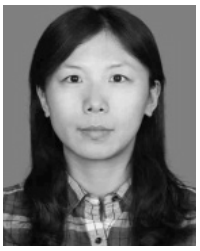
We give the security analysis in the standard model, and have implemented the scheme or realtime cloud servers to evaluate the performance. Evaluation shows that DR-GPOS not only efficient than other schemes, but also more practical for application of real world.

REFERENCES

[1] X. Huang and X. Du, "Achieving data privacy on hybrid cloud," *Secur. Commun. Netw.*, vol. 8, no. 18, pp. 3771–3781, 2015.
[2] A. Pawloski, L. Wu, X. Du, and L. Qian, "A practical approach to the attestation of computational integrity in hybrid cloud," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Garden Grove, CA, USA, Feb. 2015, pp. 72–76.
[3] J. Li, Z. Guan, X. Du, Z. Zhang, and Z. Zhou, "A low-latency secure data outsourcing scheme for cloud-WSN," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, San Francisco, CA, USA, Mar. 2017, pp. 1–6.
[4] G. Ateniese *et al.*, "Provable data possession at untrusted stores," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 598–609.
[5] A. Juels and B. S. Kaliski, Jr., "PORs: Proofs of retrievability for large files," in *Proc. 14th ACM Conf. Comput. Commun. Secur.*, 2007, pp. 584–597.
[6] Y. Wang, Q. Wu, B. Qin, X. Chen, X. Huang, and Y. Zhou, "Group-oriented proofs of storage," in *Proc. ACM Symp. Inf.*, 2015, pp. 73–84.
[7] H. Wang, L. Zhu, C. Xu, and Y. Lilong, "A universal method for realizing non-repudiable provable data possession in cloud storage," *Secur. Commun. Netw.*, vol. 9, no. 14, pp. 2291–2301, Sep. 2016.
[8] H. Wang, L. Zhu, and Y. Lilong, "Group provable data possession with deduplication in cloud storage," *J. Softw.*, vol. 27, no. 6, pp. 1417–1431, 2016.
[9] S. Agrawal and D. Boneh, "Homomorphic MACs: MAC-based integrity for network coding," in *Applied Cryptography and Network Security*. Berlin, Germany: Springer-Verlag, 2009, pp. 292–305.
[10] S. Fehr, D. Hofheinz, E. Kiltz, and H. Wee, "Encryption schemes secure against chosen-ciphertext selective opening attacks," in *Advances in Cryptology—EUROCRYPT* (Lecture Notes in Computer Science), vol. 6110. Berlin, Germany: Springer-Verlag, 2010, pp. 381–402.
[11] T. P. Pedersen, "Non-interactive and information-theoretic secure verifiable secret sharing," in *Advances in Cryptology—CRYPTO*. Berlin, Germany: Springer-Verlag, 1992, pp. 129–140.
[12] K. S. McCurley, "The discrete logarithm problem," in *Proc. Symp. Appl. Math.*, vol. 42, 1990, pp. 49–74.
[13] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," *J. ACM*, vol. 33, no. 4, pp. 792–807, Aug. 1986.
[14] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Advances in Cryptology—ASIACRYPT*. Berlin, Germany: Springer-Verlag, 2008, pp. 90–107.
[15] Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proc. 2nd ACM Conf. Data Appl. Secur. Privacy*, 2012, pp. 1–12.
[16] B. Wang, B. Li, and H. Li, "Oruta: Privacy-preserving public auditing for shared data in the cloud," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2012, pp. 295–302.
[17] R. Houlihan and X. Du, "An effective auditing scheme for cloud computing," in *Proc. IEEE Global Commun. Conf. (GLOBECOM)*, Anaheim, CA, USA, Dec. 2012, pp. 1599–1604. doi: 10.1109/GLOCOM.2012.6503342.
[18] J. Li, J. Li, D. Xie, and Z. Cai, "Secure auditing and deduplicating data in cloud," *IEEE Trans. Comput.*, vol. 65, no. 8, pp. 2386–2396, Aug. 2016. doi: 10.1109/TC.2015.2389960.
[19] G. Ateniese *et al.*, "Remote data checking using provable data possession," *ACM Trans. Inf. Syst. Secur.*, vol. 14, no. 1, pp. 1165–1182, 2011.
[20] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proc. 16th ACM Conf. Comput. Commun. Secur.*, 2009, pp. 213–222.
[21] A. Juels, B. S. Kaliski, Jr., K. D. Bowers, and A. M. Oprea, "Proof of retrievability for archived files," U.S. Patent 8 381 062, Feb. 19, 2013.
[22] Y. Dodis, S. Vadhan, and D. Wichs, "Proofs of retrievability via hardness amplification," in *Theory of Cryptography*. Berlin, Germany: Springer-Verlag, 2009, pp. 109–127.
[23] C. Hanser and D. Slamanig, "Efficient simultaneous privately and publicly verifiable robust provable data possession from elliptic curves," in *Proc. Int. Conf. Secur. Cryptogr. (SECRYPT)*, Reykjavík, Iceland, Jul. 2013, pp. 15–26.
[24] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu, and S. S. Yau, "Efficient provable data possession for hybrid clouds," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 756–758.
[25] Y. Zhu, H. X. Hu, G.-J. Ahn, and M. Yu, "Cooperative provable data possession for integrity verification in multicloud storage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 12, pp. 2231–2244, Dec. 2012.
[26] 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," *IEEE Trans. Services Comput.*, vol. 6, no. 2, pp. 227–238, Apr./Jun. 2013.
[27] 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*. Berlin, Germany: Springer-Verlag, 2009, pp. 355–370.
[28] C. Wang, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for data storage security in cloud computing," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.
[29] C. Wang, Q. Wang, K. Ren, N. Cao, and W. Lou, "Toward secure and dependable storage services in cloud computing," *IEEE Trans. Services Comput.*, vol. 5, no. 2, pp. 220–232, Apr./Jun. 2012.
[30] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for secure cloud storage," *IEEE Trans. Comput.*, vol. 62, no. 2, pp. 362–375, Feb. 2013.
[31] F. Armknecht, J.-M. Bohli, G. O. Karame, Z. Liu, and C. A. Reuter, "Outsourced proofs of retrievability," in *Proc. ACM SIGSAC Conf. Comput. Commun. Secur.*, 2014, pp. 831–843.
[32] B. Wang, B. Li, and H. Li, "Knox: Privacy-preserving auditing for shared data with large groups in the cloud," in *Applied Cryptography and Network Security*, 2012, pp. 507–525.
[33] B. Wang, B. Li, and H. Li, "Panda: Public auditing for shared data with efficient user revocation in the cloud," *IEEE Trans. Serv. Comput.*, vol. 8, no. 1, pp. 92–106, Jan./Feb. 2015.

LIEHUANG ZHU received the Ph.D. degree in computer science from the Beijing Institute of Technology, Beijing, China, in 2004, where he is currently a professor with the School of Computer Science and Technology. His research interests include security protocol analysis and design, group key exchange protocol, wireless sensor networks, and cloud computing.

HONGYUAN WANG received the bachelor's and Ph.D. degrees in computer science from the Beijing Institute of Technology, in 2011 and 2017, respectively. She is interested in information security and cloud security.



CHANG XU received the bachelor's and master's degrees from the School of Computer Science and Technology, Jilin University, in 2005 and 2008, respectively, and the Ph.D. degree in computer science from Beihang University, in 2013, where she is currently an Associate Professor with the School of Computer Science and Technology. Her research interests include security and privacy in VANET, and big data security.

KASHIF SHARIF received the M.S. degree in information technology, in 2004, and the Ph.D. degree in computing and informatics from the University of North Carolina at Charlotte, Charlotte, NC, USA, in 2012. He is currently an Associate Professor with the Beijing Institute of Technology, China. His research interests include wireless and sensor networks, network simulation systems, software-defined and data center networking, ICN, and the Internet of Things. He is a member of the ACM.

RONGXING LU received the Ph.D. degree from the Department of Electrical and Computer Engineering, University of Waterloo, Canada, in 2012. He was a Postdoctoral Fellow with the University of Waterloo, from 2012 to 2013. He was an Assistant Professor with the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore, from 2013 to 2016. He has been an Assistant Professor with the Faculty of Computer Science, University of New Brunswick, Canada, since 2016. His research interests include applied cryptography, privacy-enhancing technologies, and the IoT-Big Data security and privacy. He currently serves as the Secretary of the IEEE ComSoc CIS-TC. He is currently a Senior Member of the IEEE Communications Society. He received the most prestigious Governor General's Gold Medal and the 8th IEEE Communications Society (ComSoc) Asia Pacific Outstanding Young Researcher Award, in 2013.