A Survey of Public Auditing for Secure Data Storage in Cloud Computing

Wei-Fu Hsien[1], Chou-Chen Yang[1], and Min-Shiang Hwang[2,3]
(Corresponding author: Min-Shiang Hwang)

Department of Management Information System, National Chung Hsing University[1]
Department of Computer Science and Information Engineering, Asia University[2]
No. 500, Lioufeng Rd., Wufeng, Taichung 41354, Taiwan
(Email: mshwang@asia.edu.tw)
Department of Medical Research, China Medical University Hospital, China Medical University[3]
No. 91, Hsueh-Shih Road, Taichung 40402, Taiwan
(Received Mar. 15, 2015; revised and accepted May 11 & May 26, 2015)

Abstract

Cloud computing has been popular as the IT architecture. Cloud service providers offer many services based on cloud computing. Cloud storage service is the cloud services which can provide a huge storage space to solve the bottleneck of the storage space of local end users. However, cloud storage service may have data security because the users' data is not stored in their own storage. In this paper, we will focus on data integrity in the cloud storage service. Public auditability is a model of outsourcing data integrity verification, which can achieve efficiency and security. Therefore, we survey the previous researches of data integrity based on public auditability which includes collecting the basic requirements and evaluation metrics, providing the representative with approaches to analyze security and efficiency. Finally, we propose some future developments.

Keywords: CGA generation algorithm, hash functions, multithreading

1 Introduction

Cloud computing is a computing technology, and the Internet has grown in recent years. It can share the software and hardware resources, and provide resources to a user's computer or mobile device. The user can obtain a more efficient service because cloud computing can integrate resources. Therefore, in order to achieve cloud computing technology, it must satisfy five basic features: On-demand self-service, Broad network access, Resource pooling, Rapid elasticity and Measured service [10]. However, is very difficult for general users or small and medium enterprises to construct cloud environment because they cannot afford the huge costs. Therefore, many information technology companies are finding business opportunities to cloud services. Thus, cloud service providers have joined to build cloud environments and provide services to the user. Cloud service providers offer three services including Software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS). The cost for users to rent cloud service is cheaper than the cost for users to build cloud environment.

Cloud storage service is the most common and popular service among many cloud services (e.g. Google Drive, Dropbox, Amazon S3 and Microsoft OneDrive) for general users. Users have a bottleneck in local storage space because there are more and more users to save data in cloud storage, so cloud storage service has high capacity which solves users' difficult problem. Besides, cloud storage service provides high capacity space, and, in order to achieve ubiquitous service, it also provides to access cloud services from web service or applications that utilize the application programming interface (API) by mobile devices (e.g. laptop, table computer and smart phones).

Although cloud storage service has many advantages, it brings a lot of challenging issues which include efficacy and security [5, 9]. One of the big challenges is verifying the integrity of the data because users cannot know how the cloud storage service handles their data. These cloud storage services are provided by commercial enterprises, so it cannot be fully trusted by users. Therefore, the cloud service provider may hide data loss and data errors in the service because their benefits. It is very serious when a user stores data in untrusted cloud storage, for example, a large size of the outsourced data and the client's limited resource capability, and the client how to find an efficient way to achieve integrity verifications without the local copy of data files.

In order to solve the problem of data integrity verification in the cloud storage service, many studies present

different systems and security models [1, 2, 4, 6, 7, 8, 12, 13, 14, 15]. In these studies, the role of the verifier can fall into two categories: private auditability and public auditability. Private auditability implies the data owner directly verifying data in the cloud storage service is an efficient way. Public auditability implies the data owner allowing other to verify the data owner's data is inefficient. In general, the data owner may have a lot of data files which are stored in cloud storage service. However, the data owner cannot frequently verify their data because it will consume their resources which cannot process other action. In order to achieve an efficient verification of data integrity, the data owner can delegate a trusted third party auditor (TPA) to assist the validation data reduction to consume the data owner's computing resources.

The rest of paper is organized as follows: In Section 2, we review the related work of public auditability. We classify the basic requirements of function, security and efficiency in Section 3. In Section 4, we discuss the representative approaches of public auditability in detail. In Section 5, we analyze the basic requirement in the representative approaches. Finally, we summarize and discuss the future work in Section 6.

2 Related Work

In recent years, many of the literatures have pursued the context of remotely stored data verification [1, 2, 4, 6, 7, 8, 12, 13, 14, 15]. In 2007, Ateniese et al. [1] proposed the provable data possession (PDP) model which can provide public auditability and ensure possession of files on untrusted storage. They use RSA-based homomorphic verifiable tags to audit outsourced data. Their scheme first provides blockless verification and public verifiability at the same time. However, Ateniese et al.'s scheme cannot support dynamic data verification because their scheme only considers static data situation which means the client stores outsourced data and will not modify it. Therefore, Ateniese et al. [2] proposed a scalable PDP scheme to improve dynamic data verification in 2008. Nevertheless, their scheme cannot support fully dynamic data which cannot support block insertions because their scheme only allows simple block operation which implies partially dynamic data like block modification and block deletion. Wang et al. [14] proposed a challenge-response protocol which can determine the data correctness and locate possible errors. However, their scheme only supports partially dynamic data operation. Erway et al. [4] proposed a dynamic provable data possession which extends the PDP model to support fully dynamic data. They use another authenticated data structure which is a rank-based authenticated skip lists to prove and update the remote stored data. However, their scheme cannot support public verification because they only considers to achieve fully dynamic data.

Juels and Kaliski [6] proposed the proof of retrievability (POR) model, where spot-checking and error-correcting codes can make sure possession and retrievability of data files on remote archive service systems. However, their scheme only suits static data storage because the number of queries a client can perform is fixed a priori and embedding special blocks (call sentinels) which prevent the development of dynamic data updates. Shacham and Waters [12] proposed an improved POR scheme which uses BLS signature [3] to replace the RSA-based signature to reduce the proof size. They use public verifiable homomorphic linear authenticators that are built from BLS signature and secure random oracle model. They prove that it is secure in a polynomial extraction algorithm to reveal message. However, they only consider static data operation.

In order to satisfy public verification and dynamic data Wang et al. [15] proposed a new scheme in the Figure 1. Their scheme improves the index of data block which can support fully dynamic data. They extended their scheme to support batch auditing which can improve efficiency. Wang et al. [13] pointed out that Wang et al.'s scheme has data privacy issues which imply TPA can get the client's data information. Therefore, they use a random mask technology to avoid TPA learning knowledge on every verification process. Li et al. [7] consider that the client's resource-constrained device is simple and lightweight. Therefore, they propose a scheme which can delegate TPA to execute high computing process and solve the client's bottleneck. Liu et al. [8] think that previous studies are not efficient in dynamic data update because it is a fixed-size block update. Therefore, they propose a scheme which can support variable-size blocks in dynamic data update. In Section 4, we will describe these representative approaches in detail.

3 Basic Requirements and Evaluation Metrics

According to [1, 2, 4, 6, 7, 8, 12, 13, 14, 15] studies, where they provide the basic requirements of security and performance. In our paper, we classify and describe these requirements. Then we use these requirements to analyze the existing scheme in Section 4.

1) Security Evaluation:

Blockless Verification. The auditor can verify data blocks, and need not to retrieve all audited data blocks in the cloud storage service. Stateless Verification: the auditor needs not to maintain and update data situation because data situation is maintained by the client and cloud storage service together.

Batch Auditing. The auditor can verify the data of different clients at the same time because the auditor can be delegated by a lot of clients.

Dynamic Data. The data owner can insert, modify and delete data blocks in the cloud storage

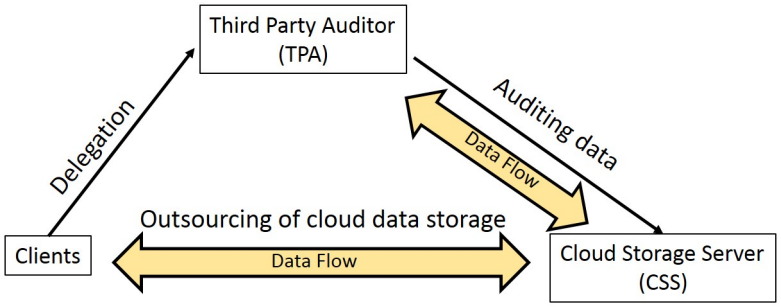

Figure 1: Public auditability in cloud data storage architecture

service because their data can be continuously updated at any time.

Privacy Presenting. The auditor cannot get knowledge which is the delegated data from the response of the cloud storage service.

2) Performance Evaluation:

Computing Cost. In order to achieve an efficient public auditing, we will analyze the client, TPA and cloud storage service cost on the computing resources.

Storage Cost. Because the client will upload data to the cloud storage service without the local copy of data files, we will analyze the client, TPA and cloud storage service cost on the storage spaces.

4 Representative Approaches

In the section we explain a preliminary concept and a system model before we introduce representative approaches.

4.1 Preliminary

Bilinear Pairing. Boneh et al. proposed a bilinear pairing mechanism to achieve a more efficient and secure verification. The mechanism will be explained as follows: Let G be additive group and G_T be a multiplicative group, all of prime order p. There exists a bilinear map $e : G \times G \rightarrow G_T$ and satisfies the following three properties [3]:

1) Bilinear: for all g_1, $g_2 \in G$ and $a, b \in Z_p$, $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab}$.
2) Non-degenerate: if g is the generator of G, and $e(g, g)$ is the generator of G_T. It needs to satisfy $e(g, g) \neq 1$.
3) Computability: an efficient algorithm exists to compute $e(g_1, g_2)$ for any g_1, $g_2 \in G$.

Merkle Hash Tree. The Merkle Hash Tree (MHT) is an authenticated data structure intended to efficiently and securely prove that a set of elements are undamaged and unaltered [11]. It is constructed as a binary tree where the leaves in the MHT are the hashes of authentic data values.

The MHT is demonstrated in Figure 2, and the verifier wants to check whether a set of element are undamaged in outsourcing storage. First the verifier randomly chooses number of elements (where we assume a set of element have eight nodes and only chooses an element x_2) to send to the prover. The prover responses the verifier with the auxiliary authentication information (AAI) $\Omega_2 =< h(x_2), h_d, h_b >$. The verifier computes $h(x_2)$, $h_c = h(h(x_1)||h(x_2))$, $h_a = h(h_c||h_d)$ and $h_r = h(h_a||h_b)$ and then checks if the value h_r is the same as the authentic one. In public auditing, the MHT is used to authenticate both the values and the positions of data blocks. The root is constructed by the leaf nodes as the left-to-right sequence. Therefore, the leaf nodes positions can be uniquely determined by the way of computing the root in MHT.

4.2 System Model

Client. an individual consumer or organization has a lot of data files and needs to store in the cloud. It depends on the cloud to manage data and computation, so it can reduce storage cost.

Cloud Storage Service (CSS). A cloud service provider has huge storage space and computation resource to provide the clients' data.

Third Party Auditor (TPA). A trusted organization has expertise and capabilities that the clients do not have. It is responsible for assessing the clients' data on cloud storage service.

4.3 Public Auditing of Dynamic Data

Wang et al. [15] was the first to propose the scheme which can support public verification and fully dynamic data at the same time because previous studies only supported to modify and delete on a data file. They define public auditability which implies public verification is delegated by a trusted third party auditor (TPA) to verify.

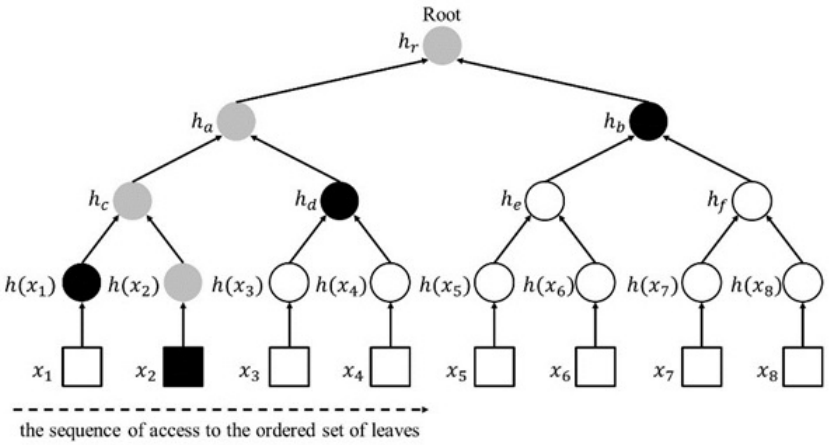

Figure 2: Merkle hash tree authentication of data elements. The leaf nodes $h(x_1), h(x_2), \cdots, h(x_n)$ is arranged in left-to-right sequence.

They propose a scheme to improve complex the file index information because this needs to consume a lot of computed resource. For example, when a file is inserted into the data block, this is required to recalculate the signature of the new index in the all files under the data block.

In order to solve the problem, they use $H(m_i)$ as the tag for block m_i instead of $H(name||i)$ [12] or $H(v||i)$ [6], and then single data operation on any file block will not affect the others. In the existing PDP or POR models $H(name||i)$ [12] or $H(v|i)$ [6] should be generated by the client in the verification process. However, in their scheme the client has no capability to compute $H(m_i)$ without the data file. In order to achieve blockless verification, the cloud storage service will process the computing $H(m_i)$ and then response it to the TPA. Because clients delegate audit services more and more, how to perform efficient audit service is a big problem. Therefore, in order to enhance the efficiency of audit services, they proposed a batch auditing protocol which can audit different client data files simultaneously. Their scheme is as follows:

1) Setup:

Setup 1. The client generates a signing key pair which is composed of signing public key (spk) and signing secret key (ssk). Then chooses a random number $\alpha \leftarrow Z_p$ and computes $v \leftarrow g^{\alpha}$. Thus, client's key pair is that the secret key is $sk = (\alpha, ssk)$ and the public key is $pk = (v, spk)$.

Setup 2. The client selects a file F which is split n blocks as $F = (m_1, m_2, \cdots, m_n)$, chooses a random element $u \leftarrow G$ and computes the file tag $t = name||n||u||SSig_ssk(name||n||u)$. The client computes signature $\sigma_i = (H(m_i) \cdot u^{m_i})^{\alpha}$ for each block and collects a signature set of $\phi = \{\sigma_i\}_{1 \le i \le n}$.

Setup 3. The client generates a root R from each hash value $H(m_i)$ of block i as a leaf node by the construction of the MHT. Then signs the root R as $Sig_{sk}(H(R)) \leftarrow (H(R))^{\alpha}$.

Setup 4. The client sends $\{F, t, \phi, sig_{sk}(H(R))\}$ to CSS. If CSS has received, the client will delete $\{F, \phi, sig_{sk}(H(R))\}$ from local storage.

2) Default Integrity Verification:

Setup 1. TPA selects c elements as a subset $I = \{s_1, s_2, \cdots, s_c\}$ from the auditing file, and chooses a random element $v_i \leftarrow Z_p$ for each block in I. Then TPA sends the challenged message $\{(i, v_i)\}_{(i \in I)}$ to CSS.

Setup 2. CSS receives the challenge of the client before computes $u = \sum_{i=s_1}^{s_c} v_i m_i \in Z_p$ and $\sigma = \Pi_{i=s_1}^{s_c} \sigma_i^{v_i} \in G$ from the stored block m_i with corresponding v_i. Then, CSS sends the proof $\{\{u, \sigma, H(m_i), \omega_i\}_{s_1 \le i \le s_c}, sig_{sk}(H(R))\}$ to TPA.

Setup 3. TPA generates the root R using $\{H(m_i), \omega_i\}_{s_1 \le i \le s_c}$ by checking

$$e(sig_{sk}(H(R)), g) \stackrel{?}{=} e(H(R), g^{\alpha}).$$

If the result is true, TPA verifies

$$e(\sigma, g) \stackrel{?}{=} e(\Pi_{i=s_1}^{s_c} H(m_i)^{v_i} \cdot u^U, v)$$

using the challenge message. Finally, if all results are true, TPA can make sure the client's data integrity in CSS.

3) Dynamic Data Operation with Integrity Assurance:

Setup 1. The client wants to modify the ith block m_i to m_i' and generates a new signature of block $\sigma_i' = (H(m_i') \cdot u^{m_i'})^{\alpha}$. Then the client sends the update request message (M, i, m_i', σ_i') to CSS.

Setup 2. CSS receives modification request from the client, and replaces (m_i, σ_i) with (m_i', σ_i'). CSS computes a new root R', and sends $\{\omega_i, H(m_i), sig_{sk}(H(R)), R'\}$ to the client.

Setup 3. The client generates root R using $\{\omega_i, H(m_i)\}$ and verifies

$$e(sig_{sk}(H(R)), g) \stackrel{?}{=} e(H(R), g^{\alpha}).$$

If the result is true, the client generates new root R_{new} using $\{\omega_i, H(m_i')\}$ and compares it with R'. If the result is true, the client signs R' as $sig_{sk}(H(R'))$ and sends it to CSS.

Setup 4. CSS receives the new root signature and updates it on the client file.

4) Batch Auditing:
Assume there are K clients delegate TPA to audit their data in CSS, and each client k has data files $F_i = (m_{(k,1)}, m_{(k,2)}, \cdots, m_{(k,n)})$, where $k \in \{1, 2, \cdots, K\}$. Their batch auditing protocol is as follows. In the setup and signature phase, one of the clients k chooses a random number $x_k \leftarrow Z_p$, and computes $v_k = g^x$, then the client's secret key is $sk = (x_k)$ and the public key $pk = (v_k)$. Client k chooses a random element $u_k \leftarrow G$ and computes signature $\sigma_{k,i} = (H(m_{(k,i)}) \cdot u_k^{m_{(k,i)}})^{x_k} \in G$. In the proof phase, CSS receives the challenge message $\{(i, v_i)\}_{s_1 \le i \le s_c}$ and computes $u_k = \sum_{(i,v_i)_{s_1 \le i \le s_c}} v_i m_{k,i} \in Z_p$ and $\sigma = \Pi_{i=1}^{k}(\Pi_{\{(i,v_i)\}_{s_1 \le i \le s_c}} \sigma_{k,i}^{v_i})$ for each client k ($k \in \{1, 2, \cdots, K\}$). CSS sends the proof $\{\sigma, \{u_k\}_{1 \le k \le K}, \{\omega_{k,i}\}, \{H(m_{k,i})\}\}$ to TPA. In the verification phase, first TPA generates the roots using $\{\{\omega_{k,i}\}, \{H(m_{k,i}\}\}$ and verifies the roots for each client's file. If the result is true, TPA verifies $e(\sigma, g) \stackrel{?}{=} \Pi_{k=1}^{K} e(\Pi_{(i,v_i)_{s_1 \le i \le s_c}} (H(m_{k,i}))^{v_i} \cdot (u_k)^{u_k}, v_k)$ using the challenge message to combine the bilinear map. Finally, if all results are true, TPA can make sure the clients' data integrity in CSS.

4.4 Public Auditing of Privacy-Preserving Data

Wang et al. [13] proposed a privacy protection scheme which is considered user's data privacy in the public auditability. Data privacy implies personally identifiable information or sensitive information whether they can be shared with third parties. As far as users are concerned what they depend on TPA just for the outsourced storage security of their data. However, most studies do not consider the protection of clients' private information in the auditing phase. This is a serious problem because an auditor may leak information without the client's authorization. Besides, there are legal regulations, such as the Health Insurance Portability and Accountability Act (HIPPA), it guarantees patient confidentiality for all healthcare-related data and demands the outsourced data not to be leaked to external parties.

Because public auditing model allows third-party auditors to assist clients to verify their data integrity, TPA obtains partly data blocks and learns by each sample to collect information in the auditing phase. For instance, Wang et al. [15] proposed a scheme where TPA sends the challenged message $\{(i, v_i)\}_{i \in I}$ to CSS and CSS responses the proof $\{\{u, \sigma, H(m_i), \omega_i\}_{s_1 \le i \le s_c}, sig_{sk}(H(R))\}$ to TPA. Therefore, TPA uses Homomorphic Linear Authenticator (HLA) characteristic to combine the blocks $u = \sum_{i=s_1}^{s_c} v_i m_i$, and it can potentially reveal user's data. TPA can gather the same set of c block $(m_1, m_2, \cdots, m_c)$ with corresponding random coefficients $\{v_i\}$. TPA can get the user's data $(m_1, m_2, \cdots, m_c)$ by computing different linear combinations $(u_1, u_2, \cdots, u_c)$. Thus it can be seen that it infringes the privacy-preserving guarantee.

Wang et al. proposed to integrate the homomorphic linear authenticator with random masking technique, and it achieves privacy-preserving public auditing. Because the random masking technique affects TPA learning knowledge, it can avoid TPA getting user's data. We are not going to elaborate on their scheme because it is similar to Wang et al.'s scheme on dynamic data and batch auditing operation.

Their scheme is as follows:

1) Setup:

Setup 1. The client chooses a random signing key pair (spk, ssk), a random number $x \leftarrow Z_p$, a random element $u \leftarrow G$, and computes $v \leftarrow g^x$. The secret key is $sk = (x, ssk)$ and the public key is $pk = (spk, v, g, u, e(u, v))$.

Setup 2. The client computes signature $\sigma_i = (H(W_i) \cdot u^{m_i})^x$ for each block where $W_i = name||i$ is combined with the user's identification $name$ and the block index i. Then, the client collects a signature set of $\phi = \{\sigma_i\}_{1 \le i \le n}$ and computes the file tag $t = name||SSig_{ssk}(name)$.

Setup 3. The client sends (F, ϕ, t) to CSS. If CSS has received, the client will delete (F, ϕ) from local storage.

2) Integrity Verification:

Setup 1. TPA selects c elements as a subset $I = \{s_1, s_2, \cdots, s_c\}$ from the auditing file, and chooses a random element $v_i \leftarrow Z_p$ for each block in I. Then TPA sends the challenged message $\{(i, v_i)\}_{i \in I}$ to CSS.

Setup 2. CSS receives challenge $\{(i, v_i)\}_{i \in I}$ and generates a response proof of data storage correctness. First, CSS computes $u_i' = \sum_{i=s_1}^{s_c} v_i m_i$ and $\sigma = \Pi_{i=s_1}^{s_c} \sigma_i^{v_i}$ which are corresponding to m_i and σ_i in the CSS's storage. Second, CSS randomly chooses an element $r \leftarrow Z_p$ and computes $R = e(u, v) \in G_T$ and $\gamma = H(R) \in Z_p$.

Finally, CSS computes $u = r + \gamma$ which is random masking technique, and sends $\{u, \sigma, R\}$ to TPA.

Setup 3. TPA receives the response proof of storage correctness, and computes $\gamma = H(R)$ to verify the equation $R \cdot e(\sigma^{\gamma}, g) \stackrel{?}{=} e((\Pi_{i=s_1}^{s_c} H(W_i^{v_i})^{\gamma} \cdot u^u, v)$. If the result is true, TPA can make sure the the client's data integrity, and cannot learn any knowledge about the data content stored in CSS.

4.5 Public Auditing of Resource-constrained Devices

Li et al. [7] propose a public auditability scheme in resource-constrained devices. Li et al.'s cloud data storage architecture is shown in Figure 3. Resource-constrained device is a simple and lightweight composition. Thus, these devices have low computation and storage capacity. However, these devices can achieve high mobility which allows users to carry and easily to use. Because the client may require repeatedly modified data in cloud storage service, this operation needs to compute in every update. Therefore, in the public audit model, the client needs a high burden of computing resources to operate dynamic data (such as signature $\sigma_i = (H(m_i) \cdot u^{m_i})^{\alpha}$ [15]) which is required to perform exponentiation and multiplication operation. In order to reduce the client's computation Li et al. propose a scheme which delegates trusted TPA to generate key, signature and delete file tag function. The clients can effectively reduce the computing resources because they only upload data to TPA. Therefore, the client will not have to compute signature on data update every time. Their scheme is as follows:

1) Integrity Verification:

Setup 1. TPA selects c elements as a subset $I = \{s_1, s_2, \cdots, s_c\}$ from the auditing file, and chooses a random element $v_i \leftarrow Z_p$ for each block in I. Then TPA sends the challenged message $\{(i, v_i)\}_{i \in I}$ to CSS.

Setup 2. CSS receives the challenge of the client before computes $u_j = \sum_{1 \le i \le c} v_i M_{ij} \in Z_p$ for $j = 1, 2, \cdots, s$ and $\sigma = \Pi_{1 \le i \le c} \sigma_i^{v_i} \in G$. CSS also provides some relevant information to TPA to verify client's data, and it includes $\{H(M_i, \Omega_i)\}_{1 \le i \le c}$ and $sig_{sk}(H(R))$. Finally, CSS responses $P = \{\{u_j\}_{1 \le i \le s}, \sigma, \{H(M_i, \Omega_i)\}_{1 \le i \le c}, sig_{sk}(H(R))\}$ to TPA.

Setup 3. TPA receives the response proof of storage correctness. First TPA generates the root R' using $\{H(M_i, \Omega_i)\}_{1 \le i \le c}$ and checks $sig_{sk}(H(R')) \stackrel{?}{=} sig_{sk}(H(R))$. Second TPA checks $e(t, g) \stackrel{?}{=} e(H(R), v)$. Finally, TPA checks whether $e(\sigma, g) \stackrel{?}{=} e(\Pi_{1 \le i \le c} H(M_i)^{v_i} \cdot \Pi_{j=1}^{s} u_j^{u_j}, v)$. If all results are true, TPA can make sure the clients' data integrity in CSS.

4.6 Authorized Public auditing of Fine-Grained Update

These schemes can support public auditing and dynamic data update. However, these schemes [15, 13, 7] support to insert, delete and modify operation in a fixed-size block which is later termed as coarse-grained updates. For instance, when a data block is partially modified, the block will be completely modified in coarse-grained updates. Therefore, this will cost additional resource. Liu et al. [8] propose a variable-size block scheme which is later termed as fine-grained updates in the public auditing. Their scheme can reduce an additional operation in partially modified block update. They also consider an authentication process to improve between the client and TPA because Wang et al.'s scheme [13] proposes challenge issues where TPA may learn the client's data by the verification process. Their scheme is as follows:

1) Integrity Verification:

Setup 1. TPA selects c elements as a subset $I = \{s_1, s_2, \cdots, s_c\}$ from the auditing file, and chooses a random element $v_i \leftarrow Z_p$ for each block in I. In order to achieve authentication, they add sig_{AUTH} and $\{VID\}_{PK_{CSS}}$ where $sig_{AUTH} = Sig_{ssk}(AUTH||t||VID)$ is include the client and TPA information and $\{VID\}_{PK_{CSS}}$ is means use CSS's public key to encrypt TPA's identification. Then TPA sends the challenged message $\{sig_{AUTH}, \{VID\}_{PK_{CSS}}, (i, v_i)\}_{i \in I}$ to CSS.

Setup 2. CSS receives the challenge of the client before verifies sig_{AUTH} with $AUTH, t, VID$ and the client's public key. If these verification are false, CSS reject it. Otherwise, CSS will compute $u_k = \sum_{i \in I} v_i m_{ik}$, $(k \in [1, w]$ and $w = max\{s_i\}_{i \in I})$ and compute $\sigma = \Pi_{i \in I} \sigma_i^{v_i}$. CSS provides the client's signature information sig from cloud storage. Finally, CSS responses $P = \{\{u_k\}_{k \in [1,w]}, \{H(m_i, \Omega_i)\}_{i \in I}, sig\}$ to TPA.

Setup 3. TPA receives the response proof of storage correctness. First TPA generates the root R' using $\{H(m_i, \Omega_i)\}_{i \in I}$ and checks $e(sig, g) \stackrel{?}{=} e(H(R'), v)$. Second TPA checks $e(\sigma, g) \stackrel{?}{=} e(w, v)$. Finally, TPA checks whether $e(\sigma, g) \stackrel{?}{=} e(\Pi_{i \in I} H(m_i)^{v_i} \cdot \Pi_{k \in [1,w]} u_k^{u_k}, g^{\alpha})$. If all results are true, TPA can make sure the clients' data integrity in CSS.

2) Dynamic Data Operation with Integrity Assurance:

Setup 1. The client wants to partial modify the ith block m_i to m_{new}. Therefore, the client computes update length in the ith block m_i. Then

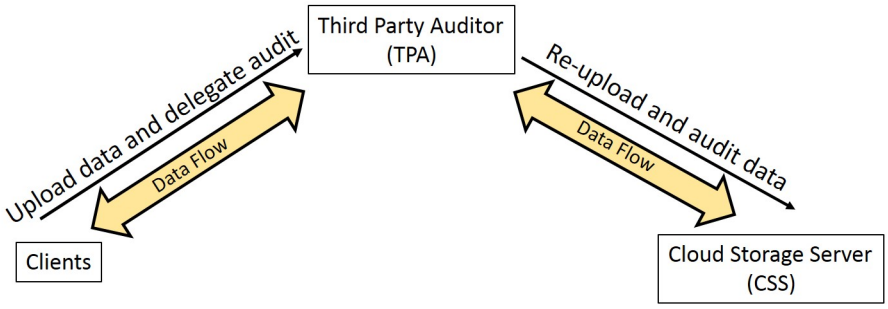

Figure 3: Li et al.'s cloud data storage architecture

the client sends the update request message $\{PM, i, o, m_{new}\}$ to CSS.

Setup 2. CSS receives the request from the client. First, CSS can ensure that the request is the partial modification (PM). Second, CSS uses $\{o, m_{new}\}$ to gather the sectors not involved in this update, which denote as $\{m_{ij}\}_{j\in M}$. Third, CSS will perform the update to get m_i' and use $\{m_i', \Omega_i\}$ to compute R'. Finally, CSS responses the proof $P = \{\{m_ij\}_{j\in M}, H(m_i), \Omega_i, R', sig\}$ to the client.

Setup 3. The client generates root R using $\{\Omega_i, H(m_i)\}$ and verifies the signature $sig \stackrel{?}{=} (H(R))^{\alpha}$. If it success, then computes m_i' using $\{m_{ij}\}_{j\in M}, m_{new}\}$. Thus, CSS can compute R_new using $\{m_i', \Omega_i\}$ and verifies $R_{new} \stackrel{?}{=} R'$. If all results are true, the client computes the new signature block $\sigma_i' = (H(m_i')\Pi_{j=1}^{s_i} u_j^{m_{ij}'})^{\alpha}$ and the new signature root $sig' = (H(R'))^{\alpha}$, and then returns update message $\{\sigma_i', sig'\}$ to CSS.

Setup 4. CSS receives the message, and then updates it on the client file.

5 Analysis

In the section, we will analyze these schemes [7, 8, 13, 15] which contain functional requirement, security and performance. And we also use the tables to present a corresponding requirement in each scheme.

5.1 Functional Requirement

In order to raise efficiency in verification, every scheme can support blockless verification. The comparison of functional requirement with related schemes is shown as Table 1. Because Li et al.'s scheme [7] needs TPA to assist the client's data file, their scheme does not satisfy stateless verification. Although Li et al. [7] and Liu et al. [8] did not explain whether their scheme support batch audit, we analyze whether their scheme can be extended to achieve it. In the dynamic data, because these scheme [7, 13, 15] do not consider partially modified data update, Liu et al. [8] only considered to update variable-size blocks. Wang et al. [13] only considered privacy presenting using random mask technology because other schemes assume that TPA can be fully trusted.

5.2 Performance Evaluation

We will analyze three phases: setup phase, auditing phase and dynamic data update phase. Before we analyze the performance evaluation, first we introduce the notations in Table 2. In Tables 3, 4, 5, we analyze the computation cost in setup, auditing, and dynamic data phases, respectively.

In the setup phase, Wang et al.'s scheme [15] is better than these schemes [7, 8, 13] because their scheme does not compute the number of sectors of a block. However, Li et al's scheme [7] is best on the client's point of view because the client delegates the whole operation process to TPA.

In the auditing phase, Wang et al.'s scheme [13] is better because the auditor reduces computation which cannot construct the root in the auditing phase. However, Liu et al.'s scheme [8] requires costly computing, but their scheme is the only way to achieve between TPA and CSS authentications.

In the dynamic data update phase, Liu et al.'s scheme [8] is better because their scheme can support partially modified data update which can reduce computing.

In the Table 6, we analyze storage cost in public auditing. Liu et al.'s scheme [12] requires a large storage space because their scheme can support partially modified data update and authentication. Li et al.'s scheme [7] needs to store some information on the TPA because their scheme make the client delegate TPA to perform signature.

6 Conclusions and Future Work

Because users' data is stored in the cloud storage service, it brings users' data security issues. In the public auditability model, users can delegate the third party auditor to verify their data is efficient. According to the literature, we sort out the basic requirements in public

Table 1: Comparison of functional requirements

	Wang et al. [15]	Wang et al. [13]	Li et al. [7]	Liu et al. [12]
Blockless verification	Yes	Yes	Yes	Yes
Stateless verification	Yes	Yes	No	Yes
Batch auditing	Yes	Yes	Yes	Yes
Dynamic data	Partial	Partial	Partial	Yes
Privacy presenting	No	Yes	No	No

Table 2: Notations

Notation	Description
T_E/T_D	The computing time of asymmetric encryptions;
T_{Ge}	The computing time of exponentiation in group operation;
T_{BLS}	The computing time of BLS signature;
T_B	The computing time of bilinear pairing;
T_M	The computing time of multiplication;
T_A	The computing time of addition;
T_{GM}	The computing time of multiplication in group operation;
T_h	The computing time of hash function;
n	The number of block in a file;
i	The number of verified block;
l	The number of inside node that needed in MHT;
o	The number of auxiliary authentication information (AAI);
s	The number of sectors of a block;
s_{max}	The maximum number of sectors a block.

Table 3: Comparison of computation in Setup phase

	Client	TPA
Wang et al. [15]	$(n+3)T_{BLS}+n(T_{GM}+T_{Ge})+(n+l)T_h$	No
Wang et al. [13]	$(n+3)T_{BLS}+n(T_{GM}+T_{Ge})+(2n+l)T_h$	No
Li et al. [7]	No	$(n+3)T_{BLS}+n(T_{GM}+s_{max}T_{Ge}))+(n+l)T_h$
Liu et al. [8]	$(n+3)T_{BLS}+n(T_{GM}+s_{max}T_{Ge}))+(n+l)T_h$	No

Table 4: Comparison of computation in Auditing phase

	TPA	CSS
Wang et al. [15]	oT_h+2T_B	$i(T_M+T_A+T_{Ge}+T_{GM})+(i+o)T_h$
Wang et al. [13]	T_h+T_B	$(i+1)(T_{GM}+T_A+T_{Ge})+iT_M+T_h$
Li et al. [7]	oT_h+2T_B	$i(T_M+T_A+T_{Ge}+T_{GM})+(o+i)T_h$
Liu et al. [8]	$T_E+oT_h+2T_B$	$T_D+i(T_M+T_A+T_{Ge}+T_{GM})+(o+i)T_h$

Table 5: Comparison of computation in Dynamic Data phase

	Client	TPA	CSS
Wang et al. [15]	$(2o+1)T_h + 2T_{BLS} + T_B$ $+(s_{max}+1)T_{GM} + s_m axT_{Ge}$	No	$(o+l+2)T_h$
Wang et al. [13]	$(2o+1)T_h + 2T_{BLS} + T_B$ $+(s_{max}+1)T_{GM} + s_{max}T_{Ge}$	No No	$(o+l+2)T_h$ $(o+l+2)T_h$
Li et al. [7]	No No	$(2o+1)T_h + 2T_{BLS} + T_B$ $++(s_{max}+1)T_{GM}s_{max}T_{Ge}$	$(o+l+2)T_h$ $(o+l+2)T_h$
Liu et al. [8]	$(2o+1)T_h + 2T_{BLS} + T_B + T_{GM}$ $+(s+1)T_{GM} + sT_{Ge}$	No No	$(o+l+2)T_h$ $(o+l+2)T_h$

Table 6: Comparison of storage

Storage cost	Wang et al. [15]	Wang et al. [13]	Li et al. [7]	Liu et al.[8]
Auditor	No	No	$sig_{sk}(R)$	No
Cloud Storage Service	$F, t, \phi, sig_{sk}(H(R))$	F, t, ϕ	F, ϕ	$F, T, t, \phi, R, sig_{sk}(H(R))$

auditability, which can be classified to the case for your application.

For future development, with big data generation, data verification will be more and more difficult. Because big data have three characteristics including volume, velocity and variety, these characteristics will affect the implementation of data verification. Therefore, it will be a major challenge how to efficiently verify data integrity in big data. However, this scheme must also satisfy basic requirements.

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*, pp. 598–609, Virginia, USA, 2007.

[2] G. Ateniese, R. D. Pietro, L. V. Mancini, and G. Tsudik, "Scalable and efficient provable data possession," in *Proceedings of the 4th International Conference on Security and Privacy in Communication Netowrks*, pp. 9:1–9:10, Istanbul, Turkey, 2008.

[3] D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Proceedings of the 7th International Conference on the Theory and Application of Cryptology and Information Security: Advances in Cryptology (ASIACRYPT'01)*, pp. 514–532, Gold Coast, Australia, 2001.

[4] C. Erway, A. K, C. Papamanthou, and R. Tamassia, "Dynamic provable data possession," in *Proceedings of the 16th ACM Conference on Computer and Communications Security*, pp. 213–222, Illinois, USA, 2009.

[5] I. A. T. Hashem, I. Yaqoob, N. B. Anuar, S. Mokhtar, A. Gani, and S. U. Khan, "The rise of big data on cloud computing: Review and open research issues," *Information Systems*, vol. 47, no. 6, pp. 98–115, 2015.

[6] A. Juels and J. Burton S. Kaliski, "Pors: Proofs of retrievability for large files," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, pp. 584–597, Virginia, USA, 2007.

[7] J. Li, X. Tan, X. Chen, D. Wong, and F. Xhafa, "OPoR: Enabling proof of retrievability in cloud computing with resource-constrained devices," accepted and to be publish in *IEEE Transactions on Cloud Computing*, Oct. 2014.

[8] C. Liu, J. Chen, L. T. Yang, X. Zhang, C. Yang, R. Ranjan, and K. Ramamohanarao, "Authorized public auditing of dynamic big data storage on cloud with efficient verifiable fine-grained updates," *IEEE Transactions on Parallel and Distributed Systems*, vol. 25, no. 9, pp. 2234–2244, 2014.

[9] C. Liu, C. Yang, X. Zhang, and J. Chen, "External integrity verification for outsourced big data in cloud and iot: A big picture," *Future Generation Computer Systems*, vol. 49, no. 6, pp. 58–67, 2015.

[10] P. M. Mell and T. Grance, "The nist definition of cloud computing," Technical Report: SP 800-145, 2011.

[11] R. C. Merkle, "Protocols for public key cryptosystems," in *IEEE Symposium on Security and Privacy*, pp. 122–134, California, USA, 1980.

[12] H. Shacham and B. Waters, "Compact proofs of retrievability," in *Proceedings of the 14th International Conference on the Theory and Application of Cryptology and Information Security (ASIACRYPT'08)*, pp 90–107, Melbourne, Australia, 2008.

[13] C. Wang, S. S. M. Chow, Q. Wang, K. Ren, and W. Lou, "Privacy-preserving public auditing for se-

cure cloud storage," *IEEE Transactions on Computers*, vol. 62, no. 2, pp. 362–375, 2013.

[14] C. Wang, Q. Wang, K. Ren, and W. Lou, "Ensuring data storage security in cloud computing," in *Proceedings of the 17th International Workshop on Quality of Service (IWQoS'09)*, pp. 1–9, South Carolina, USA, 2009.

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

Wei-Fu Hsien received his B. S. in Department of Information Management from National Kaohsiung Marine University, Kaohsiung, Taiwan, ROC, in 2013. He is currently pursuing the M.S. degree with the Department of Management Information Systems from National Chung Hsing University. His research interests include security and privacy of cloud computing, and applied cryptography.

Chou-Chen Yang received his B.S. in Industrial Education from the National Kaohsiung Normal University, in 1980, and his M.S. in Electronic Technology from the Pittsburg State University, in 1986, and his Ph.D. in Computer Science from the University of North Texas, in 1994. From 1994 to 2004, Dr. Yang was an associate professor in the Department of Computer Science and Information Engineering, Chaoyang University of Technology. Currently, he is a professor in the Department of Management Information Systems, National Chung Hshing University. His research interests include network security, mobile computing, and distributed system.

Min-Shiang Hwang received the B.S. in Electronic Engineering from National Taipei Institute of Technology, Taipei, Taiwan, Republic of China, in 1980; the M.S. in Industrial Engineering from National Tsing Hua University, Taiwan, in 1988; and the Ph.D. in Computer and Information Science from National Chiao Tung University, Taiwan, in 1995. He also studied Applied Mathematics at National Cheng Kung University, Taiwan, from 1984-1986. Dr. Hwang passed the National Higher Examination in field "Electronic Engineer" in 1988. He also passed the National Telecommunication Special Examination in field "Information Engineering", qualified as advanced technician the first class in 1990. From 1988 to 1991, he was the leader of the Computer Center at Telecommunication Laboratories (TL), Ministry of Transportation and Communications, ROC. He was also a project leader for research in computer security at TL in July 1990. He obtained the 1997, 1998, and 1999 Distinguished Research Awards of the National Science Council of the Republic of China. He is a member of IEEE, ACM, and Chinese Information Security Association. His current research interests include database and data security, cryptography, image compression, and mobile communications.