Group-oriented Proofs of Storage

Yujue Wang
1. School of Computer
Wuhan University, China
2. City University of Hong
Kong, Hong Kong S.A.R.
yjwang@whu.edu.cn

Qianhong Wu
School of Electronic and
Information Engineering
Beihang University
Beijing, China
qianhong.wu@buaa.edu.cn

Bo Qin
School of Information
Renmin University of China
Beijing, China
bo.qin@ruc.edu.cn

Xiaofeng Chen
State Key Laboratory of
Integrated Service Networks
Xidian University
Xi'an, China
xfchen@xidian.edu.cn

Xinyi Huang
School of Mathematics and
Computer Science
Fujian Normal University
Fuzhou, China
xyhuang81@gmail.com

Yunya Zhou
School of Electronic and
Information Engineering
Beihang University
Beijing, China
maomaozyy@163.com

ABSTRACT

We introduces and formalizes the notion of *group-oriented proofs of storage* (GPoS). In GPoS, each file owner, after being authorized as a member by a group manager, can outsource files to a group storage account maintained by an untrusted party, for example, a cloud storage server, while anyone can efficiently verify the integrity of the remotely stored files without seeing the files. The file owner's identity privacy is preserved against the cloud server while the group manager can trace the one who outsourced any suspicious file for liability investigation. By novelly identifying and exploiting several useful properties, that is, homomorphic composability and homomorphic verifiability in some signatures, we propose a generic GPoS construction relying on the security of the underlying signature scheme and the hardness of the computational Diffie–Hellman (CDH) problem. Following the generic construction, we instantiate a concrete GPoS scheme with the well-known Boneh–Boyen short signature. By leveraging the polynomial commitment technique, the proposed GPoS proposal is optimized with constant-size bandwidth consumption in proof of storage by the cloud server. Theoretical analyses and comparisons show that our GPoS proposal is advantageous over existing PoS-like schemes in user privacy, public audibility and/or performance in a multi-user setting.

Categories and Subject Descriptors

E.5 [**Data**]: Files

ASIA CCS'15, April 14–17, 2015, Singapore, Singapore.

http://dx.doi.org/10.1145/2714576.2714630 .

Keywords

Cloud storage, Proofs of Storage, Provable Data Possession, Proof of Retrievability, Public auditability

1. INTRODUCTION

Cloud computing provides convenient storage services to the clients and is widely believed to be a promising technique to reduce the clients' local hardware and software maintenance burden of large-scale data storage [6]. However, this remote storage paradigm also brings security concerns about integrity and privacy over the outsourced files [25, 7]. To address the integrity concern, a number of cryptographic concepts have been introduced, for example, Proofs of Storage [3], Provable Data Possession (PDP) [1] and Proofs of Retrievability (PoR) [11].

Although PoS/PDP/PoR have received considerable attentions, most existing proposals, for example, [2, 9, 11, 13, 15, 20, 22, 27], cannot well fit in some real-world applications. Consider a company purchases remote storage services from some cloud storage provider (CSP) for storing files. The employees should be authorized (e.g., by the IT department of the company) in a way such that they can upload files to the company's account maintained by the cloud storage server. It is easy to see that some specific information (e.g., identity) of the file owner should be embedded into the outsourced version of the file so that the company manager can know who outsourced the file if necessary. Otherwise, some dishonest employee may misuse/abuse his/her file-outsourcing capability and even worse, unauthorized outsiders may also upload malicious data to the company's cloud storage account without being caught. Also, the file uploading information may be used to count up the employee's workload, which is usually taken as a critical factor for evaluating employees' performance. However, it is usually the case that the company or the employees would not like to let the cloud storage server know the identity of the file owner who outsources the file.

1.1 Our Work

Motivated by the above application, we introduce and formalize a system of *group-oriented proof of storage* equipped with a number of enjoyable functionalities: (1) A trusted

group manager issues secret keys associated with the members' unique identities. Every member can validate his/her secret key from the group manager. The members do not need to share any secret parameters among them. (2) Each group member can locally process the files with the issued secret key, without any further help from the group manager. The produced meta-data in the processed file is aggregatable, which implies economic bandwidth consumption and storage occupation. (3) Any one can serve as an auditor to validate the integrity of the outsourced files for unbounded times by only interacting with the cloud storage server. The integrity can be successfully audited with overwhelming probability without retrieving the outsourced file or inspecting all the file blocks. (4) The file owner is anonymous to the cloud storage server. The server can only know the file is uploaded by a legal member of some group, but cannot trace the identity of the member. In the meanwhile, the group manager can reveal the identity of the file owner for liability investigation.

We provide a secure generic construction of GPoS schemes. We first formalize the security model for GPoS schemes with public auditability. A GPoS scheme should be secure against conspiracy attacks from group members for forging secret keys and meta-data. It should be also able to prevent the cloud storage server involved in auditing the outsourced files from forging an integrity proof and from extracting the file owner's identity from maintained files. The meta-data must be (1) generated with a secret key issued by the group manager, (2) aggregatable, (3) publicly auditable, and (4) ownership privacy-preserving. These requirements raise a great challenge in GPoS construction. We manage to address this challenge by identifying some useful properties, that is, homomorphic composability and homomorphic verifiability (defined in Section 3.1) in some signature schemes. By exploiting these properties, we propose a generic GPoS construction proven secure if the underlying signature is existentially unforgeable and the standard Computational Diffie–Hellman (CDH) assumption holds.

We implement efficient concrete GPoS schemes. Observe that the well-known Boneh–Boyen short signature scheme [4] satisfies the required properties. By following the generic construction, we instantiate a practical GPoS scheme under the CDH assumption in the standard model. Particularly, the Boneh–Boyen short signature scheme is employed to issue secret keys for group members. The resulting GPoS scheme takes linearly computation and communication complexities. The instantiation is further optimized in terms of communication overheads and computation costs at the auditor side for auditing the integrity of the outsourced files. This optimization leverages the polynomial commitment technique [12] to commit to the aggregated file block, which means that the created commitment instead of the aggregated block itself will be responded to the auditor. To this end, the security of the optimized instantiation is reduced to the s-SDH assumption. One may observe that GPoS/PoS/PDP/PoR schemes built over symmetric bilinear groups may be optimized in the same way, which indicates that the polynomial commitment based communication optimization approach is general and universally useful.

The performance of our GPoS instantiations is comprehensively analyzed and compared. The analysis shows that our schemes are advantageous over existing PoS/PDP/PoR schemes in the multi-user setting. Our optimized instantiation enjoys an efficient tradeoff for the basic one, that is, although some additional but affordable computation burden is brought to the resource-redundant cloud server, the communication overheads are significantly saved during audition of the outsourced files. Particularly, the transmitted proof comprises of only three elements, independent of the sector number in a block and the block number in a challenge.

1.2 Related Work

Considerable efforts have been made to remotely check the integrity of the outsourced files in a single-user setting. Ateniese *et al.* [1] and Juels and Kaliski [11] independently investigated secure storage in untrusted clouds and introduced the notions of PDP and PoR, respectively. With these techniques, the integrity of the outsourced files can be audited without retrieving them back. Specifically, Ateniese *et al.*'s scheme supports unbounded number of integrity auditing. Shacham and Waters [13] proposed privately and publicly auditable PoR schemes with strong security proofs. In privately auditable PDP/PoR, the integrity of the outsourced file can only be audited by the one who has the corresponding secret key, that is, the file owner, while anyone can play the role as an auditor in publicly auditable schemes. Ateniese, Kamara, and Katz [3] showed how to convert a homomorphic identification protocol into a public-key homomorphic linear authenticator (HLA) and further constructed PoS from HLA. Dodis, Vadhan, and Wichs [8] improved PoR using the tools based on the coding and complexity theory. Wang *et al.* [20] incorporated a third party auditor (TPA) into the scheme. Such a TPA is able to audit the outsourced files on behalf of owners, but learns nothing about file contents. Zheng and Xu [30] discussed how to avoid maintaining multiple copies of the same file in PoS setting. In [24], Xu and Chang presented a privately auditable PoR using polynomial commitment technique [12], which greatly saves communication overheads compared with the scheme in [13]. Yuan and Yu [27] showed a publicly auditable PoR with the same polynomial commitment technique. Noticed too many heavy computations should be taken at the client side, Wang *et al.* [23] investigated how to offload PDP schemes by securely outsourcing them to a computation server.

In practice, there is a need of dynamic PDP/PoR/PoS schemes in which the outsourced files can be further updated at a block level with insertion, deletion and modification operations. Some early works [1, 2] can partially support those operations. The recent works [22, 9, 29] are equipped with all those functionalities. Among them, Wang *et al.*' scheme [22] is constructed using Merkle Hash Tree, while Erway *et al.*'s [9] schemes are based on the authenticated skip lists and RSA trees. Zhang and Blanton [29] presented a dynamic PDP scheme by employing balanced update trees. Built on oblivious RAM, Cash, Küpçü, and Wichs [5] investigated how to ensure the latest version of the outsourced file maintained by the storage server. Shi, Stefanov, and Papamanthou [14] provided a more efficient dynamic PoR based on special authenticated structures.

Some proposals have been proposed to address the integrity concern of the remote files in a multi-user setting. Wang, Li, and Li [16] investigated how to share data by a group of members through clouds. Although their scheme achieves privacy of the group member's identity in auditing the integrity, it cannot support public auditability or

Table 1: Comparison with Related Works in Multi-user Setting

Scheme	Issuing member's key	Shared secret para.	ID-based	Auditability	Ownership privacy
Wang, Li, and Li [16]	Group manager	Yes	×	TPA (Private)	✓
Wang, Li, and Li [17]	Local	No	×	TPA	✓
Wang *et al.* [15]	–	No	×	Public	✓
Wang, Li, and Li [19]	Group manager	Yes	×	TPA	✓
Wang *et al.* [18]	Local	No	×	TPA	×
Yuan and Yu [28]	Local	No	×	Public	×
Wang *et al.* [21]	Group manager	No	✓	Public*	✓
Yu *et al.* [26]	Local	Yes	×	TPA	×
Our instantiations	Group manager	No	✓	Public	✓

*The public auditor should hold a secret parameter of the file owner when auditing outsourced files.

identity-based deployment. In [17], they revisited the same problem using ring signatures. Each group member should locally prepare his/her secret key, instead of issuing by a group manager. Wang *et al.* [15] considered a scenario such that the group members do not hold secret keys. For any file to be outsourced, the file owner should process it by interacting with a security-mediator in a blind manner, which ensures that the mediator cannot know the file content. Wang, Li, and Li [19] proposed a secure cloud storage scheme that supports dynamic group member changes (for example, join and revocation). However, the group secret key must be delivered to all group members, which is not desirable in practice. Wang *et al.* [18] presented a scheme that enables user revocation without requiring any secret information to be shared among group members. Yuan and Yu [28] presented a scheme that supports multiple users in a group to modify the outsourced file. To this end, the involved members should locally generate secret keys and cooperate in producing public parameters in key generating algorithm. The scheme proposed by Wang *et al.* [21] requires the auditor to hold a secret parameter of file owner when verifying the integrity of the outsourced files. Also, in their scheme [21], the secret key of each group member consists of two elements in $\mathbb{G}$ and $\mathbb{Z}_p$. Yu *et al.*'s proposal [26] achieves ownership privacy-preserving against TPA when auditing the outsourced files, although the group members should locally generate their keys and interact to negotiate a pair of group public/secret keys.

Table 1 summarizes the comparison among the above mentioned PoS-related schemes in multi-user setting. They are compared in terms of comparable properties, that is, the manner of generating secret keys for group members, whether some secret parameters should be shared among group members, whether the scheme is proposed in identity-based setting, auditability of the outsourced files, and whether the identity of file owner is privacy-preserving against the cloud storage server. From the table, it can be seen that our schemes are advantageous over related schemes in ownership privacy, public audibility, freeness of sharing secret parameters and/or freeness of certificates (due to our identity-based design).

2. MODELLING GPOS

In this section, we define the architecture of GPoS schemes and formalize the corresponding security model as well.

2.1 System Architecture

A GPoS system involves four types of entities (as illustrated in Figure 1). The cloud storage server which is a semi-trusted party and maintained by some cloud storage provider, offers the remote storage services to cloud users, for example, file owners. This server also has powerful computation capability to respond the clients' requests. The group manager trusted by its members initiates the system and issues the secret keys for all the members. A file owner, that is, a group member, holds files and would like to outsource them to the cloud storage server. The auditor is also a cloud user, but may be not a group member. The group public information as well as the file tags of the outsourced files are accessible to the auditor. In this way, the auditor can audit the integrity of these files on behalf of group manager and the file owners.

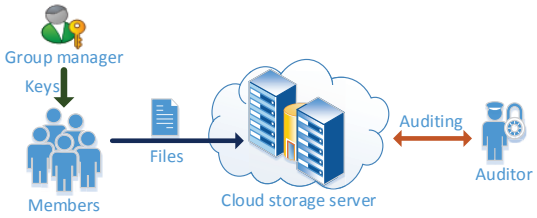

Figure 1: GPoS system model

2.2 Formal System Definition

A GPoS scheme with public auditability comprises six polynomial-time computable algorithms, that is, Setup, KeyExt, PrFile, Chall, PrfGen and Verify.

- $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda)$: On input a security parameter λ, the system setup algorithm, run by the group manager, outputs a pair of group public key and group secret key $(\mathsf{gpk}, \mathsf{gsk})$.

- $\mathsf{sk}_\ell \leftarrow \mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$: On input a group public key gpk, a group secret key gsk and a member identity id_ℓ, the key extraction algorithm, run by the group manager, outputs a secret key sk_ℓ. This secret key is verifiable by its holder, that is, member id_ℓ, under group public key gpk.

- $(\tau, F^*) \leftarrow \mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$: On input a group public key gpk, a member's secret key sk_ℓ and a file $F \in \{0,1\}^*$, the processing file algorithm, run by the file

owner id_ℓ (a group member), outputs a file tag τ and a processed file F^* that comprises of F and a number of meta-data $\vec{\sigma}$.

- $C \leftarrow \mathsf{Chall}(\mathsf{gpk}, \tau)$: On input a group public key gpk and a file tag τ, the challenge generation algorithm, run by the auditor, outputs a challenge C.
- $R \leftarrow \mathsf{PrfGen}(\mathsf{gpk}, F^*, C)$: On input a group public key gpk, a processed file F^* and a challenge C, the proof generation algorithm, run by the cloud storage server, outputs a proof R.
- $0/1 \leftarrow \mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$: On input a group public key gpk, a file tag τ and a challenge/proof pair (C, R), the verification algorithm, run by the auditor, outputs "1" if R is a valid proof for C, or "0" otherwise.

Informally, the correctness property of a GPoS scheme requires that for any file that processed by any member in the group, if the cloud storage server performs honestly, then the proof R is valid for the challenge C in any round of integrity auditing protocol, that is, Chall-PrfGen-Verify. Formally, we have

Definition 1 (Correctness). *A GPoS scheme* (Setup, KeyExt, PrFile, Chall, PrfGen, Verify) *is* correct *if for any group key pair* $(\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda)$*, any member* id_ℓ *with secret key* $\mathsf{sk}_\ell \leftarrow \mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$*, and any file* $F \in \{0,1\}^*$*, let* $(\tau, F^*) \leftarrow \mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$*, the verification equation* $\mathsf{Verify}(\mathsf{gpk}, \tau, C, \mathsf{PrfGen}(\mathsf{gpk}, F^*, C)) = 1$ *holds for any challenge* $C \leftarrow \mathsf{Chall}(\mathsf{gpk}, \tau)$.

2.3 Adversary Model and Security Definitions

Note that we do not need to specially consider outsider attackers as they cannot be more powerful than insider attackers, that is, the colluding members or the server. Intuitively, a GPoS scheme may encounter the following conspiracy attacks from the group members and the cloud storage server.

- Secret key forgery. Several group members may collude to forge a secret key with respect to another group member.
- Meta-data forgery. The group members or the cloud storage server may forge meta-data for some file and group member.
- Proof forgery. The cloud storage server and group member may collude to forge a proof when auditing the integrity of some outsourced file with respect to some group member.

An insightful observation indicates the second type of attacks about meta-data forgery can be captured by the third case. Hence, in the following discussion, we only need to consider the secret key forgery and proof forgery. To capture them, we define a formal security model with the following game where a probabilistic polynomial-time (PPT) adversary $\mathcal{A}$ interacts with the challenger $\mathcal{C}$.

Setup: Suppose that the adversary controls a set S_c of corrupted members in the group and sends this set to $\mathcal{C}$. The challenger runs $\mathsf{Setup}(1^\lambda)$ to generate a pair of group public/secret keys $(\mathsf{gpk}, \mathsf{gsk})$ and gives gpk to $\mathcal{A}$.

Queries: The adversary adaptively queries the challenger. For each query, the challenger records the queried information as well as response in lists that are initiated as empty.

- *Key extraction:* The adversary adaptively queries the challenger to obtain the secret key of a member in the corrupted set S_c. The challenger returns the result of KeyExt for each queried member identity.
- *Processing file:* The adversary sends a file F and a member identity id_ℓ to the challenger. If the secret key of member id_ℓ has not been queried before, then the challenger should first run $\mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$. The challenger computes $(\tau, F^*) \leftarrow \mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$ and sends (τ, F^*) to $\mathcal{A}$. Note that for each file, there is a unique file identifer in the file tag τ. For guaranteeing its uniqueness, we let it be randomly chosen by the challenger during the query.
- *Integrity auditing:* For any processed file in above processing file queries, the challenger (acting as the auditor) can audit its integrity by challenging the adversary (acting as the prover). That is, they jointly carry out the integrity auditing protocol. In detail, for any file F in the query list, the challenger can challenge $\mathcal{A}$ with $C \leftarrow \mathsf{Chall}(\mathsf{gpk}, \tau)$, and $\mathcal{A}$ responds with a proof R. Then, the challenger verifies R by invoking $\mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$ and gives the results to $\mathcal{A}$.

End-Game: Finally, the adversary outputs a secret key sk'_ℓ for some member id'_ℓ, or a pair of challenge/proof (C', R') with regard to some file F' identified by file tag τ'.

Definition 2 (Soundness). *A GPoS scheme is* sound *if for any PPT adversary* $\mathcal{A}$ *playing the above mentioned security game by interacting with the challenger, the outputs are neither of the following cases:*

- ***Case 1.*** *The secret key* sk'_ℓ *is valid under the group public key* gpk *for a member* id'_ℓ *but* $\mathsf{id}'_\ell \notin S_c$.
- ***Case 2.*** *The pair of challenge/proof* (C', R') *is valid but* R' *does not equal to that generated by the challenger from locally maintained information.*

A secure GPoS scheme also requires that, in the entire life span of an outsourced file, its owner identity should be hidden from the cloud storage server. Essentially, this *ownership privacy-preserving* property requires the files should be uploaded in the name of the group. As in the real application scenario, the files are uploaded by the employee under the company's account. Also, in any round of integrity auditing protocol with regard to any outsourced file, the cloud storage server should be able to respond with a valid integrity proof R without using the file owner's identity. More technically,

Definition 3 (Ownership privacy). *A GPoS scheme is* ownership privacy-preserving *against the cloud storage server if for any file* $F \in \{0,1\}^*$ *and any two distinct members* $\mathsf{id}_{\ell,1}$ *and* $\mathsf{id}_{\ell,2}$ *in the same group, the following two distributions are identical from the view of the cloud storage server:*

$$\left\{ \vec{\sigma}_1 : \begin{array}{r} (\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_{\ell,1} \leftarrow \mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_{\ell,1}), \\ (\tau_1, F_1^*) \leftarrow \mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_{\ell,1}, F) \end{array} \right\},$$

and

$$\left\{ \vec{\sigma}_2 : \begin{array}{r} (\mathsf{gpk}, \mathsf{gsk}) \leftarrow \mathsf{Setup}(1^\lambda), \\ \mathsf{sk}_{\ell,2} \leftarrow \mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_{\ell,2}), \\ (\tau_2, F_2^*) \leftarrow \mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_{\ell,2}, F) \end{array} \right\}.$$

3. GENERIC GPOS CONSTRUCTION

Suppose $\mathbb{G}_1 = \langle g_1 \rangle$ and $\mathbb{G}_2 = \langle g_2 \rangle$ be (multiplicative) cyclic groups of prime order p with efficient group actions. The groups $(\mathbb{G}_1, \mathbb{G}_2)$ are bilinear if there exists a (multiplicative) cyclic group $\mathbb{G}_T$ of the same order and an efficient bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ such that:

(1) Bilinearity: $\forall h \in \mathbb{G}_1, \hbar \in \mathbb{G}_2$, and $\forall \alpha, \beta \in \mathbb{Z}_p^*$, $\hat{e}(h^\alpha, \hbar^\beta) = \hat{e}(h, \hbar)^{\alpha\beta}$;

(2) Non-degeneracy: $\hat{e}(g_1, g_2) \neq 1$.

Our GPoS scheme and instantiations rely on the following well-known computational assumptions.

Computational Diffie–Hellman Assumption: Given a triple (g, g^α, g^β) for $\alpha, \beta \in_R \mathbb{Z}_p^*$, any PPT algorithm $\mathcal{A}$ has negligible probability, that is,

$$\Pr[\mathcal{A}(g, g^\alpha, g^\beta) = g^{\alpha\beta} : \alpha, \beta \in_R \mathbb{Z}_p^*]$$

to compute $g^{\alpha\beta}$.

s-Strong Diffie–Hellman (s-SDH) Assumption [4]: Given a $(s+1)$-tuple

$$(g, g^\alpha, \cdots, g^{\alpha^s})$$

for randomly chosen $\alpha \in_R \mathbb{Z}_p^*$, any PPT algorithm $\mathcal{A}$ has negligible probability

$$\Pr\left[\mathcal{A}(g, g^\alpha, \cdots, g^{\alpha^s}) = (z, g^{\frac{1}{\alpha+z}}) : \alpha \in_R \mathbb{Z}_p^*, z \in \mathbb{Z}_p^* \setminus \{-\alpha\}\right]$$

to compute a pair $(z, g^{\frac{1}{\alpha+z}})$.

3.1 Useful Properties in Signatures

To construct GPoS, we first need a secure way to generate secret keys for the group members. It is well-known that in group-oriented cryptographic primitives, the group members should not be allowed to (freely) generate secret keys by themselves. Instead, these keys are usually generated and distributed for all members by a (trusted) group manager. The secret key of a group member should be associated with his/her identity. To this end, the group manager usually signs the identity of the member and outputs the signature as the secret key of the member. An advantage of this approach is that it is naturally collusion-resistant in the sense that even colluding users cannot forge a valid secret key if the underlying signature is existentially unforgeable. However, this approach also brings a challenge in constructing group-oriented PoS schemes, not only because the files are processed by group members with their secret keys and each meta-data should incorporate at least a file identifier and a file block with many sectors, but also the produced meta-data should be aggregatable and publicly auditable. In fact, the situation will be even worse in constructing generic GPoS schemes equipped with user privacy.

We observe that some signature schemes enjoy several interesting properties useful to address the above challenge. We refer to these properties as homomorphic composability and homomorphic verifiability defined as follows. Let $\mathcal{S} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme defined over some cyclic group $\mathbb{G} = \langle g \rangle$, and $\mathcal{M}$, $\mathcal{K}$ and Σ be the message space, secret key space and signature space of $\mathcal{S}$, respectively, where the signature space is assumed to be a finite multiplicative cyclic group.

Definition 4 (Homomorphic Composability). *A signature scheme $\mathcal{S}$ is (φ, F)-homomorphic composable if*

(1) there exist two efficiently computable functions such as $\varphi : \mathcal{K} \times \mathcal{M} \to \mathcal{R}$ where $\mathcal{R}$ denotes some ring, and $F : \mathbb{G} \times \mathcal{R} \to \Sigma$; and

(2) for each message $m \in_R \mathcal{M}$ and every key pair $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathcal{S}.\mathsf{KGen}(1^\lambda)$, the corresponding signature has the form $\sigma = F(g, \varphi(\mathsf{sk}, m)) \leftarrow \mathcal{S}.\mathsf{Sign}(\mathsf{sk}, m)$. It holds that

$$F(g, \varphi(\mathsf{sk}, m))^x = F(g^x, \varphi(\mathsf{sk}, m))$$

for any value $x \in_R \mathcal{R}$, where $F()^x$ is the exponentiation operation over Σ.

Definition 5 (Homomorphic Verifiability). *A signature scheme $\mathcal{S}$ is (φ, F)-homomorphic verifiable if*

(1) there is an efficient test algorithm $\Xi(\mathsf{pk}; m, \sigma; \ddot{x}, \ddot{y})$ which takes a public key pk, a message/signature pair (m, σ), and a pair of elements $\ddot{x} \in \mathbb{G}$ and $\ddot{y} \in \Sigma$; and

(2) the algorithm Ξ outputs "1" if the given pair of message/signature is valid under pk, that is, $\mathcal{S}.\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1$, and $\varphi_1 = \varphi_2$ in $\mathcal{R}$, where $\varphi_1 = F^{-1}(g, \sigma)$ and $\varphi_2 = F^{-1}(\ddot{x}, \ddot{y})$. Otherwise, outputs "0".

We provide two exemplary signature schemes that satisfy these properties.

Boneh–Boyen scheme [4]. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be an asymmetric bilinear map, where $\mathbb{G}_1 = \langle g_1 \rangle$, $\mathbb{G}_2 = \langle g_2 \rangle$ and $\mathbb{G}_T$ are (multiplicative) cyclic groups of prime order p.

- $\mathsf{KGen}(1^\lambda)$: Pick a random value $\gamma \in_R \mathbb{Z}_p^*$ to be the secret key sk, and compute $\varpi = g_2^\gamma$. The public key is
$$\mathsf{pk} = (\hat{e}, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, p, \varpi)$$
- $\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, m)$: For any given message $m \in_R \mathbb{Z}_p$, compute the signature
$$\sigma = g_1^{\frac{1}{\gamma+m}}$$
If $m + \gamma = 0$, then set the signature as the identity element in $\mathbb{G}_1$.
- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$: If the equality
$$\hat{e}(\sigma, \varpi \cdot g_2^m) \stackrel{?}{=} \hat{e}(g_1, g_2)$$
holds, then the signature σ is valid for m and thus output "1"; otherwise, output "0".

We define

$$\varphi(\mathsf{sk}, m) = \frac{1}{\gamma + m} \mod p$$

and

$$F(g_1, \varphi(\mathsf{sk}, m)) = g_1^{\varphi(\mathsf{sk}, m)}$$

For any $x \in_R \mathbb{Z}_p^*$, the function F satisfies homomorphic composability property, that is,

$$F(g_1, \varphi(\mathsf{sk}, m))^x = g_1^{x \cdot \frac{1}{\gamma+m}} = F(g_1^x, \varphi(\mathsf{sk}, m)).$$

Define the testing algorithm Ξ to output "1" if and only if both

$$\hat{e}(\sigma, \varpi \cdot g_2^m) = \hat{e}(g_1, g_2)$$

and

$$\hat{e}(\ddot{y}, \varpi \cdot g_2^m) = \hat{e}(\ddot{x}, g_2)$$

hold. Hence, it also satisfies homomorphic verifiability.

Gennaro–Halevi–Rabin scheme [10]. Let $H : \{0,1\}^* \to \{2n+1 : n \in \mathbb{Z}\}$ be a collision-resistant hash function.

- $\mathsf{KGen}(1^\lambda)$: Pick two random primes $p, q \in_R [2^\lambda, 2^{\lambda+1}]$ such that both $\frac{p-1}{2}$ and $\frac{q-1}{2}$ are also primes. Compute $N = pq$. Let $\mathbb{G}$ be a subgroup of squares in $\mathbb{Z}_N^*$. Pick a random element $g \in_R \mathbb{G}$. The secret key is $\mathsf{sk} = (p, q)$, and the public key is $\mathsf{pk} = (N, g)$.

- $\mathsf{Sign}(\mathsf{pk}, \mathsf{sk}, m)$: For any given message $m \in \{0,1\}^*$, compute the signature

$$\sigma = g^{\frac{1}{H(m)}}$$

- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$: If the equality

$$\sigma^{H(m)} \stackrel{?}{=} g$$

holds, then the signature σ is valid for m and thus output "1"; otherwise, output "0".

For GHR signature scheme, we have

$$\varphi(\mathsf{sk}, m) = \frac{1}{H(m)} \bmod \xi(N)$$

and

$$\digamma(g, \varphi(\mathsf{sk}, m)) = g^{\varphi(\mathsf{sk}, m)} \bmod N$$

where $\xi(N)$ is the Euler totient function of N. For any $x \in_R \mathbb{Z}^*$, the function $\digamma$ satisfies homomorphic composability property, that is,

$$\digamma(g, \varphi(\mathsf{sk}, m))^x = g^{x \cdot \frac{1}{H(m)}} = \digamma(g^x, \varphi(\mathsf{sk}, m)).$$

Define the testing algorithm Ξ to output "1" if and only if both

$$\sigma^{H(m)} = g$$

and

$$\ddot{y}^{H(m)} = \ddot{x}$$

hold. Hence, it also satisfies homomorphic verifiability.

3.2 Our GPoS Construction

Let $\mathcal{S}_k = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a secure signature scheme which has the properties described in Section 3.1. Also let $\mathcal{S}_t = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a secure standard signature scheme. Suppose the group consists of n members and $H_0 : \{0,1\}^* \to \mathcal{R}$ be a collision-resistant hash function.

For GPoS construction with the above defined properties, a group member id_ℓ only holds his/her secret key in the form of $\digamma(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell))$, where gsk denotes a group secret key. Thus, he/she cannot directly use gsk or $\varphi(\mathsf{gsk}, \mathsf{id}_\ell)$ for producing meta-data (signature for a file block). We explain how to employ trapdoor technique to circumvent this issue. In detail, the group member id_ℓ can randomly pick (secret) values $\alpha_i \in_R \mathcal{R}$ and compute the corresponding powers $u_i = g^{\alpha_i}$ as public parameters. When signing on a list of messages $\{m_i\}$, the member id_ℓ first evaluates a linear function

$$f_e = f(\{m_i, \alpha_i\}) = \sum \alpha_i m_i$$

over ring $\mathcal{R}$, and then computes

$$\sigma' = (\digamma(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)))^{f_e} = \digamma(g^{f_e}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)).$$

In this way, it is easy to see that the component g^{f_e} can be recovered using public parameters $\{u_i\}$ and messages $\{m_i\}$. Hence, under group public key gpk, the test algorithm Ξ can go through as follows

$$\Xi\left(\mathsf{gpk}; \mathsf{id}_\ell, \digamma(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)); g^{f_e}, \sigma'\right) = 1.$$

Now we are ready to describe our generic construction.

$\mathsf{Setup}(1^\lambda)$: The group manager invokes $\mathcal{S}_k.\mathsf{KGen}(1^\lambda)$ to obtain a pair of public/secret keys, and sets them as the group public key and group secret key $(\mathsf{gpk}, \mathsf{gsk})$.

$\mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$: For each member id_ℓ $(1 \le \ell \le n)$ in the group, the group manager computes a signature of his/her identity as follows

$$\mathsf{sk}_\ell \leftarrow \mathcal{S}_k.\mathsf{Sign}(\mathsf{gsk}, \mathsf{id}_\ell) \in \Sigma.$$

Such a signature serves as his/her secret key. When receiving sk_ℓ, the member id_ℓ is able to validate it by invoking $\mathcal{S}_k.\mathsf{Vrfy}$ with group public key gpk.

$\mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$: Given a file F, the member id_ℓ splits it into blocks such that each block has s sectors (as elements in $\mathcal{R}$) as follows

$$F = \{F_i = (f_{i,1}, \cdots, f_{i,s}) : 1 \le i \le r\}. \quad (1)$$

Then choose a random file identifier $fid \in_R \mathcal{R}$ and $(s+1)$ random values $\alpha_0, \alpha_1, \cdots, \alpha_s \in_R \mathcal{R}$. Compute

$$u_j = g^{\alpha_j} \in \mathbb{G}$$

for each $0 \le j \le s$. Let τ_0 be the concatenation string of

$$(\mathsf{gpk}, \mathsf{id}_\ell, u_0, u_1, \cdots, u_s, fid, r)$$

Generate the file tag by the following steps:

- Compute $(\mathsf{tpk}, \mathsf{tsk}) \leftarrow \mathcal{S}_t.\mathsf{KGen}(1^\lambda)$ to obtain a pair of public/secret keys;
- Compute $\vartheta \leftarrow \mathcal{S}_t.\mathsf{Sign}(\mathsf{tsk}, \tau_0)$ to obtain a signature of string τ_0.
- Send the file tag $\tau = (\tau_0, \mathsf{tpk}, \vartheta)$ to the group manager.

For each file block F_i $(1 \le i \le r)$, the member id_ℓ computes

$$\theta_i = \alpha_0 H_0(fid \parallel i) + \sum_{j=1}^{s} \alpha_j f_{i,j} \in \mathcal{R}$$

and generates meta-data as

$$\sigma_i \leftarrow \mathsf{sk}_\ell^{\theta_i} \in \Sigma$$

Send the processed file

$$F^* = \{(F_i, \sigma_i) : 1 \le i \le r\}$$

to the cloud storage server. Deletes the random values $\alpha_0, \alpha_1, \cdots, \alpha_s$, secret key tsk and the file locally.

$\mathsf{Chall}(\mathsf{gpk}, \tau)$: The auditor runs the algorithm as follows.

1. Invoke $\mathcal{S}_t.\mathsf{Vrfy}(\mathsf{tpk}, \tau_0, \vartheta)$ to validate the file tag τ. If it is invalid, outputs "0" and terminates.

2. Pick a random subset $Q \subseteq [1, r]$ and choose a random value $\beta_i \in_R \mathcal{R}$ for each $i \in Q$. Send the challenge

$$C = (fid, Q, \{\beta_i : i \in Q\})$$

to the cloud storage server.

$\mathsf{PrfGen}(\mathsf{gpk}, F^*, C)$**:** The cloud storage server computes the aggregated file block $\vec{\mu} = (\mu_1, \cdots, \mu_s)$ and meta-data σ as follows

$$\mu_j = \sum_{i \in Q} \beta_i f_{i,j} \in \mathcal{R} \text{ for each } j \in [1, s],$$

and

$$\sigma = \prod_{i \in Q} \sigma_i^{\beta_i} \in \Sigma. \tag{2}$$

Return the proof $R = (\vec{\mu}, \sigma)$ to the auditor.

$\mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$**:** If R cannot be parsed, output "0" and terminates. Otherwise, check

$$\Xi\left(\mathsf{gpk}; \mathsf{id}_\ell, \mathsf{sk}_\ell; u_0^{\sum_{i \in Q} \beta_i H_0(fid \| i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, \sigma\right) \stackrel{?}{=} 1. \tag{3}$$

If so, output "1"; otherwise, output "0".

3.3 The Security

THEOREM 1. *The proposed GPoS scheme is correct.*

PROOF. We only show Equation (3) holds as the other parts are straightforward. Observing that

$$\mathsf{sk}_\ell = \digamma(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell))$$

we have

$$\begin{aligned}
\sigma &= \prod_{i \in Q} \sigma_i^{\beta_i} = \prod_{i \in Q} \digamma\Big(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big)^{\theta_i \beta_i} \\
&= \digamma\Big(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big)^{\sum_{i \in Q} \theta_i \beta_i} \\
&= \digamma\Big(g^{\sum_{i \in Q} \beta_i (\alpha_0 H_0(fid \| i) + \sum_{j=1}^{s} \alpha_j f_{i,j})}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big) \\
&= \digamma\Big(\prod_{i \in Q} g^{\beta_i \alpha_0 H_0(fid \| i)} \cdot \prod_{i \in Q} \prod_{j=1}^{s} g^{\beta_i \alpha_j f_{i,j}}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big) \\
&= \digamma\Big(\prod_{i \in Q} g^{\beta_i \alpha_0 H_0(fid \| i)} \cdot \prod_{j=1}^{s} u_j^{\sum_{i \in Q} \beta_i f_{i,j}}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big) \\
&= \digamma\Big(u_0^{\sum_{i \in Q} \beta_i H_0(fid \| i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big).
\end{aligned}$$

It can be seen that σ has the same component $\varphi(\mathsf{gsk}, \mathsf{id}_\ell)$ as the secret key sk_ℓ. According to the definition of algorithm Ξ, the correctness follows. □

THEOREM 2. *Suppose that the signature scheme $\mathcal{S}_t$ for file tags is existentially unforgeable. The proposed GPoS scheme is sound for any PPT adversary $\mathcal{A}$ if the CDH assumption holds.*

PROOF. Since the unforgeability of group members' secret keys is guaranteed by a secure signature scheme $\mathcal{S}_k$, it is omitted in the following discussion while only the soundness with regard to the integrity auditing of outsourced files is considered.

Suppose a PPT adversary $\mathcal{A}$ who controls a set S_c of group members and breaks the soundness of the proposed generic GPoS scheme. Let the adversary play the security game described in Definition 2 by interacting with the challenger $\mathcal{C}$. We show that if the adversary can output a (forged) valid pair of challenge/proof (C', R') with regard to a file F' and a tag τ', then we can construct an algorithm $\mathcal{B}$ by interacting with $\mathcal{A}$ to break the CDH assumption.

Setup: The adversary $\mathcal{A}$ sends the corrupted set S_c to $\mathcal{B}$. On input a security parameter λ, the challenger generates a pair of group public key and secret key $(\mathsf{gpk}, \mathsf{gsk})$. Send gpk to $\mathcal{B}$, who forwards it to the adversary $\mathcal{A}$.

Queries: The adversary can adaptively interact with the simulator $\mathcal{B}$ who maintains all the intermediate information, that is, all the queries and responses.

- *Processing file:* For each query from $\mathcal{A}$ with a file F and a member identity id_ℓ, the simulator $\mathcal{B}$ passes them to $\mathcal{C}$. If the identity has not been queried before, then the challenger first invokes KeyExt to extract a private key sk_ℓ for it. After running the algorithm PrFile, the challenger gives the private key sk_ℓ, a file tag τ and a list of meta-data $\{\sigma_i\}_{1 \le i \le r}$ to $\mathcal{B}$, where the file identifier fid (an element in τ) is randomly chosen by $\mathcal{C}$. The simulator $\mathcal{B}$ forwards the received information to $\mathcal{A}$. If $\mathsf{id}_\ell \notin S_c$, then sk_ℓ cannot be given to $\mathcal{A}$.

- *Integrity auditing:* For this type of queries, the simulator and the adversary play the roles as a verifier and a prover, respectively. For any file F that has been queried for processing, the simulator $\mathcal{B}$ runs $\mathsf{Chall}(\mathsf{gpk}, \tau)$ to generate a challenge

  $$C = (fid, Q, \{\beta_i : i \in Q\})$$

  and sends it to $\mathcal{A}$. The adversary responds with a proof $R = (\vec{\mu}, \sigma)$. The simulator verifies the proof P by running $\mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$ and gives the verification results to $\mathcal{A}$.

End-Game: Finally, the adversary outputs a (forged) valid pair of challenge/proof (C', P') with regard to a file F' and a tag

$$\tau' = (\tau_0', \mathsf{tpk}, \vartheta')$$

that is, the pair of

$$C' = (fid, Q, \{\beta_i : i \in Q\})$$

and

$$R' = (\vec{\mu}', \sigma')$$

satisfy the testing algorithm Ξ. Note that file F' has been queried for processing. Assume it belongs to some member $\mathsf{id}_\ell \notin S_c$ and has identifier fid, both of which are specified in τ'. As we discussed, the forged and real proofs have the following representations, that is,

$$\begin{aligned}
\sigma' &= \digamma\Big(u_0^{\sum_{i \in Q} \beta_i H_0(fid \| i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j'}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big) \\
&= \digamma\Big(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\Big)^{\alpha_0 \sum_{i \in Q} \beta_i H_0(fid \| i) + \sum_{j=1}^{s} \alpha_j \mu_j'}
\end{aligned}$$

and

$$\sigma = \mathcal{F}\left(u_0^{\sum_{i\in Q}\beta_i H_0(fid\|i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\right)$$
$$= \mathcal{F}\left(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)\right)^{\alpha_0 \sum_{i\in Q}\beta_i H_0(fid\|i) + \sum_{j=1}^{s}\alpha_j\mu_j}.$$

Notice that, at least one pair of $\{(\mu'_j, \mu_j)\}_{1\le j\le s}$ should be different, since otherwise, $\sigma' = \sigma$ would also hold. Without loss of generality, assume $\mu'_1 \neq \mu_1$, while $\mu'_j = \mu_j$ for $2 \le j \le s$. In this case, the simulator, having the private key

$$\mathsf{sk}_\ell = \mathcal{F}(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell)) \in \mathbb{G}$$

and the public parameter $u_1 = g^{\alpha_1} \in \mathbb{G}$ with unknown exponents, can compute the CDH answer $\mathcal{F}(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell))^{\alpha_1}$ as follows

$$\left(\frac{\sigma'}{\sigma}\right)^{\frac{1}{\mu'_1-\mu_1}} = \left(\mathcal{F}(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell))^{\alpha_1(\mu'_1-\mu_1)}\right)^{\frac{1}{\mu'_1-\mu_1}}$$
$$= \mathcal{F}(g, \varphi(\mathsf{gsk}, \mathsf{id}_\ell))^{\alpha_1}.$$

Thus, the simulator breaks the CDH assumption. □

THEOREM 3. *The proposed GPoS scheme is ownership privacy-preserving against the cloud storage server.*

PROOF. On one hand, it is easy to see that, when auditing an outsourced file with respect to a group member id_ℓ, the cloud storage server does not know this specific membership. On the other hand, the produced meta-data in the processed file look random to the cloud server if the elements in both the ring $\mathcal{R}$ and the group $\mathbb{G}$ are uniformly distributed. Since all the values such as α_i-es are randomly chosen in $\mathcal{R}$, the meta-data $\sigma_i \in \mathbb{G}$ are random elements in $\mathbb{G}$ and independent of the file owner's identity in the view of the cloud storage server. □

4. INSTANTIATIONS

In this section, we first present an instantiation based on the Boneh–Boyen short signature [4]. We next show that a special type of GPoS instantiations can be further optimized with a polynomial commitment technique [12]. The signature scheme $\mathcal{S}_t$ is the same as in Section 3.2. Let $H_0 : \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function.

4.1 A CDH-based Instantiation

$\mathsf{Setup}(1^\lambda)$: The group manager picks a bilinear pairing $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$, where $\mathbb{G} = \langle g \rangle$ and $\mathbb{G}_T$ are cyclic groups of prime order p. Choose a random value $\gamma \in_R \mathbb{Z}_p^*$ and computes $\varpi = g^\gamma$. Let $H : \{0,1\}^* \to \mathbb{Z}_p^*$ be a collision-resistant hash function. The group public key is

$$\mathsf{gpk} = (\hat{e}, \mathbb{G}, \mathbb{G}_T, g, \varpi, H)$$

and the group secret key is $\mathsf{gsk} = \gamma$.

$\mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$: For each member id_ℓ $(1 \le \ell \le n)$ in the group, the group manager computes a secret key

$$\mathsf{sk}_\ell = g^{\frac{1}{\gamma + H(\mathsf{id}_\ell)}}.$$

Once receiving sk_ℓ, the member id_ℓ can validate it by checking whether

$$\hat{e}\left(\mathsf{sk}_\ell, \varpi \cdot g^{H(\mathsf{id}_\ell)}\right) \stackrel{?}{=} \hat{e}(g, g)$$

holds.

$\mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$: The member id_ℓ splits the file F into blocks as shown in Equation (1). Choose a random file identifier $fid \in_R \mathbb{Z}_p^*$ and $(s+1)$ random values $\alpha_0, \alpha_1, \cdots, \alpha_s \in_R \mathbb{Z}_p^*$, and compute

$$u_j = g^{\alpha_j} \in \mathbb{G}$$

for each $0 \le j \le s$. Then, generate the file tag τ in the same way as in Section 3.2 and send it to the group manager.

For each file block F_i $(1 \le i \le r)$, do the follows:

- Compute

$$\theta_i = \alpha_0 H_0(fid \parallel i) + \sum_{j=1}^{s} \alpha_j f_{i,j} \in \mathbb{Z}_p$$

- Generate meta-data as

$$\sigma_i \leftarrow \mathsf{sk}_\ell^{\theta_i} \in \mathbb{G}$$

Then, send the processed file

$$F^* = \{(F_i, \sigma_i) : 1 \le i \le r\}$$

to the cloud storage server, and deletes the random values $\alpha_0, \alpha_1, \cdots, \alpha_s$, secret key tsk and the file locally.

$\mathsf{Chall}(\mathsf{gpk}, \tau)$: The auditor runs the algorithm in two steps.

1. The same to Section 3.2 for validating the file tag. If τ is invalid, output "0" and terminate.
2. Pick a random subset $Q \subseteq [1, r]$ and choose a random value $\beta_i \in_R \mathbb{Z}_p^*$ for each $i \in Q$. Send the challenge

$$C = (fid, Q, \{\beta_i : i \in Q\})$$

to the cloud storage server.

$\mathsf{PrfGen}(\mathsf{gpk}, F^*, C)$: Compute the aggregated file block $\vec{\mu} = (\mu_1, \cdots, \mu_s)$ as follows

$$\mu_j = \sum_{i\in Q} \beta_i f_{i,j} \mod p \text{ for each } j \in [1, s],$$

and calculate the aggregated meta-data σ as shown in Equation (2) but over $\mathbb{G}$. Return the proof $R = (\vec{\mu}, \sigma)$ to the auditor.

$\mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$: If R cannot be parsed, output "0" and terminates. Otherwise, check whether the following equality holds

$$\hat{e}\left(\sigma, \varpi \cdot g^{H(\mathsf{id}_\ell)}\right) \stackrel{?}{=} \hat{e}\left(u_0^{\sum_{i\in Q}\beta_i H_0(fid\|i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, g\right).$$

If so, output "1"; otherwise, output "0".

THEOREM 4. *The proposed GPoS instantiation is correct.*

PROOF. Observe the following equalities

$$\begin{aligned}
&\hat{e}\left(\sigma, \varpi \cdot g^{H(\mathsf{id}_\ell)}\right) \\
&= \hat{e}\left(\prod_{i\in Q} \sigma_i^{\beta_i}, g^{\gamma} \cdot g^{H(\mathsf{id}_\ell)}\right) \\
&= \hat{e}\left(\prod_{i\in Q} \left(g^{\frac{1}{\gamma+H(\mathsf{id}_\ell)}}\right)^{\theta_i\beta_i}, g^{\gamma} \cdot g^{H(\mathsf{id}_\ell)}\right) \\
&= \hat{e}\left(\prod_{i\in Q} g^{\beta_i(\alpha_0 H_0(fid\|i)+\sum_{j=1}^{s}\alpha_j f_{i,j})}, g\right) \\
&= \hat{e}\left(\prod_{i\in Q} g^{\beta_i\alpha_0 H_0(fid\|i)} \cdot \prod_{i\in Q}\prod_{j=1}^{s} g^{\beta_i\alpha_j f_{i,j}}, g\right) \\
&= \hat{e}\left(\prod_{i\in Q} g^{\beta_i\alpha_0 H_0(fid\|i)} \cdot \prod_{j=1}^{s} u_j^{\sum_{i\in Q}\beta_i f_{i,j}}, g\right) \\
&= \hat{e}\left(u_0^{\sum_{i\in Q}\beta_i H_0(fid\|i)} \cdot \prod_{j=1}^{s} u_j^{\mu_j}, g\right).
\end{aligned}$$

Hence, the correctness follows. □

According to Theorem 2 and Theorem 3, we have the following corollaries.

COROLLARY 1. *Suppose that the signature scheme $\mathcal{S}_t$ for file tags is existentially unforgeable. The proposed GPoS instantiation is sound for any PPT adversary $\mathcal{A}$ if the CDH assumption holds.*

COROLLARY 2. *The proposed GPoS instantiation is ownership privacy-preserving against the cloud storage server.*

For easiness to compare with the improved instantiation as shown in next section, the above instantiation is presented over symmetric bilinear groups. In fact, it can also be implemented over asymmetric bilinear groups, that is, by setting the group private key sk_ℓ, the public parameters u_i $(0 \le i \le s)$ and the meta-data $\{\sigma_i : 1 \le i \le r\}$ in $\mathbb{G}_1$, while defining ϖ in $\mathbb{G}_2$.

4.2 Optimized Instantiation

We proceed to optimize the above proposed GPoS instantiation by the leveraging polynomial commitment technique [12]. Our optimization approach is universal in the sense that the communication overheads for integrity auditing in a GPoS scheme may be further reduced if the PoS schemes are built over symmetric bilinear groups. At a high level, similarly to [24, 27], the public parameters u_j-es are generated using a single random element α, that is, they are associated with different powers of α. Furthermore, when an outsourced file is audited, a polynomial commitment with respect to the challenged blocks is produced by the cloud storage server and then validated by the auditor.

$\mathsf{Setup}(1^\lambda)$**:** The same to Section 4.1.

$\mathsf{KeyExt}(\mathsf{gpk}, \mathsf{gsk}, \mathsf{id}_\ell)$**:** The same to Section 4.1.

$\mathsf{PrFile}(\mathsf{gpk}, \mathsf{sk}_\ell, F)$**:** The member id_ℓ splits the file F into blocks as shown in Equation (1). Choose a random file identifier $fid \in_R \mathbb{Z}_p^*$ and two random values $\alpha_0, \alpha \in_R \mathbb{Z}_p^*$, and compute

$$\upsilon = g^{\alpha_0} \in \mathbb{G}$$

and

$$u_j = g^{\alpha^j} \in \mathbb{G}$$

for each $0 \le j \le s-1$. Generate the file tag τ and sends it to the group manager in the same way as Section 3.2 while τ_0 denotes a concatenation string of

$$(\mathsf{gpk}, \mathsf{id}_\ell, \upsilon, u_0, u_1, \cdots, u_{s-1}, fid, r)$$

For each file block F_i $(1 \le i \le r)$, do the follows:

- Compute

$$\theta_i = \alpha_0 H_0(fid \parallel i) + \phi_{\vec{\pi}_i}(\alpha) \mod p$$

  where

$$\phi_{\vec{\pi}_i}(\alpha) = \sum_{j=0}^{s-1} f_{i,j}\alpha^j \mod p$$

- Generate the meta-data as $\sigma_i \leftarrow \mathsf{sk}_\ell^{\theta_i} \in \mathbb{G}$.

Then, send the processed file

$$F^* = \{(F_i, \sigma_i) : 1 \le i \le r\}$$

to the cloud storage server and locally discards the random values α_0, α, secret key tsk and the file information.

$\mathsf{Chall}(\mathsf{gpk}, \tau)$**:** The auditor runs the algorithm as follows.

1. The same to Section 3.2 for validating the file tag. If τ is invalid, outputs "0" and terminates.
2. Pick a random subset $Q \subseteq [1, r]$ and chooses two random values $z, \delta \in_R \mathbb{Z}_p^*$. Sends the challenge

$$C = (fid, Q, z, \delta)$$

   to the cloud storage server.

$\mathsf{PrfGen}(\mathsf{gpk}, F^*, C)$**:** For each $i \in Q$, the cloud storage server calculates a value $\beta_i = \delta^i \mod p$. Generate the aggregated file block

$$\vec{\mu} = (\mu_0, \mu_1, \cdots, \mu_{s-1})$$

by computing

$$\mu_j = \sum_{i\in Q} \beta_i f_{i,j} \mod p \text{ for each } j \in [0, s-1],$$

and calculate the aggregated meta-data σ as shown in Equation (2) but over $\mathbb{G}$. Then, define a polynomial

$$\phi_{\vec{\mu}}(x) = \sum_{j=0}^{s-1} \mu_j x^j \mod p$$

and calculates $\kappa = \phi_{\vec{\mu}}(z) \mod p$. Compute the following polynomial

$$\psi_{\vec{\omega}}(x) = \frac{\phi_{\vec{\mu}}(x) - \phi_{\vec{\mu}}(z)}{x - z}$$

using polynomial long division. Let

$$\vec{\omega} = (\omega_0, \omega_1, \cdots, \omega_{s-2})$$

be the coefficient vector of $\psi_{\vec{\omega}}(x)$. Compute

$$\zeta = g^{\psi_{\vec{\omega}}(\alpha)} = \prod_{j=0}^{s-2} \left(g^{\alpha^j}\right)^{\omega_j}.$$

Return the proof $R = (\zeta, \kappa, \sigma)$ to the auditor.

$\mathsf{Verify}(\mathsf{gpk}, \tau, C, R)$: If R cannot be parsed, output "0" and terminate. Otherwise, check the following equality

$$\hat{e}\left(\sigma, \varpi \cdot g^{H(\mathsf{id}_\ell)}\right) \stackrel{?}{=} \hat{e}\left(\upsilon^{\sum_{i\in Q} \delta^i H_0(fid\|i)} \cdot g^{\kappa} \cdot \zeta^{-z}, g\right) \cdot \hat{e}(\zeta, u_1). \quad (4)$$

If so, output "1"; otherwise, output "0".

THEOREM 5. *The optimized GPoS instantiation is correct.*

PROOF. Since

$$\begin{aligned}
&\hat{e}\Big(\sigma, \varpi \cdot g^{H(\mathsf{id}_\ell)}\Big) \\
&= \hat{e}\left(\prod_{i\in Q} \sigma_i^{\beta_i}, g^{\gamma} \cdot g^{H(\mathsf{id}_\ell)}\right) \\
&= \hat{e}\left(\prod_{i\in Q} \left(g^{\frac{1}{\gamma + H(\mathsf{id}_\ell)}}\right)^{\theta_i \beta_i}, g^{\gamma} \cdot g^{H(\mathsf{id}_\ell)}\right) \\
&= \hat{e}\left(\prod_{i\in Q} g^{\beta_i(\alpha_0 H_0(fid\|i) + \phi_{\vec{\pi}_i}(\alpha))}, g\right) \\
&= \hat{e}\left(\prod_{i\in Q} \upsilon^{\beta_i H_0(fid\|i)} \cdot \prod_{i\in Q} g^{\beta_i \phi_{\vec{\pi}_i}(\alpha)}, g\right) \\
&= \hat{e}\left(\upsilon^{\sum_{i\in Q} \beta_i H_0(fid\|i)}, g\right) \hat{e}\Big(g^{\sum_{i\in Q} \beta_i \phi_{\vec{\pi}_i}(\alpha)}, g\Big) \\
&= \hat{e}\left(\upsilon^{\sum_{i\in Q} \delta^i H_0(fid\|i)}, g\right) \hat{e}\Big(g^{\phi_{\vec{\mu}}(\alpha)}, g\Big)
\end{aligned}$$

and

$$\begin{aligned}
&\hat{e}\left(g^{\kappa} \cdot \zeta^{-z}, g\right) \hat{e}(\zeta, u_1) \\
&= \hat{e}\Big(g^{\phi_{\vec{\mu}}(z)} g^{-z\psi_{\vec{\omega}}(\alpha)}, g\Big) \hat{e}\Big(g^{\psi_{\vec{\omega}}(\alpha)}, g^{\alpha}\Big) \\
&= \hat{e}\Big(g^{(\alpha - z)\psi_{\vec{\omega}}(\alpha) + \phi_{\vec{\mu}}(z)}, g\Big) \\
&= \hat{e}\Big(g^{\phi_{\vec{\mu}}(\alpha) - \phi_{\vec{\mu}}(z) + \phi_{\vec{\mu}}(z)}, g\Big) \\
&= \hat{e}\Big(g^{\phi_{\vec{\mu}}(\alpha)}, g\Big),
\end{aligned}$$

the verification equation (4) is satisfied. □

THEOREM 6. *Suppose that the signature scheme $\mathcal{S}_t$ for file tags is existentially unforgeable. The optimized GPoS instantiation is sound for any PPT adversary $\mathcal{A}$ if the s-SDH assumption holds.*

PROOF. Suppose that there is a PPT adversary $\mathcal{A}$ breaking the soundness of the improved GPoS instantiation. Similarly to Theorem 2, at the end of security game, the adversary outputs a (forged) valid pair of challenge/proof (C', R') with regard to a file F' with tag τ' and member id_ℓ, where file F' has been queried for processing. Suppose the challenge $C' = (fid, Q, z, \delta)$ and the proof $R' = (\zeta', \kappa', \sigma')$ Hence, the pair (C', R') satisfies Equation (4). However, since this pair is forged, R' must be unequal to that generated from the maintained information by the simulator. It further means that $(\zeta', \kappa') \neq (\zeta, \kappa)$, since otherwise, σ' would be equal to σ due to Equation (4). Then the simulator obtains two commitments (z, ζ', κ') and (z, ζ, κ) for the same polynomial and both for the evaluation at z. According to the security results [12] of polynomial commitment scheme due to Kate, Zaverucha and Goldberg, the simulator can find a solution $(-z, g^{\frac{1}{\alpha - z}})$ for the s-SDH instance $(\mathbb{G}, u_0, u_1, \cdots, u_{s-1})$ which contradicts the security assumption. □

According to Theorem 3, we have the following corollary.

COROLLARY 3. *The optimized GPoS instantiation is ownership privacy-preserving against the cloud storage server.*

5. PERFORMANCE EVALUATION

In this section, we evaluate and compare the performance of our GPoS instantiations.

5.1 Theoretical Analysis

For comparing the efficiency of both GPoS instantiations (see Section 4), we first analyse their computation costs in each stage. We only consider the most time-consuming computations, i.e., exponentiation and pairing in $\mathbb{G}$, $\mathbb{G}_T$ and $\mathbb{Z}_p$, etc., while the other light-weight operations such as additions and multiplications are omitted. In the table, `H` denotes one hash evaluation for both H and H_0, and `E` represents one exponentiation in $\mathbb{G}$ or $\mathbb{Z}_p$. That is, these evaluations are not discriminated in different groups or ring. The computation times for polynomial long division and a pairing evaluation are denoted by `D` and `P`, respectively. The time to randomly sample an element from a group or a ring is also omitted in the analysis, since it is in fact much less than that taken by an exponentiation. We also treat the digital signature scheme $\mathcal{S}_t$ for file tag as a black-box. Specifically, we let $O(\mathcal{S}_{kgen})$, $O(\mathcal{S}_{sign})$ and $O(\mathcal{S}_{vrfy})$ denote the computation complexity of each algorithm in $\mathcal{S}_t$, that is, $\mathcal{S}_t.\mathsf{KGen}$, $\mathcal{S}_t.\mathsf{Sign}$ and $\mathcal{S}_t.\mathsf{Vrfy}$.

Table 2 summarizes the computational costs of every algorithm of both proposed instantiations. Both GPoS instantiations rely on the same key extracting algorithm to create secret keys for group members. Each key extraction takes one hash evaluation and one exponentiation in $\mathbb{G}$. Regarding the file processing algorithm, the secret values α^i $(2 \leq i \leq s-1)$ can be pre-computed by the group member in the optimized GPoS instantiation, then both instantiations take roughly the same computational complexity for processing a file. There are two ways for producing u_i in optimized GPoS instantiation. That is, either by computing $u_i = u_{i-1}^{\alpha}$ or by raising g to the power of a pre-computed value α^i. Both cases bring the identical complexity for preparing these values u_i. It can be seen from Table 2 that the algorithm PrFile in basic GPoS instantiation requires $(r+s+1)$ exponentiations in $\mathbb{G}$, which determines the overall efficiency for processing a file. In fact, r exponentiations are carried out for producing meta-data for r file blocks, while the other $(s+1)$ exponentiations are due to preparing public parameters $u_0, \cdots, u_s$. Consider a file F of L bytes. If it is split such that each sector has l bytes (as an element in $\mathbb{Z}_p$), then processing this file would take in total

$$T = \left\lceil \frac{L}{s \cdot l} \right\rceil + s + 1 \quad (5)$$

exponentiations. The case for the optimized GPoS instantiation is similar, that is, it would take $T' = T - 1$ exponentiations in $\mathbb{G}$. Furthermore, for processing file F with either GPoS instantiation, a preferable way is to set s as $\sqrt{L/l}$ since it would cost the minimum exponentiation operations.

For auditing an outsourced file in optimized instantiation, the cloud storage sever will carry out a bit more operations than the basic one. This is because that not only the coefficients β_i $(i \in Q)$ should be online calculated through exponentiations, but also an addition polynomial evaluation

for κ, polynomial long division and multi-exponentiation for ζ need to be computed. This would not degrade practicality of the instantiations as the cloud servers are usually assumed to be powerful enough. At the auditor side for verification, the efficiency depends on the parameters $|Q|$ and s. If the auditor challenges a number of file blocks less than s, then the optimized GPoS instantiation is superior to the basic one. Note that most exponentiations taken by the auditor in optimized instantiation are due to computing the coefficients β_i $(i \in Q)$. These computations can be carried out after sending out the challenge C and before receiving the response R from the cloud storage server. In this way, the auditor will take only $((|Q|+1)\mathtt{H}+4\mathtt{E}+3\mathtt{P})$ operations, which is much superior to that of basic instantiation.

We proceed to compare the communication overheads for both instantiations in auditing the integrity of the outsourced files. The details are summarized in Table 3 where S_G denotes the element size of $\mathbb{G}$. In optimized instantiation, the coefficients β_i $(i \in Q)$ are not transmitted directly across the network as in basic instantiation, but generated by both the cloud storage server and the auditor. Thus, these additional computations greatly reduce the communications from the auditor to the cloud server. The similar communication reduction occurs for avoiding directly transmitting the aggregated file block from the cloud server to auditor. As shown in Section 4.2, this is realized by polynomial commitment to commit at a random point z. It can be seen from the Table that, the polynomial commitment trick brings great savings about the overall communication overheads.

Table 3: Communication overheads of auditing integrity in both instantiations

Instantiation	Communication overheads
Section 4.1	$(2\|Q\|+s)l+1S_G$
Section 4.2	$(\|Q\|+3)l+2S_G$

6. CONCLUSION

In this paper, we introduced GPoS which guarantees the integrity of the outsourced group's files in clouds. We first formalized the system framework and the security model for GPoS schemes, and then proposed a generic GPoS construction. The construction is based on the newly identified properties of some digital signature schemes and a trapdoor trick. The effectiveness of this generic construction was showcased with two concrete instantiations over bilinear groups. The second instantiation illustrates extra advantageous features of lower communication overheads for auditing the outsourced files. We provided formal security proof and comprehensive performance evaluations. The analyses show that the proposed scheme and instantiations are secure and practical for real-world applications.

7. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments. This work was supported by the National Key Basic Research Program of China (973) (Grant No. 2012CB315905), National Natural Science Foundation of China (Grant Nos. 61370190, 61173154, 61272455, 61202450, 61472083, 61272501, 61402029, 61472429, and 61202465), and Beijing Natural Science Foundation (Grant Nos. 4132056 and 4122041), the Research Funds of Renmin University of China through project 14XNLF02 and the Open Research Fund of Beijing Key Laboratory of Trusted Computing.

8. REFERENCES

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song. Provable Data Possession at Untrusted Stores. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS'07, pages 598–609, New York, NY, USA, 2007. ACM.

[2] 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 Networks*, SecureComm'08. ACM, 2008.

[3] G. Ateniese, S. Kamara, and J. Katz. Proofs of Storage from Homomorphic Identification Protocols. In M. Matsui, editor, *Advances in Cryptology–ASIACRYPT 2009*, volume 5912 of *LNCS*, pages 319–333. Springer, Heidelberg, 2009.

[4] D. Boneh and X. Boyen. Short Signatures Without Random Oracles and the SDH Assumption in Bilinear Groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[5] D. Cash, A. Küpçü, and D. Wichs. Dynamic Proofs of Retrievability via Oblivious RAM. In T. Johansson and P. Nguyen, editors, *Advances in Cryptology–EUROCRYPT 2013*, volume 7881 of *LNCS*, pages 279–295. Springer, Heidelberg, 2013.

[6] H. Deng, Q. Wu, B. Qin, S. S. M. Chow, J. Domingo-Ferrer, and W. Shi. Tracing and Revoking Leaked Credentials: Accountability in Leaking Sensitive Outsourced Data. In *Proceedings of the 9th ACM Symposium on Information, Computer and Communications Security*, ASIA CCS '14, pages 425–434, New York, NY, USA, 2014. ACM.

[7] H. Deng, Q. Wu, B. Qin, J. Mao, X. Liu, L. Zhang, and W. Shi. Who Is Touching My Cloud. In M. Kutyłowski and J. Vaidya, editors, *Computer Security–ESORICS 2014*, volume 8712 of *LNCS*, pages 362–379. Springer International Publishing, 2014.

[8] Y. Dodis, S. Vadhan, and D. Wichs. Proofs of Retrievability via Hardness Amplification. In O. Reingold, editor, *Theory of Cryptography*, volume 5444 of *LNCS*, pages 109–127. Springer, Heidelberg, 2009.

[9] C. Erway, A. Küpçü, C. Papamanthou, and R. Tamassia. Dynamic Provable Data Possession. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS'09, pages 213–222, New York, NY, USA, 2009. ACM.

[10] R. Gennaro, S. Halevi, and T. Rabin. Secure Hash-and-Sign Signatures Without the Random Oracle. In J. Stern, editor, *Advances in Cryptology–EUROCRYPT'99*, volume 1592 of *LNCS*, pages 123–139. Springer, Heidelberg, 1999.

[11] A. Juels and B. S. Kaliski Jr. PoRs: Proofs of Retrievability for Large Files. In *Proceedings of the 14th ACM Conference on Computer and*

Table 2: Computation costs of each algorithm in both instantiations

Algorithm	Basic instantiation	Optimized instantiation
Setup	$1\mathtt{E}$	$1\mathtt{E}$
KeyExt	$n(1\mathtt{H}+1\mathtt{E})$	$n(1\mathtt{H}+1\mathtt{E})$
PrFile	$r\mathtt{H}+(r+s+1)\mathtt{E}+O(\mathcal{S}_{kgen})+O(\mathcal{S}_{sign})$	$r\mathtt{H}+(r+s)\mathtt{E}+O(\mathcal{S}_{kgen})+O(\mathcal{S}_{sign})$
Chall	$O(\mathcal{S}_{vrfy})$	$O(\mathcal{S}_{vrfy})$
PrfGen	$\lvert Q\rvert\mathtt{E}$	$(2\lvert Q\rvert+2s-3)\mathtt{E}+1\mathtt{D}$
Verify	$(\lvert Q\rvert+1)\mathtt{H}+(s+2)\mathtt{E}+2\mathtt{P}$	$(\lvert Q\rvert+1)\mathtt{H}+(\lvert Q\rvert+4)\mathtt{E}+3\mathtt{P}$

Communications Security, CCS'07, pages 584–597, New York, NY, USA, 2007. ACM.

[12] A. Kate, G. Zaverucha, and I. Goldberg. Constant-Size Commitments to Polynomials and Their Applications. In M. Abe, editor, *Advances in Cryptology–ASIACRYPT 2010*, volume 6477 of *LNCS*, pages 177–194. Springer, Heidelberg, 2010.

[13] H. Shacham and B. Waters. Compact Proofs of Retrievability. *Journal of Cryptology*, 26(3):442–483, 2013.

[14] E. Shi, E. Stefanov, and C. Papamanthou. Practical Dynamic Proofs of Retrievability. In *Proceedings of the 2013 ACM SIGSAC Conference on Computer and Communications Security*, CCS'13, pages 325–336, New York, NY, USA, 2013. ACM.

[15] B. Wang, S. S. M. Chow, M. Li, and H. Li. Storing Shared Data on the Cloud via Security-Mediator. In *Distributed Computing Systems (ICDCS), 2013 IEEE 33rd International Conference on*, pages 124–133, July 2013.

[16] B. Wang, B. Li, and H. Li. Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud. In F. Bao, P. Samarati, and J. Zhou, editors, *Applied Cryptography and Network Security*, volume 7341 of *LNCS*, pages 507–525. Springer, Heidelberg, 2012.

[17] B. Wang, B. Li, and H. Li. Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud. In *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*, pages 295–302, June 2012.

[18] B. Wang, B. Li, and H. Li. Public auditing for shared data with efficient user revocation in the cloud. In *INFOCOM, 2013 Proceedings IEEE*, pages 2904–2912, April 2013.

[19] B. Wang, H. Li, and M. Li. Privacy-preserving public auditing for shared cloud data supporting group dynamics. In *Communications (ICC), 2013 IEEE International Conference on*, pages 1946–1950, June 2013.

[20] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou. Privacy-Preserving Public Auditing for Secure Cloud Storage. *Computers, IEEE Transactions on*, 62(2):362–375, Feb 2013.

[21] H. Wang, Q. Wu, B. Qin, and J. Domingo-Ferrer. Identity-based remote data possession checking in public clouds. *Information Security, IET*, 8(2):114–121, March 2014.

[22] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou. Enabling Public Verifiability and Data Dynamics for Storage Security in Cloud Computing. In M. Backes and P. Ning, editors, *Computer Security–ESORICS 2009*, volume 5789 of *LNCS*, pages 355–370. Springer, Heidelberg, 2009.

[23] Y. Wang, Q. Wu, D. S. Wong, B. Qin, S. S. M. Chow, Z. Liu, and X. Tan. Securely Outsourcing Exponentiations with Single Untrusted Program for Cloud Storage. In M. Kutyłowski and J. Vaidya, editors, *Computer Security–ESORICS 2014*, volume 8712 of *LNCS*, pages 326–343. Springer International Publishing, 2014.

[24] J. Xu and E.-C. Chang. Towards Efficient Proofs of Retrievability. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security*, ASIACCS '12, pages 79–80, New York, NY, USA, 2012. ACM.

[25] K. Yang and X. Jia. Data storage auditing service in cloud computing: challenges, methods and opportunities. *World Wide Web*, 15(4):409–428, 2012.

[26] Y. Yu, Y. Mu, J. Ni, J. Deng, and K. Huang. Identity Privacy-Preserving Public Auditing with Dynamic Group for Secure Mobile Cloud Storage. In M. H. Au, B. Carminati, and C. C. J. Kuo, editors, *Network and System Security*, volume 8792 of *LNCS*, pages 28–40. Springer International Publishing, 2014.

[27] J. Yuan and S. Yu. Proofs of Retrievability with Public Verifiability and Constant Communication Cost in Cloud. In *Proceedings of the 2013 International Workshop on Security in Cloud Computing*, Cloud Computing '13, pages 19–26, New York, NY, USA, 2013. ACM.

[28] J. Yuan and S. Yu. Efficient public integrity checking for cloud data sharing with multi-user modification. In *INFOCOM, 2014 Proceedings IEEE*, pages 2121–2129, April 2014.

[29] Y. Zhang and M. Blanton. Efficient Dynamic Provable Possession of Remote Data via Balanced Update Trees. In *Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security*, ASIA CCS'13, pages 183–194, New York, NY, USA, 2013. ACM.

[30] 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*, CODASPY '12, pages 1–12, New York, NY, USA, 2012. ACM.