Storing Shared Data on the Cloud via Security-Mediator

Boyang Wang[†§], Sherman S. M. Chow[‡], Ming Li[§], and Hui Li[†]
[†]*State Key Laboratory of Integrated Service Networks, Xidian University, Xi'an, China*
[‡]*Department of Information Engineering, Chinese University of Hong Kong, Hong Kong*
[§]*Department of Computer Science, Utah State University, Logan, Utah, USA*
Email: bywang@mail.xidian.edu.cn, smchow@ie.cuhk.edu.hk, ming.li@usu.edu, lihui@mail.xidian.edu.cn

***Abstract*—Nowadays, many organizations outsource data storage to the cloud such that a member of an organization (data owner) can easily share data with other members (users). Due to the existence of security concerns in the cloud, both owners and users are suggested to verify the integrity of cloud data with Provable Data Possession (PDP) before further utilization of data. However, previous methods either unnecessarily reveal the identity of a data owner to the untrusted cloud or any public verifiers, or introduce significant overheads on verification metadata for preserving anonymity.**

In this paper, we propose a simple, efficient, and publicly-verifiable approach to ensure cloud data integrity without sacrificing the anonymity of data owners nor requiring significant overhead. Specifically, we introduce a security-mediator (SEM), which is able to generate verification metadata (i.e., signatures) on outsourced data for data owners. Our approach decouples the anonymity protection mechanism from the PDP. Thus, an organization can employ its own anonymous authentication mechanism, and the cloud is oblivious to that since it only deals with typical PDP-metadata, Consequently, the identity of the data owner is not revealed to the cloud, and there is no extra storage overhead unlike existing anonymous PDP solutions. The distinctive features of our scheme also include data privacy, such that the SEM does not learn anything about the data to be uploaded to the cloud at all, and thus the trust on the SEM is minimized. In addition, we extend our scheme to work with the multi-SEM model, which can avoid the potential single point of failure. Security analyses prove that our scheme is secure, and experiment results demonstrate that our scheme is efficient.

***Keywords*-Cloud computing, shared data, security-mediator, data integrity, anonymity, provable data possession**

I. Introduction

Nowadays, many organizations outsource their large-scale data storage to the cloud for saving the cost in maintaining in-house storage. With cloud storage service, the members of an organization can share data with other members easily by uploading their data to the cloud. Examples of organizations which may benefit from this cloud storage and sharing service are numerous, such as international enterprises with many employees around the world, collaborative web application providers with a large user base, or institutions dealing with big data, healthcare service providers coordinating medical data from doctors, researchers, patients, etc. While the economic benefits brought by outsourcing data can be attractive, security is one of the most significant factors that hinders its wide development [1]. Since data operations in the cloud are not transparent to users, and security breaches or improper practices are common and inevitable, users still have a huge concern about the security of their data on the cloud, especially on data integrity [2].

Generally speaking, to maintain data integrity, data owners can compute verification metadata on their data, then upload both of them to the cloud. These verification metadata should be verifiable not only by the data owner herself but preferably also by any public verifiers [3], [4]. For instance, medical researchers need to make sure health records contributed by data owners are free from any errors which may be introduced by malpractice of the cloud or communication error before their analyses.

However, the general method for protecting data integrity will conflict with another significant concern of data owners — *identity privacy*, or *anonymity*. Specifically, if digital signatures are used to serve as verification metadata, they can only be verified with a data owner's public key. The unique binding between a public key and the identity of the data owner will inevitably reveal the owner's identity to any public verifiers [5], especially under public key infrastructure where such bindings are explicit via public-key certificates.

Refer to the sample scenario above, a patient may agree to let medical researchers to analyze her health record stored in the cloud for research purposes, but is often unwilling to reveal her real identity. Note that the existing techniques of Onion Routing (e.g., Tor [6]) can help ensuring anonymity for data owners when they upload or download their cloud data, but it is not sufficient to hide the identities of data owners from public verifiers during the data integrity checking based on public keys and certificates.

Besides protecting the anonymity of data owners, another important issue is how to allow integrity checking without requiring the entire data. Since the size of cloud data could be huge, if a verifier must need to download the entire data to verify, a major benefit of cloud storage is lost, and this verifier will waste a huge amount of computation and communication resource, especially when the downloaded data has actually been tampered with. Provable data possession (PDP) proposed by Ateniese *et al.* [3] is a protocol which allows the integrity of data stored in an un-trusted server to be audited without retrieving the entire data. Many schemes [3], [7], [8], [9], [10], [11], [4], [12] also allow public verifiers to perform such efficient data integrity checking. Unfortunately, none of these schemes is able to preserve the

identities of data owners from public verifiers.

Recently, some cryptographic schemes [5], [13] extend the basic PDP such that they are able to hide the identities of data owners within an organization (or a group) from any verifiers in the *multi-owner* scenario, where data files can be owned and modified by a number of d ($d \geq 2$) members. However, both of these two schemes introduce significant overheads to ensure anonymity. Specifically, the ring-signature-based scheme [5] produces verification metadata of size increases linearly with the number of members in a group, which is quite inefficient when the size of a group is large. While the size of verification metadata is independent of the group size in the group-signature-based scheme [13], it is still much larger than its counterpart in traditional non-anonymous PDP solutions. In addition, their final scheme [13] even fails to support public verifiability[1].

In this paper, we propose a simple, efficient, and publicly verifiable approach to ensure cloud data integrity, without compromising the anonymity of data owners nor introducing significant overhead to verification metadata. The major benefit of our approach is the decoupling of anonymity protection mechanism from the PDP itself. In other words, the protection of data owners' anonymity incurs *no extra cost* for cloud service providers or any public verifiers. A key observation we made is that the anonymity (here, and [5], [13]) is defined within the group of users of an organization. A direct approach is to have a *security-mediator* (SEM) from the organization to sign on behalf of its all members. While it appears to be conceptually simple in the first place, we still need to address a number of concerns for our final solution.

First, it should outperform the trivial approach of requiring all members to first upload all their data to this SEM. Second, the trust on this SEM should be minimized. In particular, this SEM should not learn anything about the uploaded data even though it is responsible for generating signatures on data. Moreover, from the signing request, the SEM should not be able to know who are the uploaders, nor distinguish their identities based on the data and corresponding signatures. With these low computational and bandwidth requirements, and the low trust level, a typical server of the organization can serve as the SEM.

We believe that the introduction of a security mediator for an organization is a right approach for our motivating canonical scenario of storing shared data on the cloud. From the cloud's perspective, it is the organization who pays for the storage service, so it is natural for the cloud to accept uploading requests when a valid signature issued by the organization is presented. The use of SEM, as described previously, provides *non-revokable anonymity* to the members of the organization, which may be abused by misbehaving members in some cases. However, the organization can always incorporate an external anonymous authentication mechanism (e.g., [14], [15]) on top of our system to strike a right balance of anonymity and accountability. It is also better to have the SEM maintained by the organization itself since it is of the organization's interests to control who can use the data storage on its paid cloud service. In this way, less trust is placed by the organization on the cloud.

As argued above, the role of a SEM can be played by a typical server of an organization which is used for the daily authentication of its members. However, one important difference from a typical authentication server is that it owns the private signing key corresponding to the organization's public key recognized by the cloud service provider and the rest of the world. It is thus important to lower this level of trust by distributing the knowledge of this private key to multiple parties. In addition, this also avoids the bottleneck of a single SEM and the potential single point of failure. Therefore, we further extend our scheme to the *multi-SEM* model with the technique of (w, t)-Shamir secret sharing [16], where $w = 2t - 1$. In the multi-SEM model with a total number of w SEMs, even when a maximum number of $(t-1)$ SEMs are unable to sign data normally, data owners can still obtain valid signatures from the rest of SEMs.

The rest of this paper is organized as follows. We present the system model, security and privacy threats, design objectives, and overview of our design in Section II. We introduce the definition and properties of some techniques we utilized in the design of our scheme in Section III. The details of our scheme and security analyses are described in Sections IV and V. Experiment results are shown in Section VI. Finally, we briefly discuss related work in Section VII, and conclude this paper in Section VIII.

II. Model, Objectives, and Our Design

A. System Model

The system model introduced in this paper consists of four entities, including data owners, data users, the cloud server, and a security-mediator (SEM), which are described in Figure 1. Data owners generate data and upload them to the cloud for sharing. We will first consider the single-owner scenario, where each data file stored in the cloud is managed and modified by only a single data owner. The multi-owner scenario will be discussed later in this paper. Data users are able to access data uploaded by data owners, but are not allowed to modify data in the cloud. Data owners and data users are sometimes collectively termed as cloud users in this paper. The cloud server provides data storage and sharing services to data owners and data users. Both the cloud server and data users are public verifiers, who are not the owner of data but need to verify data integrity when it is necessary. The SEM provides security services to data owners by generating signatures on data for owners before these data are outsourced to the cloud.

[1]While group signatures are publicly verifiable in general, a direct combination of group signatures and PDP techniques will make the size of verification metadata even larger than the size of data itself [13]. In order to minimize this overhead, an additional private key is shared between the data owner and the verifier in order to realize functionalities similar to "homomorphic MAC", which scarifies public verifiability.

We also make the standard assumption (e.g., [10]) that the communication channel between the cloud server and the verifier is authenticated. After all, one of the major purposes for using our system and other existing PDP solutions is to ensure that the cloud users can obtain cryptographic evidences when the cloud server fail to provide a reliable storage service.

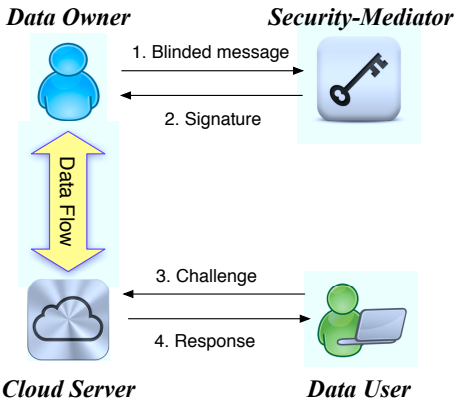


Figure 1. The system model includes data owners, data users, the cloud server, and a security-mediator (SEM).

B. Security Threats and Design Objectives

Due to the existence of external and internal attacks in the cloud, cloud users remain concerned about the integrity of their data in the cloud. On the other hand, data owners want to preserve not only their identity privacy but also data privacy when they create the data to be shared. To address the above security and privacy threats, our system should achieve the following properties:

- **Public Verifiability**. The integrity of cloud data (outsourced by data owners) should be verifiable by a public verifier.
- **Verification Efficiency**. A public verifier, especially who does not possess cloud data, should be able to verify the integrity of cloud data without retrieving the entire data from the cloud server.
- **Unforgeability**. The verification metadata (signatures) for ensuring data integrity should be existentially unforgeable under adaptive chosen-message attack.
- **Anonymity**. The identity of a data owner should not be revealed to a public verifier during the verification of data integrity. In addition, a SEM should not be able to reveal the identity of a data owner based on cloud data and corresponding signatures.
- **Data Privacy**. During the generation of signatures for a data owner, other parties, even a SEM, should not be able to learn the content of data that the data owner wants to sign.
- **Signing Efficiency**. The communication requirement between a SEM and a data owner during signature generation should be smaller than directly transferring the data to be signed.

Note that how the access control of the data will be enforced by the cloud is an orthogonal issue which can be dealt with existing solution, e.g., [14], [15]. Also, the communication channel between the SEM and a data owner can be chosen to provide various level of anonymity, e.g., over an anonymous network (such as Tor [6]). The SEM can authenticate each data owner by anonymous credential supporting both revocation and reputation, e.g., [17].

C. Solution Overview

Unlike traditional PDP schemes for secure cloud storage, which require a data owner to generate signatures by herself, our scheme requires the data owner to first obtain signatures on her data with the help of the SEM. More specifically, she first divides her data into many small blocks, processes every block with blinding techniques, and then sends the blinded version of these blocks to the SEM. The SEM then computes a blind signature with its private key, and returns this blind signature back to the data owner. Then, the data owner transforms this blind signature to obtain the actual signature on the original block. Finally, the data owner uploads all her data and corresponding signatures to the cloud server. Dividing cloud data into small blocks and generating a signature on every one of them will enable a public verifier to check data integrity by retrieving only a random combination of all the blocks instead of retrieving the entire cloud data from the cloud server, which is the typical approach in PDP schemes.

When a public verifier checks the integrity of cloud data based on its signatures, this public verifier only learns the cloud data is signed by the SEM, which proves that cloud data is correct, but has no means to reveal the identity of the data owner. Due to the properties of blind signatures, even the SEM is not able to reveal the content of data it blindly signed before, or link a data owner to the data or signatures stored on the cloud.

III. Preliminaries

A. Bilinear maps

Let $\mathbb{G}_1$ and $\mathbb{G}_2$ be two multiplicative cyclic groups of prime order p, g be a generator of $\mathbb{G}_1$. Bilinear map $\mathtt{e}$ is a map $\mathtt{e}$: $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_2$ with the following properties: (1) *Computability*: map $\mathtt{e}$ can be efficiently computed. (2) *Bilinearity*: for all $u, v \in \mathbb{G}_1$ and $a, b \in \mathbb{Z}_p$, $\mathtt{e}(u^a, v^b) = \mathtt{e}(u, v)^{ab}$. (3) *Non-degeneracy*: $\mathtt{e}(g, g) \neq 1$.

B. Blind Signatures

Blind signatures, first proposed by Chaum [18], form a special type of signatures where the message owner and the signer are different parties. More specifically, the message owner choose a blinding factor to blind the content of her message and sends the blinded message to the signer. After received the blinded message, the signer generates a signature on the blinded message and returns it to the message owner. The message owner is able to recover and

output a regular signature on the original message based on the result returned by the signer and the blinding factor.

The *blindness* properties requires that the signer cannot learn the content of the original message during the generation of a signature. For *unlinkability*, it requires that the signer cannot link a blinded message/signature to its corresponding unblinded form.

C. Shamir Secret Sharing

A (w, t)-Shamir secret sharing scheme [16], where $w = (2t-1)$, is able to divide a secret s into w pieces in such a way that this secret s can be easily recovered from any t pieces, while the knowledge of any $(t-1)$ pieces reveals absolutely no information about this secret s.

The essential idea of a (w, t)-Shamir secret sharing scheme is that, a number of t points define a polynomial of degree $(t-1)$. Suppose we want to share the secret $s \in \mathbb{Z}_p$. We set $a_0 = s$ and define the following polynomial

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + a_0, \tag{1}$$

by picking $a_{t-1}, \cdots, a_1$ uniformly at random from $\mathbb{Z}_p$. Each piece of the share is actually a point of polynomial $f(x)$, for example, $(x_i, f(x_i))$. The secret s can be recovered by at least a number of t points of polynomial $f(x)$ with Lagrange polynomial interpolation. Shamir secret sharing is generally used in key management schemes [16] and secure multi-computation [19], [20].

IV. Our Solution

A. Overview

In our solution, it is the SEM who will generate signatures on all the message blocks of a data owner. To do that, a data owner first sends a blinded version of the block to the SEM, obtain a signature on the blinded block, and eventually recover the actual signature on this block. To reduce the communication overhead and improve the efficiency of the signing services, we allow a data owner to blind an aggregation of multiple data elements in a block. Specifically, suppose a block is denoted as $\boldsymbol{m}_i = (m_{i,1}, \cdots, m_{i,k})$, where k is the number of data elements in a block, then the size of an aggregate value and the size of the blinded version of an aggregate value are both only $1/k$ of the size of this block. Unfortunately, we cannot arbitrarily increase the value of k since it will increase the communication overhead during public verification. The trade-off involved will be discussed later in Section IV-C.

B. Scheme Details

Our scheme includes seven algorithms, including **`Setup`**, **`Blind`**, **`Sign`**, **`Unblind`**, **`Challenge`**, **`Response`**, and **`Verify`**. Global parameters (implicit input of all other algorithms), public keys and private keys are generated in **`Setup`**. A data owner is able to generate signatures on her data with the help of the SEM after performing **`Blind`**, **`Sign`**, and **`Unblind`**. A public verifier is able to challenge and verify the integrity of data stored in the cloud by interacting with the cloud server in **`Challenge`**, **`Response`**, and **`Verify`**. The details of each algorithm are described as follows:

`Setup`: Given a security parameter λ, outputs global parameters as $(\mathbb{G}_1, \mathbb{G}_2, \mathtt{e}, p, g, H, u_1, \cdots, u_k)$, where $\mathbb{G}_1$, $\mathbb{G}_2$ are two multiplicative groups of prime order p (with $|p|$, the size of p, determined by λ), g is a generator of $\mathbb{G}_1$, $\mathtt{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ is a bilinear map, $H : \{0,1\}^* \to \mathbb{G}_1$ is a is hash function, $(u_1, \cdots, u_k)$ are k random $\mathbb{G}_1$ elements.

Data $\mathbf{M}$, which will be outsourced by a data owner, is divided into n blocks as $\mathbf{M} = (\boldsymbol{m}_1, \cdots, \boldsymbol{m}_n)$, and each block $\boldsymbol{m}_i = (m_{i,1}, \cdots, m_{i,k})$ contains k elements of $\mathbb{Z}_p$. The SEM generates its private key as $\mathtt{sk} = y \in \mathbb{Z}_p$ and public key as $\mathtt{pk} = g^y \in \mathbb{G}_1$.

`Blind`: Given a block $\boldsymbol{m}_i = (m_{i,1}, \cdots, m_{i,k})$, its block identifier id_i, a data owner first generates a random factor $r \in \mathbb{Z}_p$, then blinds block $\boldsymbol{m}_i$ into a blinded message $\widehat{\boldsymbol{m}}_i$ by

$$\widehat{\boldsymbol{m}}_i = [H(id_i) \prod_{l=1}^{k} u_l^{m_{i,l}}] \cdot g^r \in \mathbb{G}_1. \tag{2}$$

This blinded message $\widehat{\boldsymbol{m}}_i$ will then be sent to the SEM.

`Sign`: Given a blinded message $\widehat{\boldsymbol{m}}_i$, private key $\mathtt{sk} = y$, the SEM computes a signature on blinded message $\widehat{\boldsymbol{m}}_i$ by

$$\widehat{\sigma}_i = (\widehat{\boldsymbol{m}}_i)^y \in \mathbb{G}_1, \tag{3}$$

and returns this signature $\widehat{\sigma}_i$ back to the data owner.

`Unblind`: Given blinded message $\widehat{\boldsymbol{m}}_i$, its signature $\widehat{\sigma}_i$, the random factor r used in **`Blind`**, and public key $\mathtt{pk} = g^y$, a data owner first checks the correctness of signature $\widehat{\sigma}_i$ by

$$\mathtt{e}(\widehat{\sigma}_i, g) \stackrel{?}{=} \mathtt{e}(\widehat{\boldsymbol{m}}_i, \mathtt{pk}). \tag{4}$$

If it does not hold, the data owner discards this signature $\widehat{\sigma}_i$ and asks the SEM to generate a new one. Otherwise, the data owner obtains a signature σ_i on block $\boldsymbol{m}_i$ by

$$\sigma_i = \widehat{\sigma}_i \cdot \mathtt{pk}^{-r} = [H(id_i) \prod_{l=1}^{k} u_l^{m_{i,l}}]^y \in \mathbb{G}_1. \tag{5}$$

The correctness of Equation 4 is presented as follows:

$$\mathtt{e}(\widehat{\sigma}_i, g) = \mathtt{e}(\widehat{\boldsymbol{m}}_i^y, g) = \mathtt{e}(\widehat{\boldsymbol{m}}_i, g^y) = \mathtt{e}(\widehat{\boldsymbol{m}}_i, \mathtt{pk}).$$

The correctness of Equation 5 is easy to see. Finally, after the computation of all the signatures on all the blocks, the data owner outsources data $\mathbf{M} = (\boldsymbol{m}_1, \cdots, \boldsymbol{m}_n)$ and its corresponding signatures $(\sigma_1, \cdots, \sigma_n)$ to the cloud.

`Challenge`: A public verifier generates a challenge $\mathbf{C} = \{id_i, \beta_i\}_{i \in \mathcal{I}}$, where $\{\mathcal{I} : i \in \mathcal{I} | 1 \le i \le n\}$ is the set of all the n blocks' indices[2] and β_i is a random element of $\mathbb{Z}_p$.

`Response`: Given a challenge $\mathbf{C}$, the outsourced data $\mathbf{M}$ and its signatures $(\sigma_1, \cdots, \sigma_n)$, the cloud

[2]The size of this set is a tuneable parameter. The trade-off involved will be discussed in Section IV-C.

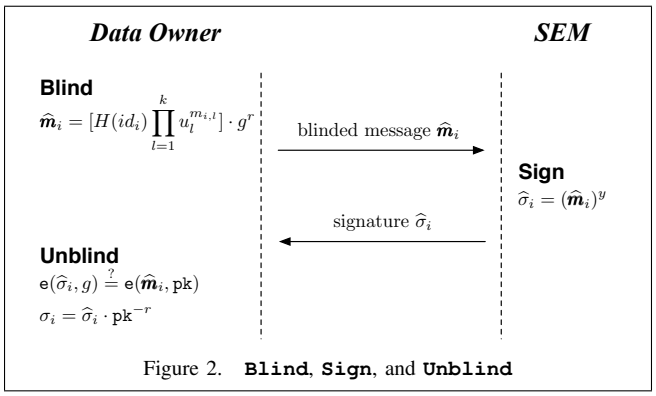


Figure 2. **Blind**, **Sign**, and **Unblind**

1) computes $\sigma^* = \prod_{i\in\mathcal{I}} \sigma_i^{\beta_i} \in \mathbb{G}_1$,
2) computes $\alpha_l = \sum_{i\in\mathcal{I}} \beta_i m_{i,l} \in \mathbb{Z}_p$, where $l \in [1, k]$,

and returns a response $\mathbf{R} = \{\sigma^*, \alpha_1, \cdots, \alpha_k\}$ to the verifier.

Verify: Given challenge $\mathbf{C} = \{id_i, \beta_i\}_{i\in\mathcal{I}}$, response $\mathbf{R} = \{\sigma^*, \alpha_1, \cdots, \alpha_k\}$, and public key pk, a public verifier checks the integrity of data $\mathbf{M}$ by

$$\mathtt{e}(\sigma^*, g) \stackrel{?}{=} \mathtt{e}(\prod_{i\in\mathcal{I}} H(id_i)^{\beta_i} \cdot \prod_{l=1}^{k} u_l^{\alpha_l}, \mathtt{pk}). \quad (6)$$

If the above equation holds, outsourced data $\mathbf{M}$ is correctly stored in the cloud. Otherwise, data $\mathbf{M}$ is not correct.

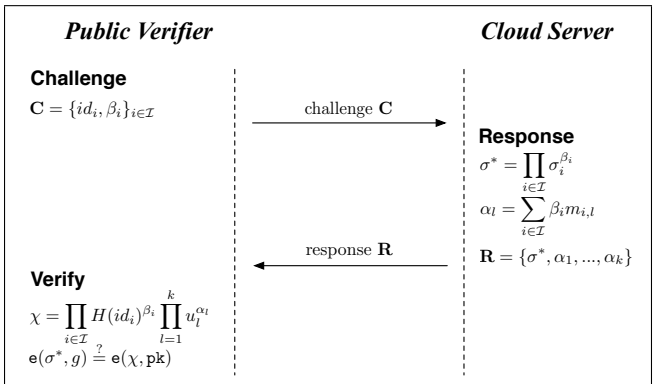


Figure 3. **Challenge**, **Response**, and **Verify**

The correctness of public verification is presented as

$$\begin{aligned}
\mathtt{RHS} &= \mathtt{e}(\prod_{i\in\mathcal{I}} H(id_i)^{\beta_i} \cdot \prod_{l=1}^{k} u_l^{\sum_{i\in\mathcal{I}} \beta_i m_{i,l}}, \mathtt{pk}) \\
&= \mathtt{e}(\prod_{i\in\mathcal{I}} H(id_i)^{\beta_i} \cdot \prod_{i\in\mathcal{I}} \prod_{l=1}^{k} u_l^{\beta_i m_{i,l}}, g^y) \\
&= \mathtt{e}(\prod_{i\in\mathcal{I}} [H(id_i) \prod_{l=1}^{k} u_l^{m_{i,l}}]^{y\beta_i}, g) \\
&= \mathtt{e}(\sigma^*, g)
\end{aligned}$$

where RHS denotes the right hand side of Equation 6.

C. Discussions

Performance Optimization. There are several methods can be used to further improve the efficiency of our scheme.

- *Batch Verification in* **Unblind**. When checking the correctness of n signatures $(\widehat{\sigma}_1, \cdots, \widehat{\sigma}_n)$ from the SEM, the data owner can perform batch verification by checking their correctness simultaneously instead of verifying them one by one. Specifically, the data owner can verify the n signatures $(\widehat{\sigma}_1, \cdots, \widehat{\sigma}_n)$ by

$$\mathtt{e}(\prod_{i=1}^{n} \widehat{\sigma}_i^{\gamma_i}, g) \stackrel{?}{=} \mathtt{e}(\prod_{i=1}^{n} \widehat{\boldsymbol{m}}_i^{\gamma_i}, \mathtt{pk}), \quad (7)$$

  where $\gamma_i \in \mathbb{Z}_p$, for $1 \le i \le n$, are random numbers generated by the data owner. The correctness of batch verification above can be easily explained based on the properties of bilinear maps.
- *Parameter* k. As shown in our scheme, a block $\boldsymbol{m}_i = (m_{i,1}, \cdots, m_{i,k})$ includes k elements of $\mathbb{Z}_p$ and the signature storage overhead of a block is $1/k$. Clearly, the data owner can choose a larger value of k to save the total signature storage overhead in the cloud. As a necessary trade off, the signature generation time, and also communication overhead during public verification, will linearly increase with an increase of k.
- *Sampling Strategies*. During public verification, a public verifier can randomly select a smaller number of blocks instead of all the n blocks, and is still able to detect the corrupted data with a high probability [3].
- *Small Exponentiations*. Random β_i can be selected from $\mathbb{Z}_q$ instead of $\mathbb{Z}_p$, where q is much smaller than p. One may refer to the discussion in [21] for relationship between the size of q and the probability of accepting a bad signature.

Data Privacy. Data privacy is also an important property that a secure cloud data storage scheme should support. Generally speaking, the content of data should be encrypted by the data owner before data is outsourced to the cloud server. In our scheme, to protect data privacy, a data owner can first encrypt a block in her data by $\boldsymbol{m'} = \mathtt{Enc}_{key}(\boldsymbol{m})$ using any symmetric key encryption, and follows algorithm **Blind**, **Sign**, **Unblind** to obtain a signature σ' on the encrypted block $\boldsymbol{m}'$. After receiving all the signatures on all encrypted blocks, the data owner outsources the encrypted version of her data and corresponding signatures to the cloud.

Other Features with Public Verification. During public verification, supporting other features, such as data dynamic [8], [9], are also believed to be important problems. Since the verification process of our scheme and these schemes are all based on BLS signatures [22], by leveraging similar techniques from these schemes [8], [9], data dynamic can be easily supported by our scheme without affecting the security and privacy of our current scheme. Since the main purpose of our scheme is to provide identity privacy for the

data owner, we do not describe the details of how to enable data dynamic with our scheme in this paper.

Multi-Owner Scenario. In many applications, cloud data is maintained by multiple owners. For example, a group of users can edit the same file by using online collaborative tools. As discussed in previous work, protecting the identity of the owner on each block in cloud data is also very important to the privacy of this group and cloud data itself [5], [13]. For instance, without protecting anonymity on multi-owner data, a public verifier may easily distinguish a particular owner is a more important member in the group than other group members because this owner maintains and signs the largest number of blocks in cloud data; a particular block can be indicated as a more important one than other blocks in the same data file because this block is frequently modified and signed by different group members.

Clearly, with our scheme, the identity privacy of group members under the multi-owner scenario can be preserved from a public verifier by asking all the members in the same group to generate signatures on cloud data with a same public/private key pair managed by the SEM. For a public verifier, it is convinced all the blocks in cloud data is signed by the SEM, but cannot distinguish any particular member or block during the verification of data integrity.

Dynamic Groups and Instant Revocation Support. Compared to previous schemes [5], [13], which are also able to preserve the identity of the owner on each block from a public verifier under the multi-owner scenario, another advantage of our scheme is its easy support of dynamic groups, where new users can be added into a group and existing members can be revoked from a group.

More specifically, once a new user joins the group, the group manager will instruct the SEM to add this new user into the group member list, and the SEM will provide signing services to this new user as other members in the group; if an existing member is revoked from the group, the group manager will instruct the SEM to remove this member from the group member list, and the SEM will no longer provide signing services to the revoked user. In this way, our scheme can still maintain identity privacy for dynamic group without the need of recomputing any signatures on cloud data. For the previous schemes [5], [13], once the membership of the group is changed, all the signatures on cloud data need to be recomputed to maintain identity privacy, which is quite inefficient.

D. Security Analysis

Since the blind signatures [23] we used are essentially extended from BLS signatures [22]. It is easy to see that the final signature resulted from the signing protocol, and the challenge-response protocol all follow exactly from the scheme in [7], which is also based on BLS signatures [22]. This clearly ensures the public verifiability for data integrity.

The identity privacy of a data owner is straightforward since all the data to be used in integrity verification is originated from the same private key of the SEM. Data privacy and anonymity during the generation of signatures follow exactly as the argument in [23]. In particular, both our scheme and the blind signature scheme in [23] blind the message with the same mechanism involving a random factor r. For every valid signature, it is always possible to find the corresponding r to form a "matching" communication transcript, and hence the signature and the communication transcript during its generation are unlinkable.

Finally, unforgeability can be proven in two steps. Firstly, the resulting signature is in the same form produced by the scheme in [7]. Moreover, the blind signing protocol can be proven to be unforgeable under the one-more discrete-logarithm assumption, similar to the proof of unforgeability for the scheme in [23].

V. Our Scheme in Multi-Signer Model

A. Scheme Overview

To avoid the possible single point failure and improve the reliability and security of our scheme, we can further extend our scheme to work with multiple SEMs. Specifically, we utilize a (w, t)-Shamir secret sharing scheme to distribute the private key $\mathtt{sk}$, which is used to compute signatures for data owners, into w pieces as $(\mathtt{sk}_1, \cdots, \mathtt{sk}_w)$, and share each piece with a SEM. When a data owner wishes to generate a signature on a block $\boldsymbol{m}$, it sends the blinded version of this block to every SEM, and each SEM is able to compute a share of the blind signature with its own piece. Once the data owner obtains t valid shares of blind signature from t SEMs, she is able to recover the signature on her original block $\boldsymbol{m}$ by using Lagrange polynomial interpolation.

B. Scheme Details

Similar to our scheme in the single-SEM model, the one in multi-SEM model also consists of seven corresponding algorithms, including **Setup**$^\diamond$, **Blind**$^\diamond$, **Sign**$^\diamond$, **Unblind**$^\diamond$, **Challenge**$^\diamond$, **Response**$^\diamond$, and **Verify**$^\diamond$. The details of our scheme in multi-SEM model are presented as below.

Setup$^\diamond$: A selected party acting as the manager of SEMs generates the private key $\mathtt{sk} = y \in \mathbb{Z}_p$ and public key $\mathtt{pk} = g^y \in \mathbb{G}_1$. This manager randomly generates the following polynomial of degree $(t-1)$:

$$f(x) = a_{t-1}x^{t-1} + \cdots + a_1 x + a_0 \tag{8}$$

where $a_0 = y$. Then, this manager computes w points of $f(x)$ by $(x_1, y_1), (x_2, y_2), \cdots, (x_w, y_w)$, where $y_j = f(x_j)$ for $1 \leq j \leq w$, and securely distributes each point as a share of $\mathtt{sk}$ to each SEM. The private key of SEM S_j is $\mathtt{sk}_j = y_j$, and its public key is $\mathtt{pk} = g^{y_j}$. After the distribution of all the w points, the manager can go offline.

Blind$^\diamond$: The blinding process of a block $\boldsymbol{m}_i$ is the same as in algorithm **Blind**. The only difference is that, after the blinding, the data owner sends the blinded message $\widehat{\boldsymbol{m}}$ to all the w SEMs instead of only one SEM.

Sign$^{\diamond}$: Given a blinded message $\widehat{\boldsymbol{m}}_i$, private key $\mathtt{sk}_j = y_j$, SEM S_j computes a share of blind signature on $\widehat{\boldsymbol{m}}_i$ by

$$\widehat{\sigma}_{i,j} = (\widehat{\boldsymbol{m}}_i)^{y_j} \in \mathbb{G}_1, \tag{9}$$

and returns $\widehat{\sigma}_{i,j}$ back to the data owner.

Unblind$^{\diamond}$: Given blinded message $\widehat{\boldsymbol{m}}_i$, a share of blind signature $\widehat{\sigma}_i$, the random factor r used in **Blind**$^{\diamond}$, and public key $\mathtt{pk}_j = g^{y_j}$, a data owner first checks the correctness of $\widehat{\sigma}_{i,j}$ by

$$\mathtt{e}(\widehat{\sigma}_{i,j}, g) \stackrel{?}{=} \mathtt{e}(\widehat{\boldsymbol{m}}_i, \mathtt{pk}_j). \tag{10}$$

If this equation does not hold, the data owner discard this share of blind signature $\widehat{\sigma}_{i,j}$ and asks SEM S_j to generate a new one. Otherwise, the data owner believes this share of blind signature is correctly given by SEM S_j.

After the data owner receives t correct shares of blind signature (without loss of generality, we assume these t shares are $\widehat{\sigma}_{i,1}, \cdots, \widehat{\sigma}_{i,t}$), the data owner computes

$$L_j(0) = \prod_{0<l\leq t, l\neq j} \frac{-x_l}{x_j - x_l}, 1 \leq j \leq t, \tag{11}$$

where Lagrange basis polynomial $L_j(0)$ is independent of polynomial $f(x)$, and can be pre-computed. Then, the data owner computes blind signature $\widehat{\sigma}_i$ on $\widehat{\boldsymbol{m}}_i$ by

$$\widehat{\sigma}_i = \prod_{j=1}^{t} \widehat{\sigma}_{i,j}^{L_j(0)} \in \mathbb{G}_1 \tag{12}$$

The correctness of the above equation can be explained by the correctness of Lagrange polynomial interpolation:

$$\begin{aligned} \prod_{j=1}^{t} \widehat{\sigma}_{i,j}^{L_j(0)} &= \prod_{j=1}^{t} [\widehat{\boldsymbol{m}}_i^{y_j}]^{L_j(0)} = \widehat{\boldsymbol{m}}_i^{\sum_{j=1}^{t} L_j(0) y_j} \\ &= \widehat{\boldsymbol{m}}_i^{\sum_{j=1}^{t} L_j(0) f(x_j)} = \widehat{\boldsymbol{m}}_i^{f(0)} = \widehat{\boldsymbol{m}}_i^{y} \\ &= \widehat{\sigma}_i. \end{aligned}$$

Finally, the data owner recovers the actual signature σ_i on block $\boldsymbol{m}_i = (m_{i,1}, \cdots, m_{i,k})$ by

$$\sigma_i = \widehat{\sigma}_i \cdot \mathtt{pk}^{-r}, \tag{13}$$

which is the same as Equation 5 in algorithm **Unblind**. After obtained all the n signatures $(\sigma_1, \cdots, \sigma_n)$ on data $\mathbf{M} = (\boldsymbol{m}_1, \cdots, \boldsymbol{m}_n)$, the data owner outsources the entire data $\mathbf{M}$ and all n the signatures together to the cloud server.

Challenge$^{\diamond}$, **Response**$^{\diamond}$ and **Verify**$^{\diamond}$: Since the final signature remains the same no matter in the single-SEM model or multi-SEM model, a public verifier is able to check the integrity of cloud data in the same way as in the single-SEM model by following algorithms **Challenge**, **Response**, and **Verify**.

Similar to algorithm **Unblind**, we can also adopt batch verification in algorithm **Unblind**$^{\diamond}$ to improve the efficiency of our scheme in the multi-SEM model. More concretely, a data owner can verify all the $2n \cdot t$ shares of blind signature with the following equation

$$\mathtt{e}(\prod_{i=1}^{n}\prod_{j=1}^{t} \widehat{\sigma}_{i,j}, g) \stackrel{?}{=} \prod_{j=1}^{t} \mathtt{e}(\prod_{i=1}^{n} \widehat{\boldsymbol{m}}_i, \mathtt{pk}_j), \tag{14}$$

which can reduce the number of pairing operations from nt to $(t+1)$. Similarly, the correctness of the above equation can be proved with the properties of bilinear maps.

VI. Performance

In this section, we analyze the computation and communication overhead of our scheme in both the single-SEM model and multi-SEM model, and then we evaluate the performance of our scheme by experiments.

A. Computation and Communication Overhead

Since exponentiation operations and pairing operations require more time to compute than other operations, we focus on these two kinds of operations during the analysis of computation overhead. In the following, we use $\mathtt{Exp}_{\mathbb{G}_1}$ to denote the computation cost of one exponentiation in $\mathbb{G}_1$, and $\mathtt{Pair}$ to denote the computation cost of one pairing operation on bilinear map $\mathtt{e}: \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$.

1) Signature Generation: As described in Section IV, in order to generate a signature on a block of data, a data owner first needs to aggregate and blind the block with algorithm **Blind**, which costs $(k+1)\mathtt{Exp}_{\mathbb{G}_1}$. After received a blinded message from the data owner, the SEM computes a signature with algorithm **Sign**, which costs $1\mathtt{Exp}_{\mathbb{G}_1}$. Finally, with algorithm **Unblind**, the data owner verifies the correctness of a blind signature, and recovers the actual signature of the original block, which costs $1\mathtt{Exp}_{\mathbb{G}_1} + 2\mathtt{Pair}$.

So, to generate signatures for the entire data, the computation cost for unblinding n signatures is $n\mathtt{Exp}_{\mathbb{G}_1} + 2n\mathtt{Pair}$. With batch verification, we can reduce the computation cost for unblinding n signatures to $3n\mathtt{Exp}_{\mathbb{G}_1} + 2\mathtt{Pair}$ based on Equation (7). The total computation cost for the generation of all the n signatures are presented in Table I.

Compare to the single-SEM model, a data owner needs to spend a higher computation cost in algorithm **Unblind**$^{\diamond}$ in the multi-SEM model, due to the verification of the $n \cdot t$ blind signature shares and the recovery of n signatures on the n original blocks with Lagrange polynomial interpolation. If a data owner verifies each blind signature separately, this data owner needs to compute $2nt$ pairing operations, which significantly decreases the efficiency of unblinding. By leveraging batch verification of the nt blind signatures and pre-computation on Lagrange basis polynomials, the total computation cost in algorithm **Unblind**$^{\diamond}$ can be reduced to $(3nt + n)\mathtt{Exp}_{\mathbb{G}_1} + (t+1)\mathtt{Pair}$.

The communication cost between a data owner and the SEM during the generation of a signature is $2|p|$ bits, where $|p|$ denotes the size of an element of $\mathbb{Z}_p$. In the multi-SEM model, the communication cost will linearly increase with the number of SEMs. If the total number of SEMs is w,

Table I
COMPUTATION COST OF THE GENERATION FOR ALL THE n SIGNATURES

	Single-SEM Model	Multi-SEM Model
Basic Performance	$n(k+3)\texttt{Exp}_{\mathbb{G}_1} + 2n\texttt{Pair}$	$n(k+2t+1)\texttt{Exp}_{\mathbb{G}_1} + 2nt\texttt{Pair}$
Optimized Performance	$n(k+5)\texttt{Exp}_{\mathbb{G}_1} + 2\texttt{Pair}$	$n(k+4t+2)\texttt{Exp}_{\mathbb{G}_1} + (t+1)\texttt{Pair}$

then the total communication cost for generating a signature is $2w|p|$ bits per block.

2) Public Verification: A public verifier checks the correctness of a response at a cost of $(n+k)\texttt{Exp}_{\mathbb{G}_1} + 2\texttt{Pair}$. The communication cost of a challenge $\mathbf{C} = \{id_i, \beta_i\}_{i\in\mathcal{I}}$ is $n(|id| + |p|)$ bits, where $|id|$ denotes the size of an identifier of a block, the communication cost of a response $\mathbf{R} = \{\sigma^*, \alpha_1, ..., \alpha_k\}$ is $(k+1)|p|$ bits. Therefore, the total communication cost of public verification is $n|id| + (n+k+1)|p|$ bits. Clearly, as mentioned in previous section, while increasing k will save the communication cost for data owners during signature generation, it will increase the communication cost during public verification. The computation and communication overhead during public verification are still the same as in the multi-SEM model, because the efficiency of public verification is independent of the number of SEMs.

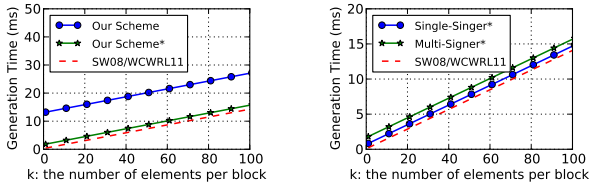
B. Experimental Results

We now evaluate the performance of our scheme, and compare it with several previous schemes using experiments. In the following experiments, we use Pairing Based Cryptography (PBC) library to simulate cryptographic operations, and we test our experiments on a Linux system with Intel Core i5 3.30 GHz Processor and 1 GB Memory. We assume the total size of cloud data is 2 GB, and $|p| = 160$ bits.

1) Signature Generation: As we discussed before, the main difference between our scheme and previous schemes [7], [4] is that, we introduce a SEM to compute signatures for data owners to support anonymity. Therefore, we first evaluate the performance of our scheme during signature generation, and compare it with previous schemes [7], [4] (denoted as SW08 and WCWRL11 in this paper), which are also able to support public verification based on BLS signatures [22] but not able to provide identity privacy.

From Figure 4(a), we can see that a data owner needs to spend much more time to generate a signature with the help of the SEM than simply generating it independently by herself, if she directly follows our scheme. For example, when $k = 100$, our scheme requires 27.07 milliseconds to generate a signature on a block while SW08/WCWRL11 only needs 14.27 milliseconds. However, if we utilize batch verification during the verification of the total number of n blind signatures on the entire data in algorithm **Unblind**, the average signature generation time can be reduced to 14.70 milliseconds per block, which is almost the same as the signature generation time in SW08/WCWRL11. It implies that introducing the SEM to provide identity privacy will not bring a huge burden to a data owner. In the following figures, we use Our Scheme* to denote the performance of this optimized version.

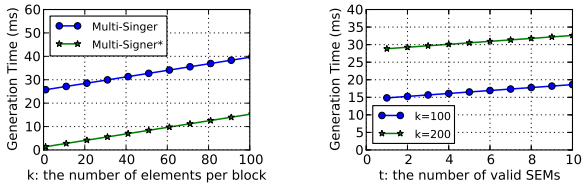
Similar to batch verification in the single-SEM model, we can also reduce the signature generation time in the multi-SEM model. As shown in Figure 4(b), the average signature generation time of our scheme in the multi-SEM model (if $t = 3$ and $k = 100$) is about 15.70 milliseconds per block, which is slightly higher than the average signature generation time of our scheme in the single-SEM model. It means that applying multiple signers to avoid the single point of failure will not dramatically affect the signature generation time.



(a) Impact of k on signature generation time (ms) (b) Impact of k on signature generation time (ms)

Figure 4. Computation Cost of Signature Generation

The performance of the signature generation time in multi-SEM model are presented in Figures 5(a) and 5(b). Clearly, if batch verification of blind signatures and precomputation of Lagrange basis polynomials are used in algorithm **Unblind**$^\diamond$, a data owner can save a lot of computation cost during the generation of a signature. Specifically, in Figure 5(a), when $k = 100$, the signature generation time is about 40 milliseconds per block if batch verification is not used in algorithm **Unblind**$^\diamond$. By taking advantage of batch verification, the average signature generation time can be reduced to 15.25 milliseconds per block.

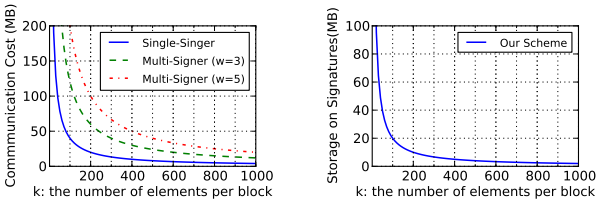


(a) Impact of k on signature generation time (ms), where $t = 2$ (b) Impact of t on signature generation time (ms)

Figure 5. Cost of Signature Generation in the Multi-SEM Model

During the signature generation, a data owner needs to send a block to the SEM to obtain a blind signature, which will cost a data owner extra communication overhead than generating signatures by herself independently. As we can see from Figure 6(a), by increasing parameter k, we can save the communication cost during the generation of signatures. When $k = 100$, the communication overhead is 40 MB, where the size of the entire data is 2 GB.

When $k = 1,000$, the communication overhead is only 4 MB, which is significantly smaller than the size of the entire data. In the multi-SEM model, the communication overhead increases linearly with the total number of SEMs. For example, if the total number of SEMs is $w = 5$ and the number of elements in each block is $k = 1,000$, then the communication overhead during the generation of signatures is 20 MB in total. The total storage for storing signatures in the cloud will also be dramatically decreased if the value of k increases, which is illustrated in Figure 6(b). Finally, the number of SEMs does not affect the signature storage cost.



(a) Impact of k on commun. cost of signature generation (MB) (b) Impact of k on signature storage (MB)

Figure 6. Communication and Storage Cost of Signatures

2) Public Verification: Since the adoption of blind signatures in our scheme does not affect the verification, the performance of public verification in our scheme is the same as SW08 [7] if the same parameters are selected. As illustrated in [3], the efficiency of public verification can be significantly improved by selecting a number of c random blocks instead of selecting all the n blocks, where $c \ll n$. As we can see from Table II, if we choose $k = 1,000$, then the total number of blocks in data is $100,000$ (because the size of data is 2 GB and $|p| = 160$ bits). When all the n blocks are selected during public verification, the computation overhead is 14.15 seconds and the communication is 2.27 MB. While if only $c = 460$ random blocks are selected, the computation overhead during public verification is only 0.21 seconds and the corresponding communication overhead is only 30.37 KB. According to previous work [3], [4], when $c = 460$, the probability to detect the incorrectness of cloud data is greater than 99%.

Table II
PERFORMANCE OF PUBLIC VERIFICATION WHEN $k = 1,000$

	$n = 100,000$	$c = 460$
Computation Overhead	14.15 seconds	0.21 seconds
Communication Overhead	2.27 MB	30.37 KB

3) Performance Comparison among the Schemes with Identity Privacy: We compare the performance of our scheme, including signature generation time, extra storage space on signatures, computation overhead, and communication overhead during public verification, with previous schemes [5], [13], which also aimed to preserve identity privacy in the multi-owner scenario. In Table III, we assume the entire data is 2 GB, the size of an element in a block is $|p| = 160$ bits, the number of elements in a block is $k = 1,000$, the total number of blocks is $n = 100,000$, and the number of multiple owners in a group is $d = 10$. During integrity verification, a number of $c = 460$ random blocks are selected. As shown in the following table, when protecting identity privacy for a group of multiple owners, our scheme has a better performance than previous work [5], [13]. Besides, our scheme can easily support group dynamic while the other two schemes cannot.

Table III
COMPARISON AMONG SCHEMES WITH IDENTITY PRIVACY

	Our scheme	[5]	[13]
Signature Generation Time (ms)	147.01	142.72	20.28
Extra Storage for Signatures (MB)	2	20	32.8
Computation Overhead (seconds)	0.21	1.20	3.97
Communication Overhead (KB)	30.37	50.55	126.05
Public Verification	Yes	Yes	No
Group Dynamic	Yes	No	No

VII. RELATED WORK

To allow a data owner to verify the integrity of data that stored in a remote and untrusted server without downloading the entire data, Ateniese *et al.* [3] first proposed provable data possession (PDP) based on homomorphic authenticators and sampling strategies. In addition, this scheme can also support public verification, Shacham and Waters [7] designed an improved public verification scheme based on BLS signatures [22]. In order to support dynamic data while still achieving public verification on the integrity of remote data, one may integrate the use of rank-based authenticated dictionary [8] or Merkle Hash Tree [9].

Wang *et al.* [10] designed a public verification scheme for cloud data, where data is encoded with erasure codes, so that the content of a user's data can be recovered even if some part of data is polluted. Moving a step forward, by using techniques from network coding, the scheme of Chan *et al.* [11] can minimize the data repair cost. Recently, Cao *et al.* [12] leveraged LT codes to make the data owner free from the burden of being online after her data outsourcing.

Apart from identity privacy, data privacy is also an important issue during public verification of the cloud data integrity. Wang *et al.* [4] considered how to preserve data privacy from a third party auditor (TPA) by using zero-knowledge proof techniques. In addition, their system also supports batch verification of multiple data auditing requests. More recently, Wang *et al.* [24], [25] designed two schemes to verify the integrity of shared data while supporting efficient user revocation with the techniques of proxy re-signatures [26], [27]. In addition, Tate *et al.* [28] proposed a scheme for verifying the integrity of multi-user data with the help of trusted hardware. However, these three schemes are not able to preserve the identities of data owners.

VIII. CONCLUSION

In this paper, we introduce what we believe is the right approach to achieve anonymity in storing data to the cloud with publicly-verifiable data-integrity in mind. Our approach decouples the anonymous protection mechanism from the

provable data possession mechanism via the use of security-mediator. Our solution not only minimizes the computation and bandwidth requirement of this mediator, but also minimizes the trust placed on it in terms of data privacy and identity privacy. The efficiency of our system is also empirically demonstrated.

ACKNOWLEDGEMENT

This work is supported by the NSF of China (No. 61272457 and 61170251), Fundamental Research Funds for the Central Universities (No. K50511010001), National 111 Program (No. B08038), Doctoral Foundation of Ministry of Education of China (No. 20100203110002), Program for Changjiang Scholars and Innovative Research Team in University (PCSIRT 1078) and U.S. NSF CNS-1218085.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A View of Cloud Computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, 2010.

[2] K. Ren, C. Wang, and Q. Wang, "Security Challenges for the Public Cloud," *IEEE Internet Computing*, vol. 16, no. 1, pp. 69–73, 2012.

[3] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," in *ACM CCS*, 2007, pp. 598–610.

[4] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-Preserving Public Auditing for Secure Cloud Storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[5] B. Wang, B. Li, and H. Li, "Oruta: Privacy-Preserving Public Auditing for Shared Data in the Cloud," in *IEEE Cloud*, June 2012, pp. 295–302.

[6] R. Dingledine, N. Mathewson, and P. Syverson, "Tor: The Second-Generation Onion Router," in *USENIX Security Symposium*, 2004.

[7] H. Shacham and B. Waters, "Compact Proofs of Retrievability," in *ASIACRYPT*, 2008, pp. 90–107.

[8] C. Erway, A. Kupcu, C. Papamanthou, and R. Tamassia, "Dynamic Provable Data Possession," in *ACM CCS*, 2009, pp. 213–222.

[9] Q. Wang, C. Wang, J. Li, K. Ren, and W. Lou, "Enabling Public Verifiability and Data Dynamic for Storage Security in Cloud Computing," in *ESORICS*. Springer-Verlag, 2009, pp. 355–370.

[10] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring Data Storage Security in Cloud Computing," in *ACM/IEEE IWQoS*, 2009, pp. 1–9.

[11] B. Chen, R. Curtmola, G. Ateniese, and R. Burns, "Remote Data Checking for Network Coding-based Distributed Stroage Systems," in *ACM CCSW*, 2010, pp. 31–42.

[12] N. Cao, S. Yu, Z. Yang, W. Lou, and Y. T. Hou, "LT Codes-based Secure and Reliable Cloud Storage Service," in *IEEE INFOCOM*, 2012, pp. 693–701.

[13] B. Wang, B. Li, and H. Li, "Knox: Privacy-Preserving Auditing for Shared Data with Large Groups in the Cloud," in *ACNS*, 2012, pp. 507–525.

[14] S. S. M. Chow, C.-K. Chu, X. Huang, J. Zhou, and R. H. Deng, "Dynamic Secure Cloud Storage with Provenance," in *Cryptography and Security - Dedicated to Jean-Jacques Quisquater*. Springer, 2012, pp. 442–464.

[15] S. S. M. Chow, Y.-J. He, L. C. K. Hui, and S. M. Yiu, "SPICE: Simple Privacy-Preserving Identity-Management for Cloud Environment," in *ACNS*, 2012, pp. 526–543.

[16] A. Shamir, "How to Share a Secret," vol. 22, no. 11, pp. 612–613, 1979.

[17] K. Y. Yu, T. H. Yuen, S. S. M. Chow, S.-M. Yiu, and L. C. K. Hui, "PE(AR)^2: Privacy-Enhanced Anonymous Authentication with Reputation and Revocation," in *ESORICS*, 2012, pp. 679–696.

[18] D. Chaum, "Blind Signatures for Untraceable Payments," in *CRYPTO*, 1982, pp. 199–203.

[19] C.-K. Chu, W. T. Zhu, S. S. M. Chow, J. Zhou, and R. H. Deng, "Secure Mobile Subscription of Sensor-Encrypted Data," in *ASIACCS*, 2011, pp. 228–237.

[20] M. Li, N. Cao, S. Yu, and W. Lou, "FindU: Private-Preserving Personal Profile Matching in Mobile Social Networks," in *IEEE INFOCOM*, 2011, pp. 2435 – 2443.

[21] A. L. Ferrara, M. Green, S. Hohenberger, and M. Ø. Pedersen, "Practical Short Signature Batch Verification," in *CT-RSA*. Springer-Verlag, 2009, pp. 309–324.

[22] D. Boneh, B. Lynn, and H. Shacham, "Short Signatures from the Weil Pairing," *J. Cryptology*, vol. 17, no. 4, pp. 297–319, 2004.

[23] A. Boldyreva, "Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme," in *PKC*, 2003, pp. 31–46.

[24] B. Wang, B. Li, and H. Li, "Public Auditing for Shared Data with Efficient User Revoation in the Cloud," in *IEEE INFOCOM*, 2013.

[25] B. Wang, H. Li, and M. Li, "Privacy-Preserving Public Auditing for Shared Cloud Data Supporting Group Dynamics," in *IEEE ICC*, 2013.

[26] G. Ateniese and S. Hohenberger, "Proxy Re-signatures: New Definitions, Algorithms and Applications," in *ACM CCS*, 2005, pp. 310–319.

[27] S. S. M. Chow and R. C.-W. Phan, "Proxy Re-signatures in the Standard Model," in *ISC*. Springer, 2008, pp. 260–276.

[28] S. R. Tate, R. Vishwanathan, and L. Everhart, "Multi-user Dynamic Proofs of Data Possession Using Trusted Harware," in *ACM CODASPY*, 2013, pp. 353–364.