PRIVACY-PRESERVING PUBLIC AUDITING WITH DATA DEDUPLICATION IN CLOUD COMPUTING

by

Naelah Abdulrahman Alkhojandi

Bachelor of Computer Science, Umm Al-Qura University, 2005

A thesis

presented to Ryerson University

in partial fulfillment of the

requirements for the degree of

Master of Science

in the Program of

Computer Science

Toronto, Ontario, Canada, 2015

PRIVACY-PRESERVING PUBLIC AUDITING WITH DATA DEDUPLICATION IN CLOUD COMPUTING

Master of Science 2015

Naelah Abdulrahman Alkhojandi

Computer Science

Ryerson University

Abstract

Storage represents one of the most commonly used cloud services. Data integrity and storage efficiency are two key requirements when storing users' data. Public auditability, where users can employ a Third Part Auditor (TPA) to ensure data integrity, and efficient data deduplication which can be used to eliminate duplicate data and their corresponding authentication tags before sending the data to the cloud, offer possible solutions to address these requirements. In this thesis, we propose a privacy-preserving public auditing scheme with data deduplication. We also present an extension of our proposed scheme that enables the TPA to perform multiple auditing tasks at the same time. Our analytical and experimental results show the efficiency of the batch auditing by reducing the number of pairing operations need for the auditing. Then, we extend our work to support user revocation where one of the users wants to leave the enterprise.

Acknowledgements

All praises and glory is due to almighty God for blessing and strengthening me in every single moment through this amazing opportunity and great journey of knowledge.

My great thanks and deepest gratitude to my academic supervisor Prof. Ali Miri for his guidance and persistent support. I appreciate his generosity to make time for my questions at any matter. I also appreciate his wise contribution and encouragement in publishing a paper on the Symposium of Foundation and Practice of Security which held between 3rd and 5th of November 2014 in Montreal, Canada. This would not be possible without his dedication in perfecting my research skills and his invaluable insights.

Furthermore, I would like to thank the Ministry of Higher Education in Saudi Arabia for sponsoring my complete journey of studies. As well great thanks to the Saudi Cultural Bureau in Ottawa for their continuous support, guidance, personal following and financial coverage.

In addition, my gratitude to the computer science committee for reviewing my research and their feedbacks. Moreover, big thanks to the administrative staff and technical support for their tremendous cooperation and assistance. I would like to thank my colleagues whom contributed a huge inspirations and guidance.

No thanks would be enough to my parents, sister, parents in law, and my husband for their endless love, continuous support, and prayers

Dedication

Dad: Abdulrahman & Mom: Aatika
Sister: Naeemah & Niece: Ghazal
My supportive husband: Hassan Bazaid & My expected babies ♡

Contents

List of Tables

List of Figures

Chapter 1

Introduction

One of the most important services in cloud computing is data storage, which allows users to store their data in the cloud. Although cloud storage offers many advantages, it also introduces new security challenges in data integrity and availability. To verify the integrity of data stored in the cloud and to save computational resources of cloud users, it is important to enable auditing services, including those done on behalf of users by the TPA that can check the integrity of their data(Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

To increase storage efficiency, storage providers often identify and remove redundant data and keep only one copy of each file (file-level deduplication) or block (block-level deduplication). Data deduplication may occur before the data is transmitted to the cloud (client-side deduplication) or after it is transmitted (server-side deduplication) (Harnik et al.(2010)Harnik, Pinkas, and Shulman-Peleg). Standard server-side deduplication, in particular those associated with Cloud Service Providers (CSPs), requires full access to the content of users' data, which may limit the types of information that can be stored by these types of services. In particular, storing sensitive organizational data may not be appropriate. Furthermore, although deduplication can be used with single user data, it has been reported (Soghoian(2011)) that an average of 60% of data can be deduplicated for individual users by using a cross-user deduplication technique that identifies redundant data among different users.

In this thesis, we use the results from (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou) and (Boldyreva(2003)) to propose a privacy-preserving public auditing protocol, which enables cross-user data deduplication achieving both data integrity and storage efficiency. At enterprise level, it is more likely that users have identical documents such as email attachments. So, cross-user data deduplication is provided by our scheme before outsourcing the data to the cloud. Moreover, our scheme eliminates not only the duplicated data but also the signatures that are used for checking data integrity. To ensure publicly verifiable data deduplication, our protocol uses an enterprise level mediator who has two main tasks to perform. Its first task is to do a client-side deduplication to eliminate duplicated blocks, so the amount of uploaded data and the bandwidth used between the enterprise and the CSP are both reduced. Since users should have the ability to check the integrity of their own stored data, the mediator's second task helps with calculating aggregated signatures that can be used by the TPA to perform its task.

In our protocol, we only assume that the TPA and the CSP are semi-trusted, and we will prove that the TPA would not learn the content of users' data while performing its tasks. Our analytical and experimental results show the efficiency of our proposed protocol. Then, we extend our work to support user revocation where one of the users wants to leave the enterprise. The signatures signed under his private key need to be re-signed by another user due to security reasons. Our extended work allows the cloud server to re-sign the blocks of the revoked user.

Thesis Outline: The rest of this chapter provides a brief introduction of cloud computing including its definition, characteristics, service models, deployment models, and security issues. It also provides an overview of the concept of data deduplication technique including definition, ratio, benefits, and types. Then, we explain some notations and preliminaries which are used in some of the literature and the proposed scheme.

Chapter 2 Literature Survey: To give a better understanding of our proposed scheme, this chapter firstly explains auditing, which means checking the integrity of the data stored in the cloud. It includes auditing types, several important requirements in designing auditing protocols, and auditing techniques that have been proposed in the literature. Secondly, an extensive survey of Proof of Storage (POS) techniques is provided including definitions, algorithms, features, and drawbacks. Next, we present some schemes related to public auditing in the cloud with data deduplication. We include definitions, algorithms, features, and drawbacks. Finally, we provide a brief comparative survey between aggregate signature and multisignature in order to choose which scheme best fits the needs of our proposal.

Chapter 3 Design and Implementation: This chapter describes in detail our proposed schemes that achieve both data integrity and storage efficiency. It includes an overview of our problem statement, the system model, and the threat model. It also describes the design goals of the proposed scheme. Then a detailed description of our scheme is provided. We present an extension to support batch auditing where the TPA performs many auditing tasks from different users at the same time. Another extension is presented to support user revocation.

Chapter 4 Evaluation: This chapter presents a detailed analysis of the security and the performance of our proposed schemes. It also shows the efficiency of the batch auditing technique which is seen in the fact that the number of pairing operations in the batch audit is less than if each task were performed as an individually.

Chapter 5 Conclusions: This Chapter presents our conclusion and some potential directions for future work.

1.1 Cloud Computing

Cloud Computing is one of the most inspiring concepts of the IT industry. It has gained attention due to its offered resources and services to enterprises and consumers.

1.1.1 Cloud Computing Definition

The most accurate definition of cloud computing is given by the National Institute of Standard and Technology (NIST). "Cloud Computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" (Mell and Grance(2011)).

1.1.2 Cloud Computing Characteristics

According to NIST (Mell and Grance(2011)), there are five essential characteristics of cloud computing, which are:

1. On-demand self-service: A consumer can provision computing capabilities by himself with no need for any human interaction with any service provider.
2. Broad network access: Cloud capabilities are available through standard internet-enabled devices such as desktops, laptops, smart-phones, and tablet devices.
3. Resource pooling: Resource pooling assigns computing resources such as storage, processing, memory, and network bandwidth to multiple customers dynamically, based on consumer demand.
4. Rapid elasticity: A consumer can increase or decrease the capabilities of cloud services at any time based on demand.
5. Measured service: A consumer is charged based on the amount of usage of services (e.g., storage, processing, bandwidth, and so on). This is known as pay-as-you-go or pay-per-use.

1.1.3 Cloud Computing Service Models

There are three service models (Mell and Grance(2011)):

- Software as a Service (SaaS): The consumer can use application software in the cloud.
- Platform as a Service (PaaS): The consumer can use platform of their choice to run their applications on the cloud.
- Infrastructure as a Service (IaaS): The consumer can access the underlying cloud infrastructure for their use.

1.1.4 Cloud Computing Deployment Models

There are four models (Mell and Grance(2011)):

1. Private Cloud: The cloud infrastructure is provided to a single corporation. It may be possessed, administrated and operated by the same corporation, a third party or by both, and it may exist inside or outside the premises (corporation's building).

2. Community Cloud: Cloud infrastructure is provided to a specific group of consumers from corporations with the same demand and objectives. It may be owned, administered, and operated by one or more corporations, a third party or both, and it may exist inside or outside the premises.

3. Public Cloud: Cloud infrastructure is provided to the general public. It may be owned, administered, and operated by any corporation, and it exists inside the premises of the cloud provider.

4. Hybrid Cloud: Cloud infrastructure is a combination of two or more different cloud infrastructures (private, community, or public).

1.1.5 Cloud Computing Security Issues

Cloud storage is one of the most commonly used application services in the cloud that provides data storage and business access services externally (Li et al.(2011)Li, Xu, Li, and Zhang). However, the security of the data stored in the cloud of increasing concerned. The consumer gains several advantages from storing his data in the cloud such as availability, reliability, efficient retrieval, data sharing, and avoiding maintenance costs (Kamara and Lauter(2010)). On the other hand, cloud storage also has several disadvantages such as network outage, incompatible networking protocols, and unauthorized access. Briefly, the main security concerns are (Delettre et al.(2011)Delettre, Boudaoud, and Riveill):

- Confidentiality of stored data.
- Privacy.
- Availability of services and/or data.
- Integrity of services and/or data.
- Loss control of services and/or data.

1.2 Data Deduplication

1.2.1 Data Deduplication Definition

Data Deduplication is a technique for eliminating the redundant data and storing only one copy of the duplicate data. Pointers are created to provide a link from the duplicate data to the unique copy (Tate et al.(2012)Tate, Beck, Hugo Ibarra, Kumaravel, and Miklas)

1.2.2 Data Deduplication Ratio

The *data duplication ratio* refers to the relationship between the size of data sent to be deduplicated and the size of the space used to store it, see Figure1.1.

$$\text{Ratio} = \frac{\text{Bytes In}}{\text{Bytes Out}}$$

Figure 1.1: Data Deduplication Ratio

1.2.3 How Data Deduplication works

According to (Rivera(2009)),

- The data is broken up into segments.
- An identifier is created for each segment.
- Identifying the duplicated data by comparing the data segment identifiers
- Only one copy of each duplicated segment is stored.
- The duplicated segments are not stored and pointers are created for them.

1.2.4 Data Deduplication Benefits

According to (Rivera(2009)), there are several benefits of data deduplication which are:

- Increased storage efficiency.
- Increased network efficiency.
- Decreased hardware costs.
- Decreased backup costs.
- Decreased costs for business continuity/disaster recovery.

1.2.5 Data Deduplication Types

The data deduplication can be categorized depending to various aspects
(Rivera(2009); Harnik et al.(2010)Harnik, Pinkas, and Shulman-Peleg; Singh(2009)):

- Location/site: where the deduplication occurs.
  1. Source-based Deduplication: acts on the data at the source (client side) before it is transmitted to the data storage.
  2. Target-based Deduplication: acts on the data at the target (data storage device or service).
- Time: When the deduplication occurs.
  1. In-line Deduplication: acts as the data is written to the data storage system
  2. Post-process Deduplication: acts after the data is written to the data storage system

- Method: How the deduplication is performed.
  1. Hash-based Deduplication: A hash algorithm such as SHA3 is used to create an identifier for the segment.
  2. Delta-based Deduplication: the data is sored in the form of differences from a standard copy.
- Granularity: How the data is broken down/divided.
  1. File-level Deduplication: is applied on full (complete) file. Only one copy of each file is stored.
  2. Block-level Deduplication: is applied on a sub-file (a file is broken up into blocks). Only one copy of each block is stored.
- Alignment: How the data is arranged.
  1. Fixed Length Block: The file is divided into fixed-length blocks
  2. Variable Length Block: The file is divided into variable-length blocks.

1.3 Security Assumptions

In cryptography, we have two security models:

Random Oracle Model: The random oracle is a random function "black box" where all the parties are responded with a random values from its output range (Ananth and Bhaskar(2013)).

Standard Model: The standard model of computation where it depends on the standard complexity theoretic assumption (Bellare and Rogaway(1993)).

Security proofs of many given protocols is based on intractability of problems considered to be hard to solve. The problem considered in most discrete logarithm based protocol is the *Diffie-Hellman* problem and one of its variants. Two are such variants are *Decision Diffie-Hellman (DDH)* and *Computational Diffie-Hellman (CDH).*

Consider a multiplicative cyclic group G, of prime order p and with a generator g.

Definition 1. Decision Diffie-Hellman (DDH): Given $a, b, c \in \mathbb{Z}_p^*$, and (g, g^a, g^b, g^c), it is difficult to determine whether $c = ab$.

Definition 2. Computational Diffie-Hellman (CDH): $a, b \in \mathbb{Z}_p^*$, and (g, g^a, g^b), it is difficult to compute g^{ab} without knowing a or b.

Any (g, g^a, g^b, g^c) or (g, g^a, g^b) satisfying the stated problems, respectively are referred to as *Diffie-Hellman (DH) tuple.*

The short signatures used in the proposed scheme in this thesis are based on the Weil pairing in (Boneh et al.()Boneh, Lynn, and Shacham), and the aggregate, verifiable signature in (Boneh et al.(2003)Boneh, Gentry, Lynn, and Shacham). The security assumption in these papers are based on *Gap Diffie-Hellman (GDH)* in which the CDH problem is considered hard, but the DDH problem is considered easy to solve.

Next, we give the definition of bilinear maps, the short signature scheme in (Boneh et al.()Boneh, Lynn, and Shacham), and the aggregate signature of (Boneh et al.(2003)Boneh, Gentry, Lynn, and Shacham). This closely follows the discussions in these two papers, and the reader is referred to those papers for more details.

Definition 3. Bilinear Maps: Let G_1, G_2 and G_T be multiplicative cyclic groups of prime order p. g_1 is a generator of G_1 and g_2 is a generator of G_2. A bilinear map is a map $G_1 \times G_2 \rightarrow G_T$ with the following properties (Boneh et al.()Boneh, Lynn, and Shacham):

- Bilinear: for all $u \in G_1, v \in G_2$ and $a, b \in \mathbb{Z}_p$, $e(u^a, v^b) = e(u, v)^{ab}$.
- Non-degenerate: $e(g_1, g_2) \neq 1$.

It is easy to extend the DH problem over these types of maps. When $G_1 \neq G_2$, they are referred to as *co-DDH*, *co-CDH*, and *co-GDH*. The next two signature schemes are proved to be secure in a the Random Oracle Model, based on the co-GDH assumption.

1.3.1 Boneh, Lynn, and Shacham (BLS) signature

The scheme is proposed by Boneh et. al. (Boneh et al.()Boneh, Lynn, and Shacham).

Definition 4. Let (G_1, G_2) be a co-Gap Diffie-Hellman group pair where $|G_1| = |G_2| = p$. A signature σ is an element of G_1. $H : \{0,1\}^* \rightarrow G_1$ is a hash function (random oracle). The scheme has three algorithms:

KeyGen: This algorithm produces the secret and the public key used by the user for the signature generation, and the public key used by the signature verifier. Select at random $x \in \mathbb{Z}_p$ and compute $v \leftarrow g_2^x$. The secret key used is x and the public key is $v \in G_2$.

Sign: This algorithm will generate the signature for a message $M \in \{0,1\}^*$, using the secret key x. Compute $h = H(M) \in G_1$. The signature will be $\sigma = h^x \in G_1$.

Verify: This algorithm will allow the verifier to determine whether the signature is genuine. Compute $h = H(M) \in G_1$. Verify that (g_2, v, h, σ) is a valid co-Diffie-Hellman tuple.

Aggregate Signature: This scheme makes it possible to aggregate multiple signatures from distinct users on distinct messages into a single short signature (Boneh et al.(2003)Boneh, Gentry, Lynn, and Shacham). It has two additional algorithms: *Aggregation*, and *Aggregate Verification*.

Suppose that we need to generate an aggregated signature for messaged $M_i \in \{0,1\}^*$, and a subset of all users U, where $|U| = k$, and $i = 1, \cdots, k$.

Aggregation: This algorithm will generate the aggregated signature for messages M_i and users u_i with secret keys x_i and public keys $v_i \in G_2$. Compute the signature $\sigma_i \in G_1$ for each distinct message M_i. The aggregate signature $\sigma = \prod_{i=1}^{k} \sigma_i$.

Aggregate Verification: This algorithm will allow the verifier to determine whether the aggregated signature is genuine. Compute $h = H(M) \in G_1$. To do this, the verifier checks the following:

- Checks to ensure that all M_i are distinct, and
- Computes $h_i = H(M_i)$ for $i = 1, \cdots, k$, and checks to see if $e(\sigma, g_2) = \prod_{i=1}^{k} e(h_i, v_i)$.

Chapter 2

Literature Survey

In this chapter, we explain auditing, which means checking the integrity of the data stored in the cloud: types, important requirements, and techniques. We give an extensive survey of those techniques: definitions, algorithms, features ,and drawbacks. Next we present related work of public auditing in the cloud with data deduplication: definitions, algorithms, features, and drawbacks. Finally, we provide a brief comparison between aggregate signature and multisignature in order to choose which one is more suitable to our proposed scheme.

2.1 Data Auditing in Cloud Computing

One of the most significant services of Cloud Computing is data storage which allows users to move their data to the cloud. Although cloud storage has many advantages, it also introduces new security challenges as data may be accessed by unauthorized users or lost in the cloud. We need to check the integrity of our stored data in the cloud for two reasons: Cloud services are still suffering outages and security breaches and CSPs may delete some of the data to obtain storage space. Due to the limited computing resources of cloud users and the large size of outsourced data, enabling public auditing is the best solution (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

2.1.1 Data Auditing Types

According to (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song) and (Yang and Jia(2012)), there are two types of auditing of cloud storage:

1. Data Owner Auditing: also known as Private Verifiability/Auditing, when the data owner verifies the integrity of the data stored in the cloud server. The system model consists of two entities: the data owner and the cloud server, as shown in Figure 2.1.

2. Third Party Auditing: also known as Public Verifiability/Auditing, when the TPA, instead of the data owner, verifies the integrity of the data stored in the cloud server. The system model consists

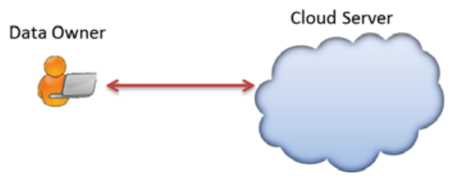


Figure 2.1: The system model of data owner auditing

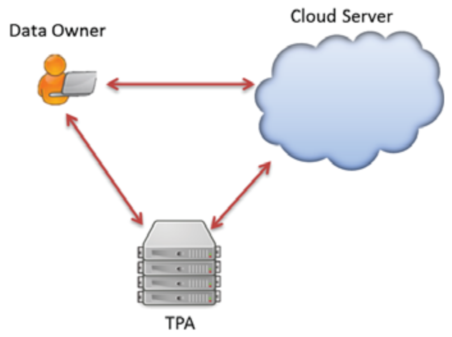


Figure 2.2: The system model of third party auditing

of three entities: the data owner, the cloud server, and the TPA, as shown in Figure 2.2.

2.1.2 Data Auditing Requirements

To construct an efficient auditing protocol, there are several requirements must be considered according to (Yang and Jia(2012)),(Worku et al.(2012)Worku, Ting, and Zhi-Guang), and (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou):

- Storage correctness / Unforgeability: ensures that the cloud cannot cheat and pass the auditing process without storing the data properly.
- Low storage cost: the additional storage needs for the auditing should be low. That depends on: type of metadata (e.g MAC, signature, tag, etc.), size of metadata , and where to store the metadata (auditor side or server side which is preferred).
- Low communication cost: the amount of communication between the parties needs for the auditing should be low.
- Low computation cost: the complexity of the computation needs for the auditing should be low. The computation cost on the data owner depends on pre-prossessing the data (e.g. error correcting code) and calculating the metadata. On the other hand, the computation cost on the auditor and the server depends on the computation of auditing process.
- Unbounded number of audits: the number of challenging the server should be unlimited. If the number is limited, the data owner needs to download the data from the server and re-computes

the metadata. The server may cheat by responding to the challenge with a previous respond if the same challenge is issued before.

- Recoverability: the ability to recover the data if failure occurred. It depends on applying error correcting code or erasure code to the data before it is sent to the server.
- Public auditing: enables the TPA to check the integrity of the stored data in the server on behalf of the user.
- Batch auditing: allows the TPA to perform multiple auditing tasks at the same time from different users.
- Blockless verification: the auditor should not retrieve any data blocks from the server during the auditing process.
- Stateless verification: the TPA does not need to maintain and update state between the auditing process. In case of failure, it is difficult to maintain state (Shacham and Waters(2008)).
- Privacy-preserving: ensures that the TPA cannot learn anything about the content of the stored data during the auditing process.
- Dynamic data: the scheme should support dynamic data operations such as data modification, insertion and deletion. The dynamic operating cost should be low.

2.1.3 Data Auditing Techniques

In this chapter, we give a review on the techniques that used to check the data integrity at untrusted server. We explain Proof of Storage protocols such as Remote Data Integrity Checking (RIC), Provable Data Possession (PDP), and Proof of Retrievability (POR). Some of theses protocols related to private verifiability and the others related to public verifiability.
There are three categories of data auditing (Yang and Jia(2012)):

1. Message Authentication Code (MAC)-based methods.
2. RSA-based Homomorphic methods.
    - RSA-based Homomorphic hash value.
    - RSA-based Homomorphic tag.
3. Boneh-Lynn-Shacham signature (BLS)-based Homomorphic methods.

MAC-Based Schemes

The simple way is the data owner computes a Message Authentication Code (MAC) of his file and sends the file to untrusted server and keeps both the computed MAC and the secret key. Whenever he wants to check the integrity of the file, he retrieves the file and recomputes the MAC to compare it with his

stored version. This way has a severe drawback which is downloading the file every time he needs to check its verification.
Another way is the data owner divides the file into blocks and computes a MAC for each block. He sends both the file and the MACs to the server and keeps only the secret key. The user may share the secret key with the TPA. Later, the owner or TPA can retrieve from the server a random number of data blocks with their MACs and check the data integrity by comparing the fresh MACs with the stored ones. This way has two drawbacks: the communication cost is linear with the size of sampled blocks and the TPA needs to know the content of the blocks for the verification(Zeng(2008); Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).
To avoid retrieving the data blocks from the server, the data owner selects randomly multiple secret keys and computes multiple MACs for the file. He sends the the file to the server and sends the MACs and the keys to the TPA. Later, the owner or TPA sends every time one of the keys to the server and asks for a fresh MAC. Although this way preserve the privacy of owner's data, it has two drawbacks: bounded number of challenges because the total number of challenges depends on the total number of MAC secret keys, and the owner or TPA needs to maintain and update state between auditing process (Zeng(2008); Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

Provable Data Possession (PDP)

PDP is a technique/protocol that allows the user who stores his data at untrusted server to check if the server indeed retains the data without retrieving it by the verifier and without accessing the whole data by the server (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song).

Protocol Overview: PDP protocol allows the user who wants to outsource his file into untrusted storage server to check if the server possess the file. The user pre-processes the file, which consists of n blocks, and generates a metadata. Then, he sends the file with the metadata to the server and may delete the local copy. At a later time, the user checks if the server retains the file by issuing a challenge to the server who computes a proof and sends it back to the user. Finally, the user verifies the response without retrieving the file blocks (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song). PDP schemes provide probabilistic and deterministic guarantees. The probabilistic guarantee utilizes a sampling technique which means the server generates the proof of data possession by accessing a random set of blocks. On the other hand, in the deterministic guarantee, all the blocks are accessed by the server(Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song).
The concept of Homomorphic Verifiable Tag (HVT)/Homomorphic Linear Authenticator (HLA) is introduced to be as the building block for the PDP schemes. According to (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou), HVTs/HLAs are unforgeable verification metadata used to check the integrity of the data blocks. The HVTs/HLAs can be aggregated to verify a linear combination of the individual data blocks. The HVTs/HLAs have another property which is *Blockless verification*, means the user verifies if the server retains the file without accessing or retrieving the file blocks.

Definition 5 (Provable Data Possession (PDP) Scheme)**.** A PDP scheme (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song) has four polynomial-time algorithms (KeyGen, TagBlock, GenProof, and CheckProof), and it consists of two phases: Setup (includes the first two algorithms) and Challenge (includes the last two algorithms).

KeyGen $(1^k) \rightarrow (pk, sk)$ is a key generation algorithm that is run by the user. It takes a security parameter k as input, and returns a pair of public and secret keys (pk, sk).

TagBlock $(pk, sk, m) \rightarrow T_m$ is a metadata generation algorithm that is run by the user. It takes a public key pk, a secret key sk, and a file block m as input, and returns the verification metadata T_m.

GenProof$(pk, F, chal, \sum) \rightarrow \mathcal{V}$ is a proof of possession algorithm that is run by the server. It takes a public key pk, a collection of blocks F, a challenge $chal$, and a collection of $\sum$ which is the verification metadata corresponding to the blocks in F. It returns a proof of possession $\mathcal{V}$.

CheckProof $(pk, sk, chal, \mathcal{V}) \rightarrow \{success, failure\}$ is run by the verifier to validate the proof. It takes a public key pk, a secret key sk, a challenge $chal$, and a proof of possession $\mathcal{V}$.

The authors of (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song) present two PDP constructions: the first scheme is Sampling Provable Data Possession (S-PDP) with strong data possession guarantee. The second scheme is Efficient PDP (E-PDP) with better efficiency and weaker guarantee.

S-PDP Scheme Details: Let k, ℓ, λ be security parameters. Let $N = pq$ be an RSA modulus with p and q prime numbers. Let g be a generator of QR_N which is the set of quadratic residues modulo N. Let $\mathbb{Z}_{\mathbb{N}}^*$ be a multiplicative cyclic group. Let $h\{0,1\}^* \rightarrow QR_N$ be a secure deterministic hash-and-encode function that maps strings uniformly to QR_N. Let H be a cryptographic hash function. let f be a pseudo-random function (PRF), let π be a pseudo-random permutation (PRP).

- Setup:
  - KeyGen (1^k): $pk = (N, g)$ and $sk = (e, d, v)$, such that $ed \equiv 1 \mod (p-1)(q-1)$, e is a large secret prime such that $e > \lambda$ and $d > \lambda$, $v \overset{R}{\leftarrow} \{0,1\}^k$.
  - TagBlock (pk, sk, m, i): For each block of $F = \{m_1, \cdots, m_n\}$, compute a tag

    $$T_{i,m} = (h(W_i) \times g^m)^d \mod N$$

    , where $W_i = v||i$
  - The user sends the file F and the tags T to the server.
- Challenge:
  - The user chooses randomly two keys k_1 for π, k_2 for f, c is number of blocks to be checked, and $g_s = g^s \mod N$. Then, he sends $chal = \{c, k_1, k_2, g_s\}$ to the server.

– GenProof ($pk, F = (m_1, \cdots, m_n), chal, \sum = (T_{1,m}, \cdots, T_{n,m})$): For $1 \leq j \leq c$: the server computes the indices: $i_j = \pi_{k_1}(j)$, and the coefficients: $a_j = f_{k_2}(j)$. Then, he computes $T = \prod_{j=1}^{c} T_{j,m}^{a_j}$ and $\rho = H(g_s{}^{\sum_{j=1}^{c} m_j a_j} \mod N)$. The server sends $\mathcal{V} = (T, \rho)$ to the user.
– CheckProof ($pk, sk, chal, \mathcal{V}$): The user computes $i_j = \pi_{k_1}(j)$, $a_j = f_{k_2}(j)$, $\tau = \frac{T^e}{\prod_{j=1}^{c} h(W_{i_j})^{a_j}}$ mod N. Then, he checks $H(\tau^s \mod N) \overset{?}{=} \rho$

The only difference between the two schemes is that the E-PDP provides guarantee of the sum of the data blocks not each individual block as in S-PDP. Thus, all the coefficients a_j are equal to 1. The PDP scheme can be modified to offer Public Verifiability property, which allows anyone, not only the data owner, to verify the correctness of the stored data (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song).

Features of the scheme: PDP scheme provides less access to the file blocks, less computation on the server, less communication between the user and the server, and unbounded number of challenges with constant amount of data (Ateniese et al.(2007)Ateniese, Burns, Curtmola, Herring, Kissner, Peterson, and Song).

Drawbacks of the scheme: Since the HLAs are based on RSA, that makes the HLAs relatively long. Although the PDP scheme provides Public Verifiability, it does not support privacy-preserving and Batch auditing properties.

Proof of Retrievability (POR)

POR is a protocol that allows the user who stores his data at untrusted server to check if the server indeed retains the data in which the user can retrieve the entire data (Juels and Kaliski Jr.(2007)).

Protocol Overview: POR protocol allows the user who wants to outsource his file into untrusted storage server to check if the server possess the file. The user encrypts the file and inserts random values which are called *sentinels*. In addition, Error-Correcting Code (ECC) is applied to the file to recover a small corruption. Then, he sends the file to the server and may deletes the local copy. At a later time, the user challenges the server by asking to return specific sentinel values. The server computes a proof and sends it back to the user. Finally, the user verifies the response without retrieving the file blocks. POR scheme provides a probabilistic guarantee (Juels and Kaliski Jr.(2007)).

Definition 6 (Proof of Retrievability (POR))**.** A POR scheme (Juels and Kaliski Jr.(2007)) has six algorithms (keygen, encode, extract, challenge, respond, and verify), and it consists of two phases: Setup and Verification.

Sentinel-based POR Scheme Details:

- Setup phase: a secret key is generated by *keygen*. *encode* function has four steps: the file F is divided into k blocks. For each block an ECC is applied, so that yields a file F'. Then, a symmetric key encryption is applied to F', that yields file F''. Next, sentinels are created and appended to F'', that yields file F'''. Finally, a pseudorandom permutation (PRP) is applied to F''', that yields file F'''' which is sent to the server.

- Verification phase: the user runs *challenge* to generate q positions for different sentinels and sends them to the server. The server sends back the values of the corresponding sentinels as a *respond* to the user who runs *verify* to check if the server returned correct values. Then, the file F is recovered by *extract* function.

Features of the scheme: The file is recovered by applying ECC. The communication and computation costs of the scheme are low (Juels and Kaliski Jr.(2007)).

Drawbacks of the scheme: POR scheme allows bounded number of challenges because the total number of challenges depends on the total number of sentinels (Juels and Kaliski Jr.(2007)).

Compact Proofs of Retrievability-2008

In this paper (Shacham and Waters(2008)), the authors present two schemes that rely on the Homomorphic Authenticators (HAs). The first scheme is based on pseudorandom functions (PRFs), is secure in the standard model, and offers private verifiability. The second scheme is based on Boneh-Lynn-Shacham (BLS) signature, is secure in the random oracle model, and offers public verifiability.

Definition 7 (Compact Proof of Retrievability (POR))**.** A Compact POR scheme (Shacham and Waters(2008)) has four algorithms (Kg, St, $\mathcal{P}$ and $\mathcal{V}$).

PRF-based POR Scheme Details: Let $f : \{0,1\}^* \times \mathcal{K}_{prf} \to \mathbb{Z}_p$ be a PRF.

Kg(): Select randomly symmetric encryption key and MAC key. sk will be (k_{enc}, k_{mac}) and there is no pk.

St(sk, M)**:** Pre-process the file M by applying the erasure code, that yields a file M'. Then, the file M' is divided into n blocks. Each block is divided into s sectors, $\{m_{ij}\}_{1\leq i\leq n, 1\leq j\leq s}$. Choose a PRF key k_{prf} and s random numbers $\alpha_1, \cdots, \alpha_s \to \mathbb{Z}_p$. Let $\tau_0 = n \parallel Enc_{k_{enc}}(k_{prf} \parallel \alpha_1 \parallel \cdots \parallel \alpha_s)$, the file tag $\tau = \tau_0 \parallel MAC_{k_{mac}}(\tau_0)$. Compute the authenticator for each block i as $\sigma_i = f_{k_{prf}}(i) + \sum_{j=1}^{s} \alpha_j m_{ij}$. The processed file M^* is $\{m_{ij}\}$ together with $\{\sigma_i\}$. The file M^* is stored on the server along with the file tag τ.

$\mathcal{V}(pk, sk, \tau)$**:** k_{mac} is used to verify the MAC on τ. k_{enc} is used to decrypt the encrypted portions and recover $n, k_{prf}, and(\alpha_1, \cdots, \alpha_s)$. Pick a random l–element subset I of the set $[1, n]$. For each $i \in I$, select a random element v_i. Then, send $Q = \{(i, v_i)\}$ to the prover. Check the prover's response $\mu_1, \cdots, \mu_s$ and σ via

$$\sigma \overset{?}{=} \sum_{(i,v_i)\in Q} v_i f_{k_{prf}}(i) + \sum_{j=1}^{s} \alpha_j \mu_j$$

$\mathcal{P}(pk, \tau, M^*)$**:** Compute $\mu_j = \sum_{(i,v_i)\in Q} v_i m_{ij}$ for $1 \leq j \leq s$, $\sigma = \sum_{(i,v_i)\in Q} v_i^{\sigma_i}$

BLS-based POR Scheme Details: Let $e : G \times G \to G_T$ be a bilinear map, g be agenerator of G, and $H : \{0,1\}^* \to G$ be the BLS hash which is treated as a random oracle (Shacham and Waters(2008)).

Kg(): Generate a random signing key pair (spk, ssk). Select a random $\alpha \to \mathbb{Z}_p$ and compute $v \to g^{\alpha}$. sk will be (α, ssk) and pk will be (v, spk).

St(sk, M)**:** Pre-process the file M by applying the erasure code, that yields a file M'. Then, the file M' is divided into n blocks. Each block is divided into s sectors, $\{m_{ij}\}_{1\leq i\leq n, 1\leq j\leq s}$. Choose a random file name $name$. Choose s random elements $u_1, \cdots, u_s \to G$. Let $\tau_0 = name \parallel n \parallel u_1 \parallel \cdots \parallel u_s$, the file tag $\tau = \tau_0 \parallel SSig_{ssk}(\tau_0)$. Compute the authenticator for each block i as

$$\sigma_i = (H(name \parallel i) \times \prod_{j=1}^{s} u_j^{m_{ij}})^{\alpha}$$

The processed file M^* is $\{m_{ij}\}$ together with $\{\sigma_i\}$. The file M^* is stored on the server along with the file tag τ.

$\mathcal{V}(pk, sk, \tau)$**:** spk is used to verify the signature on τ and recover $name, n$, and $(u_1, \cdots, u_s)$. Pick a random $l-$ element subset I of the set $[1, n]$. For each $i \in I$, select a random element v_i. Then, send $Q = \{(i, v_i)\}$ to the prover. Check the prover's response $\mu_1, \cdots, \mu_s$ and σ via

$$e(\sigma, g) \stackrel{?}{=} e(\prod_{(i,v_i)\in Q} H(name \parallel i)^{v_i} \times \prod_{j=1}^{s} u_j^{\mu_j}, v)$$

$\mathcal{P}(pk, \tau, M^*)$**:** Compute $\mu_j = \sum_{(i,v_i)\in Q} v_i m_{ij}$ for $1 \leq j \leq s$, $\sigma = \sum_{(i,v_i)\in Q} v_i^{\sigma_i}$

Features of the scheme: The server's response is short due to the homomorphic properties that aggregate the proof into one authenticator value (Shacham and Waters(2008)).

Drawbacks of the scheme: By sending the server's response, which consists of the linear combination of the sampled blocks, to the verifier, the user's data may leak to the verifier. Hence, the public verification scheme does not support privacy-preserving property (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

Privacy-Preserving Schemes

In these papers (Shah et al.(2008)Shah, Swaminathan, and Baker; Shah et al.(2007)Shah, Baker, Mogul, and Swaminathan), the proposed protocol allows the TPA to check the integrity of the stored data and assist in returning the data intact to the user. The protocol supports privacy-preservation property, so the TPA cannot learn the content of user's data.

Protocol Overview: The user who wants to store his data to untrusted server allows the TPA to verify the stored data without revealing the data content to the auditor. The user encrypts the data with a secret key and sends both of them to the server. He also sends the encrypted data with a key-commitment, which fixes a value for the key without revealing the key, to the TPA. The TPA can check if the server has intact both the encrypted data and encryption key without learn any information about the key or the data. He also assists in returning the key and encrypted data to the user in a privacy manner (Shah et al.(2008)Shah, Swaminathan, and Baker; Shah et al.(2007)Shah, Baker, Mogul, and Swaminathan).

Definition 8 (Privacy-preserving PDP scheme)**.** The scheme consists of three phases: initialization, audit, and extraction (Shah et al.(2008)Shah, Swaminathan, and Baker; Shah et al.(2007)Shah, Baker, Mogul, and Swaminathan).

Privacy-Preserving PDP Scheme Details:

- The user encrypts the data $E_K(M)$ with the secret key K, then sends both of them to the server. The user sends $E_K(M)$ with a key-commitment g^K to the TPA. Upon the agreement between the parties, the server sends the key-commitment g^K, and a hash of the encrypted data $H(E_K(M))$. The TPA checks whether both the user and the server agree on a common key and encrypted data.
- To verify the encrypted data, the TPA generates n random numbers $R_1, \cdots, R_n$ and computes L hashes $\tilde{H}_1, \cdots, \tilde{H}_n$, where $\tilde{H}_i = HMAC(R_i, E_K(M))$. The TPA keeps the pairs $L = \{(R_1, \tilde{H}_1), \cdots, (R_n, \tilde{H}_n)\}$ and deletes the encrypted data. The TPA chooses randomly $(R_j, \tilde{H}_j)$ from L and now $L = L \setminus \{(R_j, \tilde{H}_j)\}$, and sends it to the server as a challenge. The server computes the respond $\tilde{H}_s = HMAC(R_j, E_K(M))$ and sends it to the TPA. The TPA checks if $\tilde{H}_s = \tilde{H}_j$.
- To verify the encryption key, the TPA selects randomly β, computes g^β, and sends it to the server. The server computes $W_s = g^{\beta K}$ and sends it to the TPA who computes $W_a = (g^K)^\beta$ and checks if $W_a = W_s$.
- To extract the encrypted data, the server sends $E_K(M)$ to the TPA who checks whether his local copy of the encrypted data matches what he receives. If so, he sends the $E_K(M)$ to the user.
- To extract the encrypted key, a trusted fourth party generates a random secret key R and sends it to the server and user. The trusted fourth party also sends a secret-commitment key g^R to the TPA. The server sends blinded version of the key $B_s = K + R$ to the TPA who checks the blinded-key using the key-commitment and secret-commitment as $g^{B_s} = g^K g^R = g^{K+R}$ and sends B_s to the user. The user computes $B_s - R = K$ and recovers the original key.

Features of the scheme: The scheme allows public auditing of the user's data and the encryption key while supports privacy-preservation property. Moreover, the protocol can extract the digital contents from the server and deliver it to the user (Shah et al.(2008)Shah, Swaminathan, and Baker; Shah et al.(2007)Shah, Baker, Mogul, and Swaminathan).

Drawbacks of the scheme: It allows bounded number of challenges because the total number of challenges depends on the number of list values L. The TPA needs to maintain and update state between the auditing process, so the scheme does not support stateless verification (Shah et al.(2008)Shah, Swaminathan, and Baker; Shah et al.(2007)Shah, Baker, Mogul, and Swaminathan).

The authors of (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou), propose a secure cloud storage system supporting privacy-preserving public auditing. Beside this, they provide an efficient TPA with multiple auditing tasks in a batch manner.
In their proposed protocol, they utilize public key based homomorphic linear authenticator (HLA), which is based on BLS-based POR Scheme, with random masking. The proposed protocol guarantees that the TPA could not learn any knowledge about the stored data content (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

Definition 9 (Privacy-preserving POR scheme)**.** The scheme consists of four algorithms: KeyGen, SigGen, GenProof, and VerifyProof. It consists of two phases: Setup (includes the first two algorithms) and Audit (includes the last two algorithms) (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

Privacy-Preserving POR Scheme Details: Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ be a bilinear map. Let g be a generator of $\mathbb{G}_2$. $H(\cdot)$ is a secure map-to-point hash function H:$\{0,1\}^* \to \mathbb{G}_1$, and $h(\cdot)$ is a hash function h: $\mathbb{G}_T \to \mathbb{Z}_p$.

- Setup phase:
  - KeyGen: The user chooses a random signing key (spk, ssk). He also selects a random $x \to \mathbb{Z}_p$, computes $v = g^x$, and selects a random $u \to \mathbb{G}_1$. The user's secret parameter is $sk = (x, ssk)$, and public parameters are $pk = (spk, v, g, u, e(u, v))$.
  - SigGen: Given a data file $F = \{m_i\}$. The user computes the signature of each block as $\sigma_i = (H(W_i) \times u^{m_i})^x \in \mathbb{G}_1$, where $W_i = name \parallel i$. $name \in \mathbb{Z}_p$ is chosen randomly as the file ID. The file tag is computed as $t = name \parallel SSig_{ssk}(name)$ Then, the user sends $\{F = \{m_i\}, \{\sigma_i\}, t\}$ to the server and deletes the local copy.
- Audit phase:
  - TPA retrieves the file tag t and verifies the signature. If it is true, he recovers $name$. TPA can check the integrity of files/blocks on behalf of the users by sending $chal = \{(i, \nu_i)\}$ to the cloud server, where $i \in I = \{s_1, \cdots, s_c\}$ for set of blocks $[1, n]$ and ν_i is a random value.
  - GenProof: The cloud server selects $r \in \mathbb{Z}_p$, and computes $R = e(u, v)^r \in \mathbb{G}_T$. CS computes the linear combination of the sampled blocks as $\mu' = \sum_{i \in I} \nu_i m_i$, and blind it with r as $\mu = r + \gamma\mu'$, where $\gamma = h(\mathcal{R})$. CS also computes an aggregated signature $\sigma = \prod_{i \in I} \sigma_i^{\nu_i}$. Then, CS sends $\{\mu, \sigma, \mathcal{R}\}$ to the TPA.
  - VerifyProof: The TPA computes $\gamma = h(\mathcal{R})$, and verifies $\{\mu, \sigma, \mathcal{R}\}$ via:

$$\mathcal{R} \times e(\sigma^\gamma, g) \stackrel{?}{=} e((\prod_{i \in I}(H(W_i)^{\nu_i})^\gamma \times u^\mu, v)$$

Features of the scheme: The scheme is efficient in terms of providing privacy-preserving public auditing and supporting batch auditing from different users. There is constant communication overhead for the server's response due to applying HLA technique (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

2.2 Public auditing with data deduplication Schemes

In this paper (Zheng and Xu(2012)), the authors propose Proof of Storage with Deduplication (POSD) scheme. The scheme provides secure and efficient cloud storage. To improve the security of cloud storage, PDP and POR are introduced for checking the verification of the stored data. To improve the efficiency of cloud storage, Proof of Ownership (POW), data deduplication is introduced to eliminate the duplicated data thus reduce communication and storage overhead. The deduplication is done on the cloud server. To enable that, the data will be stored in the cloud in plaintext form.

Protocol Overview: A user sends his file with its tag to the cloud. Then the user claims that he has a file in the cloud, which is already sent by another user. The cloud plays the role of the auditor (challenge-response protocol) and sends a challenge to the user. Then the user computes a response and sends it back to the cloud. The latter verifies it (Zheng and Xu(2012)).

Definition 10 (POSD scheme)**.** The scheme consists of four algorithms: Keygen, Upload, AuditInt, and Dedup. "POSD = PDP/POR + POW" (Zheng and Xu(2012)).

POSD Scheme Details: Let G and G_T are cyclic groups of prime order q, and $e : G \times G \to G_T$ be a bilinear map. $H_1 : \{0,1\}^* \to G$ and $H_2 : \{0,1\}^* \to \mathbb{Z}_q$ hash functions. The file F is divided into n blocks and m symbols in $\mathbb{Z}_q$, $F_i = (F_{i1}, \cdots, F_{im})$. fid is the file ID.

- Keygen: generates key pairs. Select randomly v_1 and v_2 from $\mathbb{Z}_p^*$, p is another prime. Select randomly s_{j1} and s_{j2} from $\mathbb{Z}_q^*$ for $1 \le j \le m$. Set $z_j = v_1^{-s_{j1}} v_2^{-s_{j2}} \mod p$ for $1 \le j \le m$. Let g be a generator of G. Select u randomly from G and w randomly from $\mathbb{Z}_q^*$, $z_g = g^w$. The verification key pairs set to be $PK_{int} = \{q, p, g, u, v_1, v_2, z_1, \cdots, z_m, z_g\}$, and $SK_{int} = \{(s_{11}, s_{12}), \cdots, (s_{m1}, s_{m2}), w\}$. The deduplication key set to be $PK_{dup} = PK_{int}$, and $SK_{dup} = null$.

- Upload: For each block F_i, the user selects randomly r_{i1} and r_{i2} from $\mathbb{Z}_q^*$. Then, computes $x_i = v_1^{r_{i1}} v_2^{r_{i2}} \mod p$, $y_{i1} = r_{i1} + \sum_{j=1}^{m} F_{ij} s_{j1} \mod q$, $y_{i2} = r_{i2} + \sum_{j=1}^{m} F_{ij} s_{j2} \mod q$, $t_i = (H_1(fid \parallel) \times u^{H_2(x_i)})^w$. The user sends (fid, F, Tag_{int}) to the server, where $Tag_{int} = \{(x_i, y_{i1}, y_{i2}, t_i)\}_{1 \le i \le n}$. The server sets $Tag_{dup} = Tag_{int}$.

- AuditInt: the auditor or the user verifies the integrity of the file F. The auditor chooses randomly c elements $I = \{\alpha_1, \cdots, \alpha_c\}$ set from $[1, n]$. He also chooses randomly $\beta = \{\beta_1, \cdots, \beta_c\}$, and sends $chal = (I, \beta)$ to the server. The server computes for $1 \le j \le m$, $\mu_j = \sum_{i \in I} \beta_i F_{ij} \mod q$, $Y_1 = \sum_{i \in I} \beta_i y_{i1} \mod q$, $Y_2 = \sum_{i \in I} \beta_i y_{i2} \mod q$, and $T = \prod_{i \in I} t_i^{\beta_i}$. The server sends the response as $resp = (\{\mu_j\}_{1\leq j\leq m}, \{x_i\}_{i\in I}, Y_1, Y_2, T)$ to the auditor. Then, the auditor computes $X = \prod_{i\in I} x_i^{\beta_i}$, $W = \prod_{i\in I} H_1(fid \parallel i)^{\beta_i}$, and verifies $X = v_1^{Y_1} v_2^{Y_2} \prod_{j=1}^{m} z_j^{\mu_j} \mod p$, $e(T, g) = e(Wu^{\sum_{i\in I}\beta_i H_2(x_i)}, z_g)$.

- Dedup: the user claims that he has a file that is already sent to the server by another user. The server sends a challenge $chal = (I, \beta)$ to the user. The user computes for $1 \leq j \leq m$, $\mu_j = \sum_{i\in I} \beta_i F_{ij} \mod q$, and sends it as $resp = (\{\mu_i\}_{1\leq i\leq m})$ to the server. The server computes $Y_1 = \sum_{i\in I} \beta_i y_{i1} \mod q$, $Y_2 = \sum_{i\in I} \beta_i y_{i2} \mod q$, $W = \prod_{i\in I} H_1(fid \parallel i)^{\beta_i}$, $X = \prod_{i\in I} x_i^{\beta_i}$, and $T = \prod_{i\in I} t_i^{\beta_i}$. The server verifies $X = v_1^{Y_1} v_2^{Y_2} \prod_{j=1}^{m} z_j^{\mu_j} \mod p$, $e(T, g) = e(Wu^{\sum_{i\in I}\beta_i H_2(x_i)}, z_g)$

Features of the scheme: The scheme allows public auditing with proof of ownership (Zheng and Xu(2012)).

Drawbacks of the scheme: The scheme does not support dynamic data(Zheng and Xu(2012)).

In this paper (Yuan and Yu(2013)), The author propose a Public and Constant cost storage integrity Auditing scheme with secure Deduplication (PCAD) that provides secure and efficient cloud storage. To ensure data integrity, PDP and POR techniques are used. To improve storage efficiency, POW is used to eliminate the duplicated data. The scheme is based on techniques including polynomial-based authentication tags and homomorphic linear authenticators. The deduplication is done on the cloud server side on both the files and their corresponding authentication tags by aggregating the tags of the same file from different owners. The system model of this scheme has four entities: Trust Authority (TA), Data Owner, Cloud Server and User/TPA (Yuan and Yu(2013)).

Protocol Overview: A user sends his file with its tag to the cloud. TPA can check the integrity of the user's file by challenge-response protocol. When another user wants to send the same file to the cloud, the latter has to check if this user has indeed the same file (or if he is also the owner of the file) by sending a challenge. Then, the user computes a proof and sends it back to the cloud. The cloud verifies it. If the verification equation holds, the user becomes an owner of the file (Yuan and Yu(2013)).

Definition 11 (PCAD scheme)**.** The scheme consists of six algorithms: KeyGen, Setup, Challenge, Prove, Verify and Deduplication (Yuan and Yu(2013)).

POSD Scheme Details: Let $H(\cdot)$ be a hash function, G be a multiplicative cyclic group of prime order q, g be a generator of G, and $u \in G$. F' is the erasure coded file divided into n blocks with s elements: $\{m_{ij}\}_{1\leq i\leq n, 0\leq j\leq s-1}$. $f_{\vec{a}(x)}$ is a polynomial with coefficient vector $\vec{a} = (a_0, a_1, \cdots, a_{s-1})$ (Yuan and Yu(2013)).

- KeyGen: TA selects random number $\alpha \to \mathbb{Z}_q^*$ which is the master key, and generates public key of the system $\{g^{\alpha^j}\}_{j=0}^{s+1}$. The data owner generates a signing key pair (spk, ssk). The owner also selects random number $\epsilon \to \mathbb{Z}_q^*$, and computes $k = g^{\epsilon}$, $v = g^{\alpha\epsilon}$. $PK = \{q, k, v, spk, u, \{g^{\alpha^j}\}_{j=0}^{s+1}\}$, $SK = \{\epsilon, ssk\}$, and $MK = \{\alpha\}$.

- Setup: Given F', the data owner selects randomly a file name $name \in \mathbb{Z}_q^*$, and generates a file tag $\tau = name \parallel n \parallel Sign_{ssk}(name \parallel n)$. For each block m_i, the owner generates an authentication tag as $\sigma_i = (u^{H(name\|i)} \times \prod_{j=0}^{s-1} g^{m_{ij}\alpha^{j+2}})^{\epsilon} = (u^{H(name\|i)} \times g^{f_{\vec{\beta_i}}(\alpha)})^{\epsilon}$, where $\vec{\beta_i} = \{0, 0, m_{i,0}, m_{i,1}, \cdots, m_{i,s-1}\}$. The data owner sends $(F', \tau, \{\sigma_i\})$ to the cloud server.

- Challenge: A user retrieves the file tag τ and verifies the signature. If it is true, he recovers $name$ and n. The user chooses k–elements subset K of $[1, n]$, and two random numbers $r \to \mathbb{Z}_q^*$. Then, he sends $chal = \{K, r\}$ to the server.

- Prove: The cloud generates $\{p_i = r^i \mod q\}, i \in K$, and $y = f_{\vec{A}}(r)$, where $\vec{A} = \{0, 0, \sum_{i\in K} p_i m_{i,0}, \cdots, \sum_{i\in K} p_i m_{i,s-1}\}$. The cloud generates $\psi = \prod_{j=2}^{s+1}(g^{\alpha^j})^{w_j} = g^{f_w(\alpha)}$, computes $\sigma = \prod_{i\in K}\sigma_i^{p_i}$, and sends them as $Prf = \{\sigma, \psi, y\}$ to the user.

- Verify: The user computes $\vartheta = \sum_{i\in K} p_i H(name \parallel i)$ and $\eta = u^{\vartheta}$. He verifies via $e(\eta, k) \times e(\psi, vk^{-r}) = e(\sigma, g) \times e(k^{-y}, g)$.

- Deduplication: A user claims that he has a file F' and wants to send it to the cloud that is already sent by another user. The cloud chooses d–elements subset D of $[1, n]$, and sends D to the user who sends back the corresponding blocks. The cloud computes $\sigma' = \prod_{i\in D}\sigma_i$, $\eta' = \prod_{i\in D} u^{H(name\|i)}$, and $\psi' = e(\prod_{j=2}^{s+1}(g^{\alpha^j})^{B_j}, k) = e(g^{f_{\vec{B}}(\alpha)}, k)$ where $\vec{B} = (0, 0, \sum_{i\in D} m_{i,0}, \cdots, \sum_{i\in D} m_{i,s-1})$. Then the server verifies via $e(\eta', k) \times \psi' = e(\sigma', g)$.

- Auditing after Deduplication: If the file F' is owned by multiple owners $owner_w$ where w is the number of owners and the original owner is $owner_0$, the cloud has to aggregate the authentication tags in order to audit the file (Yuan and Yu(2013)).

  - A new owner $owner_w$ generates his public and secret key as $PK_w = \{q, k_w, v_w, spk_w, u, \{g^{\alpha^j}\}_{j=0}^{s+1}\}$, $SK_w = \{\epsilon_w, ssk_w\}$. He also generates the file tag as $\tau_w = name \parallel n \parallel Sign_{ssk_w}(name \parallel n)$ and authentication tag for each block as $\sigma_{wi} = (u^{H(name\|i)} \times \prod_{j=0}^{s-1} g^{m_{ij}\alpha^{j+2}})^{\epsilon_w} = (u^{H(name\|i)} \times g^{f_{\vec{\beta_i}}(\alpha)})^{\epsilon_w}$, where $\vec{\beta_i} = \{0, 0, m_{i,0}, m_{i,1}, \cdots, m_{i,s-1}\}$. The cloud aggregates tags for each blocks from multiple owners as $\sigma_i = \prod_{w=0}^{W}\sigma_{wi}$. When $owner_t$ wants to check the integrity of F', he sends $chal = \{K, r\}$ to the cloud who computes $\sigma = \prod^{i\in K}\sigma_i$, $k' = \prod^{k_w}, w \in W/t$, and $v' \prod v_w, w \in W/t$. The server sends $Prf = \{\sigma, \psi, y, k', v'\}$ to the user who verifies it via $e(\eta, k) \times e(\psi, v' v_t k^{-r}) = e(\sigma, g) \times e(k^{-y}, g)$, where $k = k' k_t$.

Features of the scheme: The scheme allows public auditing with proof of ownership. The scheme has constant communication and computational cost on the user side. It also allows auditing after deduplication, and batch auditing (Yuan and Yu(2013)).

Drawbacks of the scheme: The scheme does not support privacy-preservation property. It also does not consider deduplication on block-level.

2.3 Aggregate Signatures vs. Multisignatures

According to (Boneh et al.(2003)Boneh, Gentry, Lynn, and Shacham), aggregate signature is a a digital signature that can be aggregated. Suppose we have n signatures on n different messages from n different users, these signatures can be aggregated into single signatures. Multisignature is related to the aggregated signature, however all the users sign the *same* message and convince a verifier that each user signed the message.
We give a briefly comparative survey of various aggregate signatures and multisignatures, see Table2.1, in order to choose what the suitable one that fits our needs.

2.3.1 GDH Multisignature Scheme

A multisignature scheme enables subgroup of users to sign a document in which a verifier can check if each user in the group participate in signing. Unlike aggregate signature, multisignature enables us to aggregate signatures of the *same* message (Boldyreva(2003)). The proposed scheme in (Boldyreva(2003)) enables us to decide the composition of the subgroup of users after the aggregation of the signatures. It also has one round of communication between the users. The signing protocol of the scheme is non-interactive (Boldyreva(2003)). Theses features fulfill our requirements to build our proposed scheme.

Definition 12 (MGS scheme)**.** The scheme consists of three algorithms: MK, MS, and MV (Boldyreva(2003)).

MGS Scheme Details: Let G be a GDH group and I be the global information that consists of (g is a generator of G, $p = |G|$, and $H : \{0,1\}^* \to G^*$ is a hash function (random oracle)). Let $U = \{U_1, \cdots, U_n\}$ be a group of users.

MK: Each users U_i selects $x_i \to \mathbb{Z}_p^*$ and computes $y = g^{x_i}$. $sk_i = x_i$ and $pk_i = (p, g, H, y)$

MS: Any user $U_j \in U$ wants to participate in signing M, computes and broadcasts $\sigma_j = H(M)^{x_j}$. Let $L = \{U_{i1}, \cdots, U_{il}\}$ be a subgroup of users who contributed to the signing. Let $J = \{i1, \cdots, il\}$be the indices of the users. The designated signer D (implemented by any player) who knows the signer of each signature computes $\sigma = \prod_{j\in J}(\sigma_j)$ and outputs $T = (M, L, \sigma)$

MV: The verifier takes T and the public keys of L, and computes $pk_L = \prod_{j\in J}(pk_j) = \prod_{j\in J}(g^{x_j})$ and outputs $V_{DDH}(g, pk_L, H(M), \sigma)$.

Table 2.1: Comparative survey of various aggregate signature and multisignature schemes

	Scheme name	Interactive scheme	The order of the signer is required	Security Model	Assumption	Notes
1	BGLS (Boneh et al.(2003)Boneh, Gentry, Lynn, and Shacham)	No	No	Random Oracle	The scheme is based on BLS and bilinear map. It works in any group where the Decision Diffie-Hellman problem (DDH) is easy, but the Computational Diffie-Hellman problem (CDH) is hard.	The scheme must be on Distinct Messages.
2	MGS (Boldyreva(2003))	No	No	Random Oracle	The scheme is based on BLS	The subset of signers should not be known in advance. It requers one round of communication for the scheme generation protocol. The signature length and verification time is independent of the size of the subgroup and is almost the same as for the base signature scheme.
3	LOSSW (Lu et al.(2006)Lu, Ostrovsky, Sahai, Shacham, and Waters)	No	No	Standard	The scheme is based on Waters Signature scheme (Waters(2005))	
4	OMS (Boldyreva et al.(2007)Boldyreva, Gentry, O'Neill, and Yum)	No	Yes	Random Oracle	The scheme is based on MGS and bilinear map. The CDH is hard relative to its associated bilinear-group generator G.	The scheme has constant size (constant-size keys)
5	AS-CPO (Wen and Ma(2008))	yes	Yes	Random Oracle	bilinear map, computational Diffie-Hellman(CDH) is hard. Decision Diffie-Hellman (DDH) is easy.	The scheme requires only two pairing operations in verification which is independent of the number of signers. The scheme requires no MapToPoint hash function which requires more computation overhead than ordinary cryptography hash function.

Chapter 3

Design and Implementation

In this chapter, we describe in details our proposed schemes that achieve both data integrity and storage efficiency. We present an extension to support batch auditing and user revocation.

3.1 Problem Statement

Figure 3.1 depicts a typical setting for our proposed protocol. X-enterprise users want to send their data files to the cloud server. Later, they can check the correctness of their stored data by allowing the TPA to verify the data integrity. But before sending the data to the cloud, the users calculate the signature of the data for the integrity verifications. Then, they send them to the mediator. The latter calculates the hash value of the data and compares them to identify the duplicated data. Thus, the mediator has two jobs: first, eliminating the duplicated blocks before sending the data to the cloud, so we reduce the amount of the uploaded data. Hence, that minimizes the bandwidth between the enterprise and the cloud server. Second, aggregating the signature of the duplicated data. Later, the TPA can check the integrity of the stored data on behalf of the users or the mediator. The users can contact the cloud for downloading their stored files. In addition, they can directly contact the TPA to issue auditing processes.

3.1.1 System Model

The architecture of the proposed scheme includes a cloud storage service with four entities.

- *Cloud User (U):* who wants to store his data in cloud storage server.
- *Cloud Server (CS):* is a data storage service with huge storage capacity and computational resources provided by a Cloud Service Provider (CSP).
- *Third Party Auditor (TPA):* who can verify the integrity of U's data upon request and on their behalf.

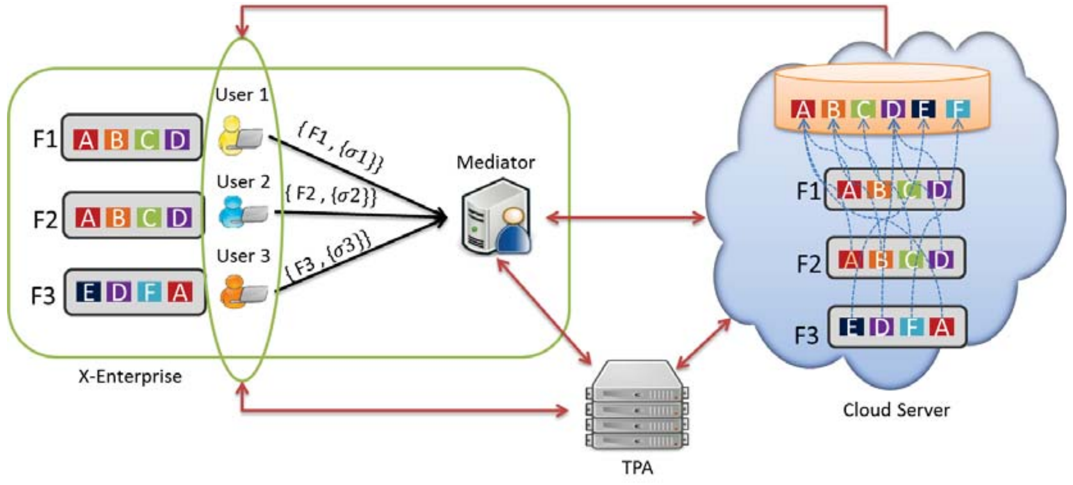


Figure 3.1: The architecture of the proposed scheme

- *Mediator:* who performs block-level deduplication on users' data before they are sent to the CSP, and also computes an aggregate signature for the duplicated blocks instead of sending multiple signatures with each block.

3.1.2 Threat Model

- Mediator: is trusted and allowed to see the content of the blocks and their signatures, but it is prohibited from knowing the private keys of the users, so it cannot generate a valid signature on behalf of any user.
- TPA: is semi-trusted and allowed to check the integrity of the blocks on behalf of the users/mediator, but it is prohibited from seeing the content of the blocks.
- CS: is semi-trusted and allowed to see the content of the blocks and their signatures, but it is required to follow the steps needed for the auditing process.

Protocol Overview:

Users have files that are to be sent to the cloud. Each user divides his file into n blocks and computes the signature of each block for the integrity verification using his private and public keys. Then, the users send their files (blocks) with the signatures to the mediator. There is no interaction between the users during the signing process. The mediator calculates the hash value of each block and compares them to identify duplicated blocks. Then, it calculates the aggregated signature of the duplicated blocks utilizing the multisignature scheme of (Boldyreva(2003)). So, Instead of sending one (identical) block with multiple signatures from multiple users, the mediator sends the aggregated signature with the block and deletes the local copy. The metadata of the deduplication process is stored locally and in the cloud.

The TPA checks the data integrity on behalf of the users or the mediator upon request. To do so, it sends a challenge message to the cloud server to make sure that the cloud has retained the data properly. To generate that challenge, the TPA picks random c-element subset of set $[1, n]$. For each element, the TPA chooses a random value. The challenge message specifies the positions of the blocks required to be checked. The cloud server generates a proof of data storage correctness and sends it to the TPA who verifies the proof using a verification equation.

3.2 Design Goals

1. Public auditability: enables the TPA to check the integrity of the data stored in the cloud on behalf of the user/mediator.
2. Storage correctness: ensures that the cloud cannot cheat and pass the auditing process without having stored the data intact.
3. Client-side deduplication at block-level: allows the Mediator to eliminate the duplicated blocks before sending the data to the cloud.
4. Privacy-preserving: ensures that the TPA cannot learn anything about content of the stored data during the auditing process.
5. Lightweight: provides the scheme with low communication and computational costs.
6. Batch auditing: allows the TPA to perform multiple auditing tasks at the same time from different users.

We design schemes that may match these goals. For example, the first scheme provides public auditabiltiy, storage correctness, and client-side deduplication. The second scheme provides goal 1 to goal 5. The last scheme involves all the goals.

3.3 Scheme Details

3.3.1 Public Auditing in Cloud with Data Deduplication Scheme

- Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map. Let g be a generator of $\mathbb{G}_2$. $H(\cdot)$ is a secure map-to-point hash function $H{:}\{0,1\}^* \rightarrow \mathbb{G}_1$.
- We define the users as $U = \{u_1, \cdots, u_m\}$. User u_i has files F_i. The blocks are denoted by $m_{i,j}$ where $1 \leq i \leq k$ and $1 \leq j \leq n$, for some k and n.
- Setup phase:

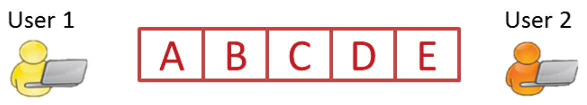


Figure 3.2: Case 1: Two users have the same file

- KeyGen: Each user u_i selects a random $x_i \to \mathbb{Z}_p$, computes $y_i = g^{x_i}$, and selects a random $u_i \to \mathbb{G}_1$. The user u_i's public and private keys (sk_i, pk_i) are set to be $(x_i, (y_i, g, u_i))$
- SigGen: Each user computes the signature of each block of his file as $\sigma_{i,j} = (H(m_{i,j}) \times u_i^{m_{i,j}})^{x_i} \in \mathbb{G}_1$. Then each user u_i sends $(F_i = \{m_{i,j}\}, \{\sigma_{i,j}\})$ to the Mediator.
- Dedup: The deduplication is done by the Mediator at the block-level. The mediator calculates the hash value of each block and compares them to identify the duplicated ones. Then, he calculates the aggregated signatures of the duplicated blocks as $\sigma_j = \prod \sigma_{i,j}$. After that, he sends $(F = \{m_j\}, \{\sigma_j\}, L)$ to the cloud, where $L=$ the subgroup of users.

• Audit phase:

- The TPA can check the integrity of files/blocks on behalf of the users/mediator by sending *chal* $= \{(s, v_s)\}$ to the cloud server $s \in I = \{s_1, \cdots, s_c\}$ of set of blocks $[1, n]$.
- GenProof: the cloud server computes $\mu = \sum_{s\in I} v_s m_s$, and $\sigma = \prod_{s\in I} \sigma_s^{v_s}$. Then, he sends $\{\mu, \sigma, \{H(m_s)\}\}$ to the TPA.
- VerifyProof: The TPA verifies $\{\mu, \sigma, \{H(m_s)\}\}$ via:

$$e(\sigma, g) \overset{?}{=} \prod_{i\in L} e((\prod_{s\in I}(H(m_s)^{v_s}) \times u_i^{\mu}, y_i) \tag{3.1}$$

Note: TPA should not know (m_s), thus he couldn't calculate the hash value of it. So, the cloud server has to send $\{\{\mu\}, \sigma, \{H(m_s)\}\}$ to the TPA.

The following steps show the correctness of the equation:

$$\begin{aligned} e(\sigma, g) &= e(\prod_{s\in I}(\prod_{i\in L}(H(m_s) \times u_i^{m_s})^{x_i})^{v_s}, g) \\ &= \prod_{i\in L} e(\prod_{s\in I}(H(m_s) \times u_i^{m_s})^{v_s}, g^{x_i}) \\ &= \prod_{i\in L} e((\prod_{s\in I}(H(m_s)^{v_s}) \times u_i^{\mu}, y_i) \end{aligned}$$

3.3.2 Privacy-Preserving Public Auditing in Cloud with Data Deduplication Scheme

For the sake of clarity, we begin with the case where two users have identical files. Then, we extend this case to where the two users have identical blocks of (possibly different) files.

Case 1: Two users have the same file

Suppose we have two users who have the same file (same blocks) as described in Figure 3.2. With no interaction, they send their files and signatures to the mediator who performs a block-level deduplication on the the files and aggregates the signatures of the duplicated blocks. Then, the mediator sends the files and the signatures after the deduplication process to the cloud. Assuming that User 1 asks the TPA to check the correctness of his file which is owned by another User 2, we have the following:

- Let $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ be multiplicative cyclic groups of prime order p, and $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ be a bilinear map. Let g be a generator of $\mathbb{G}_2$. $H(\cdot)$ is a secure map-to-point hash function H:$\{0,1\}^* \rightarrow \mathbb{G}_1$, and $h(\cdot)$ is a hash function h: $\mathbb{G}_T \rightarrow \mathbb{Z}_p$. We define the users as $U = \{u_1, \cdots, u_m\}$. User u_i has files F_i. The blocks are denoted by $m_{i,j}$ where $1 \leq i \leq k$ and $1 \leq j \leq n$, for some k and n.

- Setup phase:

  - KeyGen: Each user u_i selects a random $x_i \rightarrow \mathbb{Z}_p$, computes $y_i = g^{x_i}$, and selects a random $u_i \rightarrow \mathbb{G}_1$. The user u_i's public and private keys (sk_i, pk_i) are set to be $(x_i, (y_i, g, u_i, e(u_i, y_i)))$.

  - SigGen: Each user computes the signature of each block of his file as $\sigma_{i,j} = (H(m_{i,j}) \times u_i^{m_{i,j}})^{x_i} \in \mathbb{G}_1$. Then, each user u_i sends $(F_i = \{m_{i,j}\}, \{\sigma_{i,j}\})$ to the Mediator.

  - Dedup: The deduplication is done by the Mediator at the block-level. The mediator calculates the hash value of each block and compares them to identify the duplicated ones. Then, it calculates the aggregated signatures of the duplicated blocks as $\sigma_j = \prod \sigma_{i,j}$. For example, $\sigma_A = \prod_{i \in L}(H(m_A) \times u_i^{m_A})^{x_i}$ and so on to the end of the file. After that, the mediator sends $(F = \{m_j\}, \{\sigma_j\}, L)$ to the cloud, where L= the subgroup of users.

- Audit phase:

  - TPA can check the integrity of files/blocks on behalf of the users/mediator. TPA sends $chal = \{(s, v_s)\}$ to the cloud server $s \in I = \{s_1, \cdots, s_c\}$ for set of blocks $[1, n]$.

  - GenProof: The cloud server selects $r \in \mathbb{Z}_p$, and computes $R_i = e(u_i, y_i)^r \in \mathbb{G}_T$. CS also computes $\mathcal{R} = R_1 \times ... \times R_m$, $\mathcal{L} = y_1 \| \cdots \| y_m$, and $\gamma = h(\mathcal{R} \| \mathcal{L})$. CS computes as well $\mu' = \sum_{s \in I} v_s m_s$, $\mu = r + \gamma\mu'$, and $\sigma = \prod_{s \in I} \sigma_s^{v_s}$. Then, CS sends $\{\mu, \sigma, \{H(m_s)\}, \mathcal{R}, L\}$ to the TPA.

  - VerifyProof: The TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, and verifies $\{\mu, \sigma, \{H(m_s)\}, \mathcal{R}, L\}$ via:

$$\mathcal{R} \times e(\sigma^\gamma, g) \overset{?}{=} \prod_{i \in L} e((\prod_{s \in I}(H(m_s)^{v_s})^\gamma \times u_i^\mu, y_i) \tag{3.2}$$

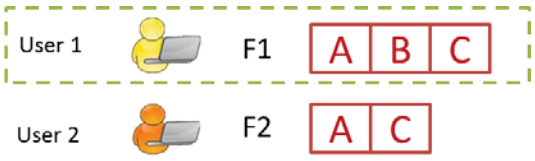


Figure 3.3: Case 2: Two users have some identical blocks

The following steps show the correctness of the equation:

$$\begin{aligned}
\mathcal{R} \times e(\sigma^{\gamma}, g) &= \prod_{i\in L} e(u_i, y_i)^r \times e((\prod_{s\in I} \sigma_s^{v_s})^{\gamma}, g) \\
&= \prod_{i\in L} e(u_i, y_i)^r \times e(\prod_{s\in I}(\prod_{i\in L}((H(m_s) \times u_i^{m_s})^{x_i})^{v_s})^{\gamma}, g) \\
&= \prod_{i\in L} e(u_i, y_i)^r \times \prod_{i\in L} e(\prod_{s\in I}(H(m_s)^{v_s} \times u_i^{v_s m_s})^{\gamma}, y_i) \\
&= \prod_{i\in L} e(\prod_{s\in I}(H(m_s)^{v_s})^{\gamma} \times u_i^{\mu' \gamma + r}, y_i) \\
&= \prod_{i\in L} e((\prod_{s\in I}(H(m_s)^{v_s})^{\gamma} \times u_i^{\mu}, y_i)
\end{aligned}$$

Case 2: Two users have some identical blocks

Suppose we have two users who have some identical blocks as described in Figure 3.3. With no interaction, they send their files and signatures to the mediator who performs the block-level deduplication on the the files and aggregates the signatures of the duplicated blocks. Then, the mediator sends the files and the signatures after the deduplication process to the cloud. Assuming that User 1 asks the TPA to check the correctness of his file $F_1 = \{A, B, C\}$ which some of his blocks are owned by another User 2, $F_2 = \{A, C\}$, we have the following:

- Setup phase: KeyGen, SigGen, and Dedup as in the previous case. The mediator calculates the aggregated signatures of the duplicated blocks as $\sigma_j = \prod \sigma_{i,j}$. For example, $\sigma_A = \prod_{i\in L}(H(m_A) \times u_i^{m_A})^{x_i}$, $\sigma_B = (H(m_B) \times u_1^{m_B})^{x_1}$ since this block is owned only by User 1, and $\sigma_C = \prod_{i\in L}(H(m_C) \times u_i^{m_C})^{x_i}$.

- Audit phase:

  - TPA can check the integrity of files/blocks on behalf of the users/mediator by sending $chal = \{(s, v_s)\}$ to the cloud server $s \in I = \{s_1, \cdots, s_c\}$ for the set of blocks $[1, n]$.

  - GenProof: The cloud server selects $r \in \mathbb{Z}_p$, and computes $R_i = e(u_i, y_i)^r \in \mathbb{G}_T$. CS also computes $\mathcal{R} = R_1 \times \cdots \times R_m$, $\mathcal{L} = y_1 \| \cdots \| y_m$, and $\gamma = h(\mathcal{R} \| \mathcal{L})$. CS computes as well $\mu'_i = \sum_{s\in I} v_s m_s$, $\mu_i = r + \gamma \mu'_i$ for each user u_i, and $\sigma = \prod_{s\in I} \sigma_s^{v_s}$. Then CS sends $\{\{\mu_i\}, \sigma, \{H(m_s)\}, \mathcal{R}, L\}$ to the TPA.

– VerifyProof: The TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, and verifies $\{\{\mu_i\}, \sigma, \{H(m_s)\}, \mathcal{R}, L\}$ via:

$$\mathcal{R} \times e(\sigma^{\gamma}, g) \overset{?}{=} \prod_{i \in L} e((\prod_{s \in I} (H(m_s)^{v_s})^{\gamma} \times u_i^{\mu_i}, y_i) \tag{3.3}$$

The following steps show the correctness of the equation:

$$\begin{aligned}
\mathcal{R} \times e(\sigma^{\gamma}, g) &= \prod_{i \in L} e(u_i, y_i)^r \times e((\prod_{s \in I} \sigma_s^{v_s})^{\gamma}, g) \\
&= \prod_{i \in L} e(u_i, y_i)^r \times e((\prod_{s \in I} (H(m_s) \times u_i^{m_s})^{x_i})^{v_s})^{\gamma}, g) \\
&= \prod_{i \in L} e(u_i, y_i)^r \times \prod_{i \in L} e(\prod_{s \in I} (H(m_s)^{v_s})^{\gamma}) \times u_i^{v_s m_s \gamma}, y_i) \\
&= \prod_{i \in L} e(\prod_{s \in I} (H(m_s)^{v_s})^{\gamma} \times u_i^{\mu_i' \gamma + r}, y_i) \\
&= \prod_{i \in L} e((\prod_{s \in I} (H(m_s)^{v_s})^{\gamma} \times u_i^{\mu_i}, y_i)
\end{aligned}$$

3.4 Support for Batch Auditing

The TPA can handle numerous auditing tasks from different users. It may be inefficient to treat them as an individual tasks rather than batch them together and audit at the same time. Suppose we have K auditing tasks on K distinct files from different users. To achieve the batch auditing, the K verification equations are aggregated (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

3.4.1 Batch Auditing of Case 1: Two users have the same file

Suppose there are L users who want to check K distinct files by delegating K auditing tasks. Each auditing task consists of two users who have the same file as described in Figure 3.4. User 1 delegates auditing Task 1 ($k = 1$) on file $F1$ which is owned also by User 2, User 3 delegates auditing Task 2 ($k = 2$) on file $F2$ which is owned also by User 4, and User 5 delegates auditing Task 3 ($k = 3$) on file $F3$ which is owned also by User 6. Each auditing task has two users, so $L_k = 2$ for each task k. In General, each user k has a file $F_k = \{m_{k,1}, \cdots, m_{k,n}\}$, which is also owned by another user $u_{k,i}$, where $k \in \{1, \cdots, K\}$ and $i \in \{1, \cdots, L\}$.

Setup phase: Each user $u_{k,i}$ selects a random $x_{k,i} \to \mathbb{Z}_p$, computes $y_{k,i} = g^{x_{k,i}}$, and selects a random $u_{k,i} \to \mathbb{G}_1$. Denote the secret key and the public key of each user $u_{k,i}$ is $(sk_{k,i}, pk_{k,i}) = (x_{k,i}, (y_{k,i}, g, u_{k,i}, e(u_{k,i}, y_{k,i})))$. Then, each user computes the signature of each block of his file as $\sigma_{k,j} = (H(m_{k,j}) \times u_{k,i}^{m_{k,j}})^{x_{k,i}} \in \mathbb{G}_1$. Then, he sends $(F_{k,i} = \{m_{k,j}\}, \{\sigma_{k,j}\})$ to the Mediator. The Mediator calculates the hash value of each block and compares them to identify the duplicates. Then, it calculates the aggregated signatures of the duplicated blocks as $\sigma_{k,j} = \prod \sigma_{k,i,j}$. After that, the mediator sends $(F_k = \{m_{k,j}\}, \{\sigma_{k,j}\}, L_k)$ to the cloud, where $L=$ the subgroup of users.

Audit phase: TPA sends the audit challenge $chal = \{(s, v_s)\}$ to the cloud server for auditing data files

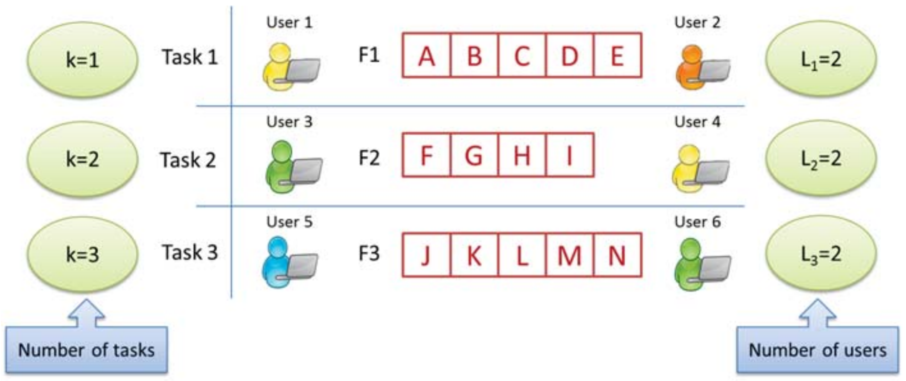


Figure 3.4: Batch Auditing of Case 1

of all K users. Upon receiving *chal*, cloud server selects $r \in \mathbb{Z}_p$ and computes $R_{k,i} = e(u_{k,i}, y_{k,i})^r \in \mathbb{G}_T$. Cloud server also computes $\mathcal{R} = R_{1,1} \times \cdots \times R_{K,m}$,and $\mathcal{L} = y_{1,1}\| \cdots \|y_{K,m}$. Then, he computes $\gamma = h(\mathcal{R}\|\mathcal{L})$. The cloud generates the proof as follows: $\mu'_k = \sum_{s\in I} v_s m_{k,s}$, $\mu_k = r + \gamma\mu'_k$, and $\sigma_k = \prod_{s\in I} \sigma_{k,s}^{v_s}$. Then, the cloud sends $\{\{\mu_k\}, \{\sigma_k\}, \{H(m_{k,s})\}, \mathcal{R}, \{L_k\}\}$ to the TPA. To verify the response, the TPA computes $\gamma = h(\mathcal{R}\|\mathcal{L})$. Next, he verifies $\{\{\mu_k\}, \{\sigma_k\}, \{H(m_{k,s})\}, \mathcal{R}, \{L_k\}\}$ via:

$$\mathcal{R} \times e(\prod_{k=1}^{K} \sigma_k^{\gamma}, g) \overset{?}{=} \prod_{k=1}^{K}(\prod_{i\in L} e((\prod_{s\in I}(H(m_{k,s})^{v_s})^{\gamma} \times u_{k,i}^{\mu_k}, y_{k,i})) \tag{3.4}$$

The following steps show the correctness of the equation:

$$\begin{aligned}
\mathcal{R} \times e(\prod_{k=1}^{K} \sigma_k^{\gamma}, g) &= R_{k,i} \times \cdots \times R_{K,m} \times \prod_{k=1}^{K} e(\sigma_k^{\gamma}, g) \\
&= \prod_{k=1}^{K}(\prod_{i\in L} R_{k,i} \times e(\sigma_k^{\gamma}, g)) \\
&= \prod_{k=1}^{K}(\prod_{i\in L} e(u_{k,i}, y_{k,i})^r \times e(\sigma_k^{\gamma}, g)) \\
&= \prod_{k=1}^{K}(\prod_{i\in L} e((\prod_{s\in I}(H(m_{k,s})^{v_s})^{\gamma} \times u_{k,i}^{\mu_k}, y_{k,i}))
\end{aligned}$$

3.4.2 Batch Auditing of Case 2: Two users have some identical blocks

Suppose there are L users who want to check K distinct files by delegating K auditing tasks. Each auditing task consists of two users who share some blocks as described in Figure 3.5. User 1 delegates auditing Task 1 on file $F1$ which has some identical blocks from $F4$ which is owned by User 4, User 2 delegates auditing Task 2 on file $F2$ which has some identical blocks from $F5$ which is owned by User 5, and User 3 delegates auditing Task 3 on file $F3$ which has some identical blocks from $F6$ which is owned by User 6. Each auditing task has two users, so $L_k = 2$ for each task k. In General, each user k

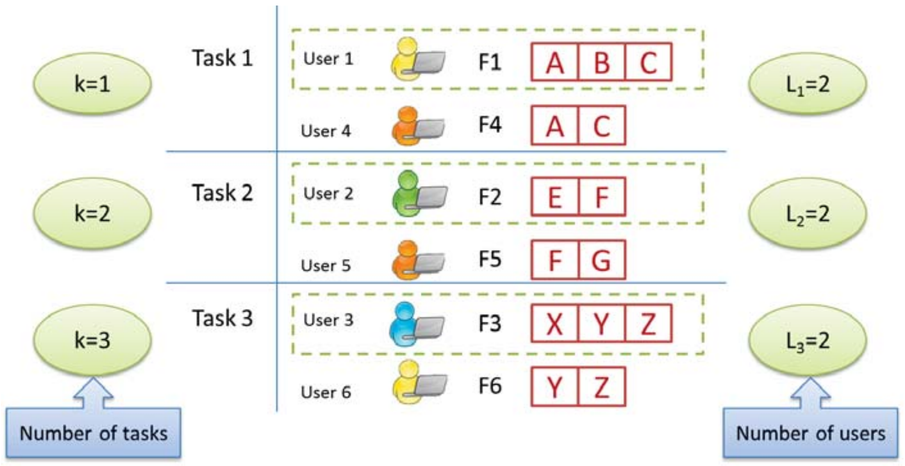


Figure 3.5: Batch Auditing of Case 2

has a file $F_k = \{m_{k,1}, \cdots, m_{k,n}\}$, and some of these blocks are also owned by another user $u_{k,i}$, where $k \in \{1, \cdots, K\}$ and $i \in \{1, \cdots, L\}$.

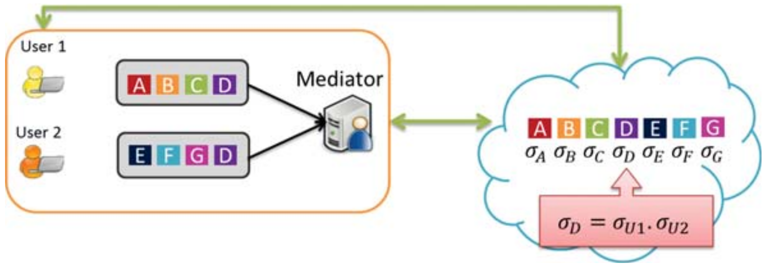
Setup phase: As in the previous one.

Audit phase: TPA sends the audit challenge $chal = \{(s, v_s)\}$ to the cloud server for auditing data files of all K users. Upon receiving $chal$, cloud server selects $r \in \mathbb{Z}_p$ and computes $R_{k,i} = e(u_{k,i}, y_{k,i})^r \in \mathbb{G}_T$. Cloud server also computes $\mathcal{R} = R_{1,1} \times \cdots \times R_{K,m}$, and $\mathcal{L} = y_{1,1} \| \cdots \| y_{K,m}$. Then, he computes $\gamma = h(\mathcal{R} \| \mathcal{L})$. The cloud generates the proof as follows: computes the linear combination of sampled blocks for each user in L in each task in K as $\mu'_{k,i} = \sum_{s \in I} v_s m_{k,s}$, $\mu_{k,i} = r + \gamma \mu'_{k,i}$. Then, he computes $\sigma_k = \prod_{s \in I} \sigma_{k,s}^{v_s}$ for each task in K. Next, the cloud sends $\{\{\mu_{k,i}\}, \{\sigma_k\}, \{H(m_{k,s})\}, \mathcal{R}, \{L_k\}\}$ to the TPA. To verify the response, the TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$. Then, he verifies $\{\{\mu_{k,i}\}, \{\sigma_k\}, \{H(m_{k,s})\}, \mathcal{R}, \{L_k\}\}$ via:

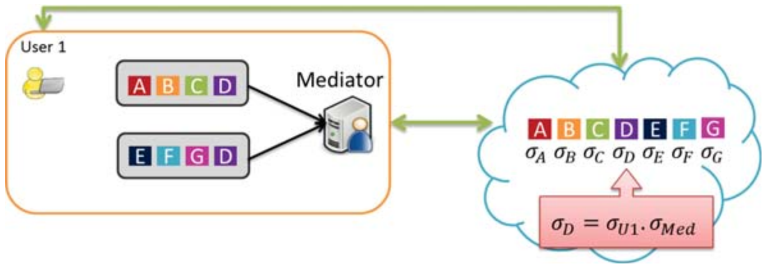
$$\mathcal{R} \times e(\prod_{k=1}^{K} \sigma_k^{\gamma}, g) \overset{?}{=} \prod_{k=1}^{K} (\prod_{i \in L} e((\prod_{s \in I} (H(m_{k,s})^{v_s})^{\gamma} \times u_{k,i}^{\mu_{k,i}}, y_{k,i})) \tag{3.5}$$

The following steps show the correctness of the equation:

$$\begin{aligned}
\mathcal{R} \times e(\prod_{k=1}^{K} \sigma_k^{\gamma}, g) &= R_{k,i} \times \cdots \times R_{K,m} \times \prod_{k=1}^{K} e(\sigma_k^{\gamma}, g) \\
&= \prod_{k=1}^{K} (\prod_{i \in L} R_{k,i} \times e(\sigma_k^{\gamma}, g)) \\
&= \prod_{k=1}^{K} (\prod_{i \in L} e(u_{k,i}, y_{k,i})^r \times e(\sigma_k^{\gamma}, g)) \\
&= \prod_{k=1}^{K} (\prod_{i \in L} e((\prod_{s \in I} (H(m_{k,s})^{v_s})^{\gamma} \times u_{k,i}^{\mu_{k,i}}, y_{k,i}))
\end{aligned}$$


User 1
User 2
A B C D
E F G D
Mediator
A B C D E F G
σ_A σ_B σ_C σ_D σ_E σ_F σ_G
$\sigma_D = \sigma_{U1}.\sigma_{U2}$

(a) Before User 2 is revoked, block D is signed by User 2


User 1
A B C D
E F G D
Mediator
A B C D E F G
σ_A σ_B σ_C σ_D σ_E σ_F σ_G
$\sigma_D = \sigma_{U1}.\sigma_{Med}$

(b) After User 2 is revoked, block D is signed by Mediator

Figure 3.6: User Revocation

3.5 User Revocation

Inside the Enterprise, the users work as a group. If any user wants to leave the group (or enterprise), the blocks that signed by the revoked user need to be re-signed by another user which is in our architecture the mediator. The simple solution is the mediator has to compute a signature of the block under his private key. Then, he computes a new aggregate signature of the block and sends it to the cloud as an updated version of the stored one as $\sigma_{new} = \sigma_{old} \times \sigma_{Med}/\sigma_{U_2}$. Then, he has to send new L to the cloud as an update. This way has a major drawback which is the mediator has to download the block from the cloud in order to compute the signature. By utilizing the proxy re-signature proposed by (Wang et al.(2013b)Wang, Li, and Li), we allow the cloud to convert signatures of the blocks of the revoked user into Mediator's signature while preserving the privacy of the secret keys of both the revoked user and the mediator. See Figure 3.6, the block D is signed by User 2 U_2. After the user revocation, the same block is signed by Mediator Med. Since the cloud has only the aggregated signature of block D which consists of the signatures of User 1 U_1 and of User 2 U_2, Proxy re-signature scheme cannot be directly adopted. The solution is: First, the cloud has to remove the signature of the revoked user from the aggregated signature he has as $\sigma_{D_{temp}} = \sigma_D/\sigma_{D_{U2}}$. Second, using the proxy re-signature scheme, the cloud converts the signature of the block into the mediator signature. Finally, he aggregates the result with the new signature of the block as $\sigma_{D_{new}} = \sigma_{D_{temp}} \times \sigma_{D_{Med}}$.

Proxy Re-signature Scheme:

- In the Setup phase: the mediator as well generates his public and private keys. (sk_{Med}, pk_{Med}) are set to be $(x_{Med}, (y_{Med}, g, u_{Med}, e(u_{Med}, y_{Med})))$.

- The mediator sends the signature of the block of the revoked user to the cloud as $\sigma_{D_{U2}}$.
- The cloud generates a re-signing key as follows: he selects a random value $r \in \mathbb{Z}_p$ and sends it to the user U_2 who computes and sends r/x_{U_2} to the mediator. The mediator computes and sends $r \times x_{Med}/x_{U_2}$ to the cloud. Finally, the cloud recovers the re-signing key x_{Med}/x_{U_2}.
- The cloud removes the signature of user U_2 from the aggregated signature ha has as

  $$\sigma_{D_{temp}} = \sigma_D/\sigma_{D_{U2}}$$

  Next, he converts the signature of U_2 into the signature of the mediator as

  $$\sigma_{D_{Med}} = (H(m_D) \times u_2^{m_D} \times (u_{Med}/u_2)^{m_D})^{x_{Med}/x_{U_2}}$$

  $$\sigma_{D_{Med}} = (H(m_D) \times u_{Med}^{m_D})^{x_{Med}}$$

  Finally, he computes the new aggregated signature of block D as

  $$\sigma_{D_{new}} = \sigma_{D_{temp}} \times \sigma_{D_{Med}}$$

- The cloud updates L which is the subgroup of users who participate in signing block D.

Notes:

- Assuming that the mediator keeps a copy of all the signatures of all the blocks but not the blocks themselves.
- By this formula $\sigma_{D_{Med}} = (H(m_D) \times u_2^{m_D} \times (u_{Med}/u_2)^{m_D})^{x_{Med}/x_{U_2}}$, the cloud converts not only the private keys x but also the values u, which are public parameters, since each user has different values.

Chapter 4

Evaluation

In this chapter, we present a detailed analysis of the security and the performance of our proposed schemes. Then, we show the efficiency of the batch auditing technique.

4.1 Security Analysis

The security guarantees of our proposed schemes are based on analyzing (a) the storage correctness and (b) the privacy preservation of these schemes (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

For the storage correctness, we have to provide proofs that the cloud server can only compute the correct reply to the TPA inquiry if the integrity of the data stored on the cloud is preserved. Such computations will be based on file or block contents in cases 1 and 2. So, although duplicate blocks or files are removed in our scheme, the reliance on the content implies that the security proofs given in (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou) and (Shacham and Waters(2008)) are directly applicable. In particular, we refer the reader to section 4 of (Shacham and Waters(2008)) where different cases for the cloud server behaviour acting as an adversary are discussed.

Similar arguments for the proof of the privacy-preserving property of our schemes can be made. That is, the proof generated by the cloud server and sent to the TPA is based on the data content. Hence, if the TPA can learn about the user's data content based on the proof generated by the cloud server, it can also do the same for non-duplicated data violating the proof of theorems 2 and 3 in (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou).

4.2 Performance Analysis

The experiment was conducted using C code on a Linux system with an Intel Core i3 processor running at 2.13 GHz, 4 GB of RAM, and a 5400 RPM Toshiba 320 GB Serial ATA drive with a 8 MB buffer. We also used Pairing-Based Cryptography (PBC) library version 0.5.13, and MNT elliptic curve with base field size 159 bits and the embedding degree 6. All experimental results represent the mean of 20 trails.

Table 4.1: Notation of cryptographic operations, (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou)

$Hash^t_{\mathbb{G}}$	Hash t values into the group $\mathbb{G}$
$Add^t_{\mathbb{G}}$	t additions in group $\mathbb{G}$
$Mult^t_{\mathbb{G}}$	t multiplications in group $\mathbb{G}$
$Exp^t_{\mathbb{G}}(\ell)$	t exponentiations g^{a_i}, for $g \in \mathbb{G}, \lvert a_i \rvert = \ell$
m-$MultExp^1_{\mathbb{G}}(\ell)$	t m-term exponentiations $\prod_{i=1}^{m} g^{a_i}$
$Pair^t_{\mathbb{G}_1,\mathbb{G}_2}$	t pairings $e(u_i, g_i)$, where $u_i \in \mathbb{G}_1, g_i \in \mathbb{G}_2$
m-$MultPair^t_{\mathbb{G}_1,\mathbb{G}_2}$	t m-term pairings $\prod_{i=1}^{m} e(u_i, g_i)$

4.2.1 Cost Analysis

We estimate the computational cost in terms of basic cryptographic operations (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou) (see Table 4.1 for notations).

User Computation:

- Each user computes the signature of each block of his file as $\sigma_j = (H(m_{i,j}) \times u_i^{m_{i,j}})^{x_i} \in \mathbb{G}_1$, then sends them to the mediator.
- The corresponding computation cost is $\sigma_j = Hash^1_{\mathbb{G}_1} + Exp^1_{\mathbb{G}_1}(|p|) + Mult^1_{\mathbb{G}_1}(|p|)$, for each block
- $|\sigma_j| = |p|$ =160 bits

Mediator Computation:

- Calculating the hash value of each block to identify the duplicated ones (sha-1 = 160 bits)
- Then, aggregating the signature of the duplicated blocks as $\sigma = \prod \sigma_j$, so the corresponding computation cost is $\sigma = Mult^t_{\mathbb{G}_1}$ (t times).
- The size of the sending data (Bandwidth) depends on the rate of the deduplication.

CS Computation:

- CS computes $\mu = \sum_{s \in I} v_s m_s \in \mathbb{Z}_p$, so the corresponding computation cost is $\mu = Mult^c_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p}$
- CS computes $\sigma = \prod_{s \in I} \sigma_s^{v_s} \in \mathbb{G}_1$, so the corresponding computation cost is $\sigma = c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)$
- The size of $\{\mu, \sigma\}$ is independent of the number of sampled blocks c, that (HLA) helps achieve constant communication overhead (Bandwidth).

TPA Computaion: Verification with No Privacy-Preserving

- TPA computes $e(\sigma, g)$, so the corresponding computation cost is $Pair^1_{\mathbb{G}_1,\mathbb{G}_2}$

- TPA computes $\prod_{i \in L} e((\prod_{s \in I}(H(m_s)^{v_s}) \times u_i^{\mu}, y_i)$, so the corresponding computation cost is $L(c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)) + Exp^L_{\mathbb{G}_1} + Mult^L_{\mathbb{G}_1} + L\text{–}MultPair^1_{\mathbb{G}_1,\mathbb{G}_2}$

CS Computation: with Privacy-Preserving, Case 1

- CS selects $r \in \mathbb{Z}_p$, $|p| = 160$ bits
- CS computes for each user $R_i = e(u_i, y_i)^r \in \mathbb{G}_T$, so the corresponding computation cost is $R_i = Exp^1_{\mathbb{G}_T}(|p|)$, and the size of R_i close to 960 bits
- CS computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- CS computes $\mu' = \sum_{s \in I} v_s m_s$, so the corresponding computation cost is $\mu' = Mult^c_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p}$
- CS computes $\mu = r + \gamma\mu'$, so the corresponding computation cost is $\mu = Add^1_{\mathbb{Z}_p} + Mult^1_{\mathbb{Z}_p}$
- CS computes $\sigma = \prod_{s \in I} \sigma_s^{v_s}$, so the corresponding computation cost is $\sigma = c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)$

TPA Computation: Verification with Privacy-Preserving, Case 1

- TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- TPA computes $\mathcal{R} \times e(\sigma^{\gamma}, g)$, so the corresponding computation cost is $Exp^1_{\mathbb{G}_1}(|p|) + Pair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^1_{\mathbb{G}_T}$
- TPA computes $\prod_{i \in L} e((\prod_{s \in I}(H(m_s)^{v_s})^{\gamma} \times u_i^{\mu}, y_i)$, so the corresponding computation cost is $(c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)) + Exp^{2L}_{\mathbb{G}_1} + Mult^L_{\mathbb{G}_1} + L\text{–}MultPair^1_{\mathbb{G}_1,\mathbb{G}_2}$

CS Computation: with Privacy-Preserving, Case 2

- CS selects $r \in \mathbb{Z}_p$, $|p| = 160$ bits
- CS computes for each user $R_i = e(u_i, y_i)^r \in \mathbb{G}_T$, so the corresponding computation cost is $R_i = Exp^1_{\mathbb{G}_T}(|p|)$, and the size of R_i close to 960 bits
- CS computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- CS computes for each user: $\mu'_i = \sum_{s \in I} v_s m_s$, so the corresponding computation cost is $\mu'_i = Mult^c_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p}$
- CS computes for each user: $\mu_i = r + \gamma\mu'_i$, so the corresponding computation cost is $\mu_i = Add^1_{\mathbb{Z}_p} + Mult^1_{\mathbb{Z}_p}$
- CS computes $\sigma = \prod_{s \in I} \sigma_s^{v_s}$, so the corresponding computation cost is $\sigma = c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)$

TPA Computation: Verification with Privacy-Preserving, Case 2

- TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- TPA computes $\mathcal{R} \times e(\sigma^\gamma, g)$, so the corresponding computation cost is $Exp^1_{\mathbb{G}_1}(|p|) + Pair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^1_{\mathbb{G}_T}$
- TPA computes $\prod_{i\in L} e((\prod_{s\in I}(H(m_s)^{v_s})^\gamma \times u_i^{\mu_i}, y_i)$, so the corresponding computation cost is $L(c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)) + Exp^{2L}_{\mathbb{G}_1} + Mult^L_{\mathbb{G}_1} + L\text{–}MultPair^1_{\mathbb{G}_1,\mathbb{G}_2}$

CS Computation: Batch auditing of Case 1

- CS selects $r \in \mathbb{Z}_p$, $|p| = 160$ bits
- CS computes for each user $R_{k,i} = e(u_{k,i}, y_{k,i})^r \in \mathbb{G}_T$, so the corresponding computation cost is $R_{k,i} = Exp^1_{\mathbb{G}_T}(|p|)$, and the size of R_i close to 960 bits
- CS computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- CS computes $\mu'_k = \sum_{s\in I} v_s m_{k,s}$, so the corresponding computation cost is $\mu'_k = Mult^c_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p}$
- CS computes $\mu_k = r + \gamma\mu'_k$, so the corresponding computation cost is $\mu_k = Add^1_{\mathbb{Z}_p} + Mult^1_{\mathbb{Z}_p}$
- CS computes $\sigma_k = \prod_{s\in I} \sigma_{k,s}^{v_s}$, so the corresponding computation cost is $\sigma_k = c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)$

TPA Computation: Batch auditing of Case 1

- TPA computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- TPA computes $\mathcal{R} \times e(\prod_{k=1}^K \sigma_k^\gamma, g)$, so the corresponding computation cost is $K\text{–}MultExp^1_{\mathbb{G}_1}(|p|) + Pair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^1_{\mathbb{G}_T}$
- TPA computes $\prod_{k=1}^K(\prod_{i\in L} e((\prod_{s\in I}(H(m_{k,s})^{v_s})^\gamma \times u_{k,i}^{\mu_k}, y_{k,i}))$, so the corresponding computation cost is $K(c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)) + Exp^{2L}_{\mathbb{G}_1} + Mult^L_{\mathbb{G}_1} + L\text{–}MultPair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^{K-1}_{\mathbb{G}_T}$

CS Computation: Batch auditing of Case 2

- CS selects $r \in \mathbb{Z}_p$, $|p| = 160$ bits
- CS computes for each user $R_{k,i} = e(u_{k,i}, y_{k,i})^r \in \mathbb{G}_T$, so the corresponding computation cost is $R_{k,i} = Exp^1_{\mathbb{G}_T}(|p|)$, and the size of R_i close to 960 bits
- CS computes $\gamma = h(\mathcal{R} \| \mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- CS computes for each user in each task: $\mu'_{k,i} = \sum_{s\in I} v_s m_{k,s}$, so the corresponding computation cost is $\mu'_{k,i} = Mult^c_{\mathbb{Z}_p} + Add^c_{\mathbb{Z}_p}$
- CS computes for each user in each task: $\mu_{k,i} = r + \gamma\mu'_{k,i}$, so the corresponding computation cost is $\mu_{k,i} = Add^1_{\mathbb{Z}_p} + Mult^1_{\mathbb{Z}_p}$

- CS computes $\sigma_k = \prod_{s\in I} \sigma_{k,s}^{v_s}$, so the corresponding computation cost is $\sigma_k = c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)$

TPA Computation: Batch auditing of Case 2

- TPA computes $\gamma = h(\mathcal{R}\|\mathcal{L})$, so the corresponding computation cost is $\gamma = Hash^1_{\mathbb{Z}_p}$
- TPA computes $\mathcal{R} \times e(\prod_{k=1}^{K} \sigma_k^{\gamma}, g)$, so the corresponding computation cost is $K\text{–}MultExp^1_{\mathbb{G}_1}(|p|) + Pair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^1_{\mathbb{G}_T}$
- TPA computes $\prod_{k=1}^{K}(\prod_{i\in L} e((\prod_{s\in I}(H(m_{k,s})^{v_s})^{\gamma} \times u_{k,i}^{\mu_{k,i}}, y_{k,i}))$, so the corresponding computation cost is $L(c\text{–}MultExp^1_{\mathbb{G}_1}(|v_i|)) + Exp^{2L}_{\mathbb{G}_1} + Mult^L_{\mathbb{G}_1} + L\text{–}MultPair^1_{\mathbb{G}_1,\mathbb{G}_2} + Mult^{K-1}_{\mathbb{G}_T}$

Experiment 1: Suppose we have two users who have the same files, case 1 for simplicity, and we want to check 100 blocks. Each auditing task consists of one file (e.g. "coding.pdf" with size of 513.4 KB and 125 blocks). This file is owned by two users. Table 4.2 shows the computation time of Mediator, CS, and TPA in seconds when the TPA checks 100 blocks of the file. The CS computed time is the time of generating the proof ,while the TPA time is the time of verifying the proof. In Figure 4.1, the chart shows that the computation time of (Mediator, CS, and TPA) is increased by increasing the number of the blocks of the file. However, the computation time of checking 100 blocks of the file consumed by the TPA is slightly larger than the computation time by CS.

Table 4.2: The computation time of (Mediator, CS, and TPA) when 100 blocks are checked. Times in seconds

File name	size	Number of blocks	Mediator computation time in seconds	CS computation time in seconds	TPA computation time in seconds
coding.pdf	513.4 KB	125	0.17	0.30	0.32
1.pdf	1.7 MB	415	1.66	0.49	0.53
4.pdf	4.8 MB	1181	12.99	0.96	1.05
6.pdf	6.3 MB	1536	21.89	1.15	1.27
7.ppt	7.1 MB	1725	26.86	1.27	1.39

Experiment 2: Suppose we have two users who have the same files, case 1 for simplicity, and we want to check the same file with different number of sampled blocks in each task. Each auditing task consists of one file (e.g. "7.ppt" with 1725 blocks). This file is owned by two users. Table 4.3 shows the computation time of Mediator, CS, and TPA in seconds when the TPA checks the file but every time he increases the number of sampled blocks. The computation time consumed by the Mediator is almost the same in each task since he checks the same file (same number of blocks). The CS computed time is the time of generating the proof ,while the TPA time is the time of verifying the proof. In Figure 4.2, the chart shows that the computation time of the CS and TPA is increased by increasing the number

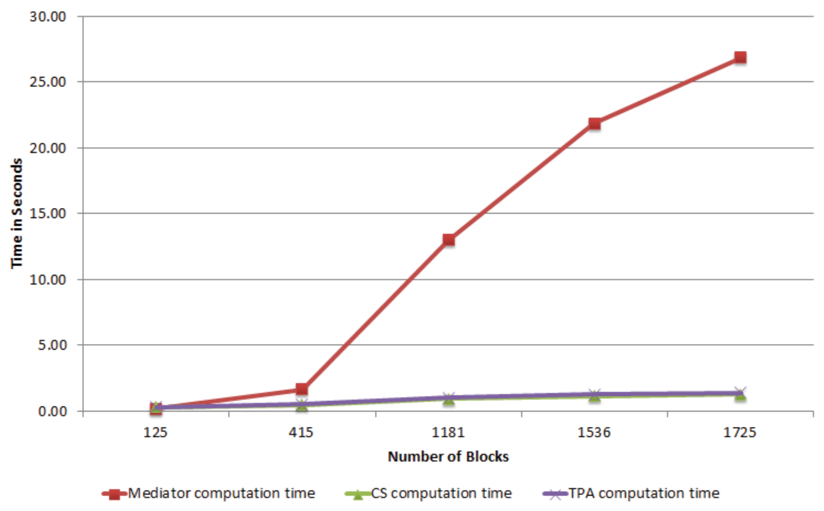


Figure 4.1: Comparison of the time consumed by the Mediator, the CS, and the TPA when 100 blocks are checked of each file in each auditing task. Each file is owned by two users

of the sampled blocks. However, the computation time consumed by the CS is slightly larger than the TPA computation time.

Table 4.3: The computation time of (the Mediator, the CS, and the TPA) when the same file is checked in each auditing task. However, the number of sampled blocks is increasing for each task. Times in seconds

Number of sampled block	Mediator computation time in seconds	CS computation time in seconds	TPA computation time in seconds
300	27.05	3.74	3.71
460	27.11	5.64	5.49
1000	26.68	11.70	11.08
1500	27.21	17.84	16.83
1725	27.57	20.74	19.44

4.3 Batch Auditing Efficiency

Due to the batch auditing process, the TPA can perform multiple auditing task in the same time. First, we compare the computational cost between the single auditing task and batch auditing task in order to test whether the batch auditing process is more efficient than multiple process of the single task. Table 4.4 shows the computational cost of the Audit phase of the single and Batch auditing in Case 1, for simplicity. Precisely, we compare the computational cost of the verification equations of both ways

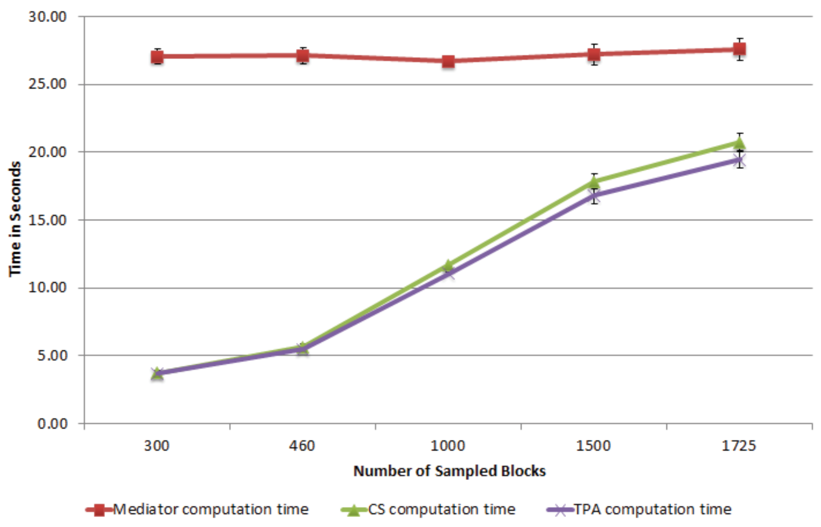


Figure 4.2: Comparison of the time consumed by (the Mediator, the CS, and the TPA) when the number of sampled blocks is increased in each auditing task of the same file "7.ppt". The file is owned by two users

(Equations: 3.2 and 3.4). The $Hash_{Z_p}$ operation is computed more in the single task than in the batch one. However, it is negligible and we may ignore it. There are more pairing operations in the Right Hand Side (RHS) of the single verification equation than in the Batch auditing equation. Moreover, the Left Hand Side (LHS) of both ways is almost the same except the batch auditing has $Mult_{G_T}$.

The author of (Martin(2013)) computed the time needed for the mathematical operations in pairing groups using CHARM benchmarking suite. Table 4.5 shows the computation time in seconds of multiplication, exponentiations, and pairing over type D elliptic curve. We use the table to compute the computational cost of the equations above and transfer Table 4.4 to computed times in Table 4.6. The latter table shows that the computation time of the LHS of both methods is almost the same. On the other hand, the the computation time of the RHS of the batch auditing when $K > 1$ is less than computation time of the single task, see Figure 4.3

In summary, the batch auditing process reduces the required number of pairing operations, the most expensive operation according to Table 4.5, from $(L+1)$ which would be required to run the single user case K times or $((L_1+L_2+\cdots+L_{K-1}+L_K)+K)$, to $((L_1+L_2+\cdots+L_{K-1}+L_K)+1)$ thus reducing the computational costs (Wang et al.(2013a)Wang, Chow, Wang, Ren, and Lou). For example, Table 4.7 shows that the number of pairing operations in batch auditing is less than if each task in performed as individual one.

Table 4.4: Comparison of the computational cost between the Single and Batch auditing tasks in the Audit phase. Where K = number of tasks, and L = consists of the number of subgroup of users in each task

	Single auditing task		
	γ	RHS	LHS
K = 1 , L = 2	$Hash^1_{Z_p}$	$Exp^1_{G_1} + Pair^1 + Mult^1_{G_T}$	$c\text{–}MultExp^1_{G_1} + Exp^4_{G_1} + Mult^2_{G_1} + 2\text{–}MultPair^1_{G_T}$
K = 1 , L = 3	$Hash^1_{Z_p}$	$Exp^1_{G_1} + Pair^1 + Mult^1_{G_T}$	$c\text{–}MultExp^1_{G_1} + Exp^6_{G_1} + Mult^3_{G_1} + 3\text{–}MultPair^1_{G_T}$
K = 2 , L = (2,2)	$Hash^2_{Z_p}$	$Exp^2_{G_1} + Pair^2 + Mult^2_{G_T}$	$2(c\text{–}MultExp^1_{G_1}) + Exp^8_{G_1} + Mult^4_{G_1} + 2\text{–}MultPair^1_{G_T}$
K = 3 , L = (2,3,2)	$Hash^3_{Z_p}$	$Exp^3_{G_1} + Pair^3 + Mult^3_{G_T}$	$3(c\text{–}MultExp^1_{G_1}) + Exp^14_{G_1} + Mult^7_{G_1} + 2\text{–}MultPair^2_{G_T} + 3\text{–}MultPair^1_{G_T}$
K = 4 , L = (3,3,2,2)	$Hash^4_{Z_p}$	$Exp^4_{G_1} + Pair^4 + Mult^4_{G_T}$	$4(c\text{–}MultExp^1_{G_1}) + Exp^20_{G_1} + Mult^10_{G_1} + 2\text{–}MultPair^2_{G_T} + 3\text{–}MultPair^2_{G_T}$

	Batch auditing task		
	γ	RHS	LHS
K = 1 , L = 2	$Hash1^1_{Z_p}$	$Exp^1_{G_1} + Pair^1_{G_T} + Mult^1_{G_T}$	$c\text{–}MultExp^1_{G_1} + Exp^4_{G_1} + Mult^2_{G_1} + 2\text{–}MultPair^1_{G_T}$
K = 1 , L = 3	$Hash1^1_{Z_p}$	$Exp^1_{G_1} + Pair^1_{G_T} + Mult^1_{G_T}$	$c\text{–}MultExp^1_{G_1} + Exp^6_{G_1} + Mult^3_{G_1} + 3\text{–}MultPair^1_{G_T}$
K = 2 , L = (2,2)	$Hash1^1_{Z_p}$	$2\text{–}MultExp^1_{G_1} + Pair^1_{G_T} + Mult^1_{G_T}$	$2(c\text{–}MultExp^1_{G_1}) + Exp^8_{G_1} + Mult^4_{G_1} + 2\text{–}MultPair^1_{G_T} + Mult^1_{G_T}$
K = 3 , L = (2,3,2)	$Hash1^1_{Z_p}$	$3\text{–}MultExp^1_{G_1} + Pair^1_{G_T} + Mult^1_{G_T}$	$3(Hash^c_{G_1} + c\text{–}MultExp^1_{G_1}) + Exp^{14}_{G_1} + Mult^7_{G_1} + 2\text{–}MultPair^2_{G_T} + 3\text{–}MultPair^1_{G_T} + Mult^2_{G_T}$
K = 4 , L = (3,3,2,2)	$Hash1^1_{Z_p}$	$4\text{–}MultExp^1_{G_1} + Pair^1_{G_T} + Mult^1_{G_T}$	$4(c\text{–}MultExp^1_{G_1}) + Exp^{20}_{G_1} + Mult^{10}_{G_1} + 2\text{–}MultPair^2_{G_T} + 3\text{–}MultPair^2_{G_T} + Mult^3_{G_T}$

Table 4.5: Computation time in seconds for mathematical operations in pairing groups

MultG_1	2.558×10^{-6}
MultG_2	2.360×10^{-5}
MultG_T	6.992×10^{-6}
ExpG_1	5.895×10^{-5}
ExpG_2	5.314×10^{-3}
ExpG_T	1.100×10^{-3}
Pairing	3.798×10^{-3}

Table 4.6: Comparison of the computationa time in seconds between the Single and Batch auditing tasks in the Audit phase. Where K = number of tasks, and L = consists of the number of subgroup of users in each task

	Computation time of single auditing task in seconds		Compuation time of batch auditing task in seconds	
	RHS	LHS	RHS	LHS
K = 1 , L = 2	3.864×10^{-3}	7.851×10^{-3}	3.864×10^{-3}	7.851×10^{-3}
K = 1 , L = 3	3.864×10^{-3}	1.178×10^{-2}	3.864×10^{-3}	1.178×10^{-2}
K = 2 , L = (2,2)	7.728×10^{-3}	1.570×10^{-2}	3.928×10^{-3}	1.571×10^{-2}
K = 3 , L = (2,3,2)	1.159×10^{-2}	2.748×10^{-2}	3.990×10^{-3}	2.749×10^{-2}
K = 4 , L = (3,3,2,2)	1.546×10^{-2}	3.925×10^{-2}	4.051×10^{-3}	3.928×10^{-2}

Table 4.7: Comparison between the number of pairing operations in the Single and Batch auditing tasks, where K = number of task, and L = the number of subgroup of users in each task

	Single Auditing Task	Batch Auditing Task
K=1, L=2	3	3
K=1, L=3	4	4
K=1, L=5	6	6
K=3, L=(2,2,2)	9	7
K=4, L=(3,2,3,2)	14	11
K=6, L=(5,3,5,5,3,5)	36	27
K=10, L=(5,5,5,5,5,5,5,5,5,5)	60	51

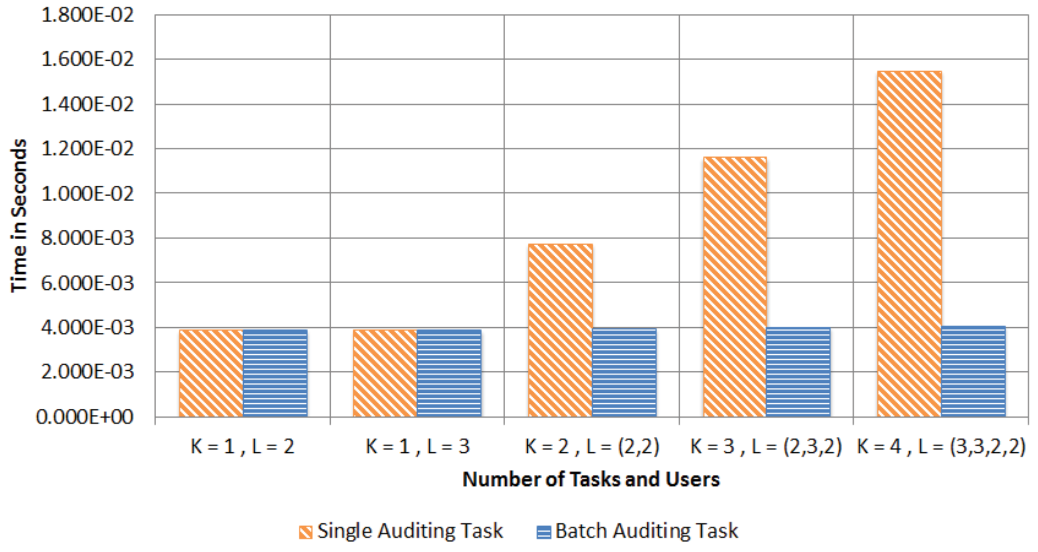


Figure 4.3: Comparison of the computational time in the RHS of the verification equations between the Single and Batch auditing tasks

Chapter 5

Conclusions and Future Work

In this chapter, we present our conclusion and some potential directions for future work.

5.1 Conclusions

Cloud data storage is one of the most commonly used services in the Cloud Computing paradigm. This service allows users to store their data remotely in the cloud and to access it from anywhere at any time. Data integrity and storage efficiency are two key requirements of outsourcing data to the cloud. To verify the integrity of the data stored in the cloud considering the large size of that data and the limited computing resources of users, it is important to enable public auditing services which is the focus of this thesis. To increase storage efficiency, data deduplication is applied to eliminate redundant copies of the data before sending it to the cloud.

We proposed a privacy-preserving public auditing scheme with data deduplication that achieves both data integrity and storage efficiency. We listed several requirements that need to be considered to construct an efficient auditing protocol. Our scheme covered most of the auditing requirements listed in 2.1.2 excepting recoverability and dynamic data operations. We can apply error correcting code to the file before it is send to the cloud, so the scheme can recover the entire data if failure occurred. The scheme also covers cross-user client side deduplication on block level that is achieved by the mediator who computes the hash value of each block to identify the duplicate data and aggregates multi-user signatures for duplicate blocks of data. So, the proposed scheme eliminates not only the duplicate data but also the signatures that used for data verification. By that, we reduce the amount of the uploaded data to the cloud storage and minimize the bandwidth between the enterprise and the cloud.

Then, We presented an extension to support batch auditing when the TPA performs many auditing tasks from different users at the same time. Another extension was also presented to support user revocation utilizing proxy re-signature scheme to allow the cloud to convert the signatures of the revoked user into mediator signatures while preserving the privacy of the private keys of both the revoked user and the mediator. By the proposed scheme, we allow the cloud to convert not only the private key of the revoked user but also one of the public parameter of that user, since each user has different private keys and public parameters.

We presented a detailed analysis of the security and the performance of our proposed schemes. We also showed the efficiency of the batch auditing technique which was demonstrated in terms of the number of pairing operations in the batch auditing, which was shown to be less than if each task is performed as an individual one. In addition, the computation time in the RHS of the verification equations done by the TPA of the batch auditing is less than the computation time of the single audit. So, our protocol is computationally lightweight, and is proven secure in the random oracle model.

There are some limitations of our proposed scheme which are: First, the data is static and we can solve that by considering dynamic data operations including data insertion, deletion, and multiplication. Second, the number of pairing operations, which is the most expensive one, needed for TPA verification is dependent of the number of users in each auditing task. Third, the data is not encrypted since we assume that the cloud is semi-trusted.

5.2 Future Work

The data in the cloud is not only being accessed by the user but also updated from time to time. We are planning to allow our proposed scheme to support dynamic data operations including: data insertion, deletion, and modification. In addition, we will allow the TPA to perform the verification process with constant number of pairing operations, which is the most expensive operation, which is independent of the number of users. Moreover, in terms of data deduplication in the cloud, Proof of Ownership (POW) schemes have been proposed. We want to expand our work to include POW to verify the ownership of duplicate data.

Bibliography

C. Wang, S. S. Chow, Q. Wang, K. Ren, and W. Lou, “Privacy-preserving public auditing for secure cloud storage,” *Computers, IEEE Transactions*, pp. 362–375, Feb 2013.

D. Harnik, B. Pinkas, and A. Shulman-Peleg, “Side channels in cloud services: Deduplication in cloud storage,” *Security Privacy, IEEE*, vol. 8, pp. 40–47, Nov 2010.

C. Soghoian. (2011, April) How dropbox sacrifices user privacy for cost savings. Accessed November 23, 2014. [Online]. Available: http://paranoia.dubfire.net/2011/04/how-dropbox-sacrifices-user-privacy-for.html

A. Boldyreva, “Threshold signatures, multisignatures and blind signatures based on the gap-diffie-hellman-group signature scheme,” in *Proceedings of the 6th International Workshop on Theory and Practice in Public Key Cryptography: Public Key Cryptography*, ser. PKC ’03. London, UK, UK: Springer-Verlag, 2003, pp. 31–46. [Online]. Available: http://dl.acm.org/citation.cfm?id=648120.747061

P. Mell and T. Grance, “The NIST definition of cloud computing,” 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

L. Li, L. Xu, J. Li, and C. Zhang, “Study on the third-party audit in cloud storage service,” pp. 220–227, 2011. [Online]. Available: http://dx.doi.org/10.1109/CSC.2011.6138525

S. Kamara and K. Lauter, “Cryptographic cloud storage,” pp. 136–149, 2010. [Online]. Available: http://dl.acm.org/citation.cfm?id=1894863.1894876

C. Delettre, K. Boudaoud, and M. Riveill, “Cloud computing, security and data concealment,” in *Proceedings of the 16th IEEE Symposium on Computers and Communications, ISCC 2011, Kerkyra, Corfu, Greece, June 28 - July 1, 2011.* IEEE, 2011, pp. 424–431. [Online]. Available: http://dx.doi.org/10.1109/ISCC.2011.5983874

J. Tate, P. Beck, H. Hugo Ibarra, S. Kumaravel, and L. Miklas. (2012, November) Introduction to storage area networks and system networking. An IBM Redbooks publication. [Online]. Available: http://www.redbooks.ibm.com/redbooks/pdfs/sg245470.pdf

T. Rivera. (2009) Understanding data deduplication. Storage Networking Industry Association (SNIA). Accessed November 23, 2014. [Online]. Available: http://www.snia.org/sites/default/education/tutorials/2009/fall/data/ThomasRivera_

UnderstandingDeduplication_A_Tutorial_Understanding_Dedupe_9-15-09.pdf

J. Singh. (2009, 09) Understanding data deduplication. Druva. Accessed November 23, 2014. [Online]. Available: http://www.druva.com/blog/understanding-data-deduplication/#.UxqW4YVnibE

P. Ananth and R. Bhaskar, "Non observability in the random oracle model," in *Provable Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2013, vol. 8209, pp. 86–103. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-41227-1_5

M. Bellare and P. Rogaway, "Random oracles are practical: A paradigm for designing efficient protocols," in *Proceedings of the 1st ACM Conference on Computer and Communications Security*, ser. CCS '93. New York, NY, USA: ACM, 1993, pp. 62–73. [Online]. Available: http://doi.acm.org/10.1145/168588.168596

D. Boneh, B. Lynn, and H. Shacham, "Short signatures from the weil pairing," in *Journal of Cryptology*. Springer-Verlag.

D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and verifiably encrypted signatures from bilinear maps," in *Proceedings of the 22Nd International Conference on Theory and Applications of Cryptographic Techniques*, ser. EUROCRYPT'03. Berlin, Heidelberg: Springer-Verlag, 2003, pp. 416–432. [Online]. Available: http://dl.acm.org/citation.cfm?id=1766171.1766207

G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 598–609. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315318

K. Yang and X. Jia, "Data storage auditing service in cloud computing: challenges, methods and opportunities," *World Wide Web*, pp. 409–428, 2012. [Online]. Available: http://dx.doi.org/10.1007/s11280-011-0138-0

S. G. Worku, Z. Ting, and Q. Zhi-Guang, "Survey on cloud data integrity proof techniques," in *Information Security (Asia JCIS), 2012 Seventh Asia Joint Conference on*, Aug 2012, pp. 85–91.

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: Advances in Cryptology*, ser. ASIACRYPT '08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 90–107. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-89255-7_7

K. Zeng, "Publicly verifiable remote data integrity," in *Information and Communications Security*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2008, pp. 419–434. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-88625-9_28

A. Juels and B. S. Kaliski Jr., "Pors: proofs of retrievability for large files," in *Proceedings of the 14th ACM conference on Computer and communications security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 584–597. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315317

M. A. Shah, R. Swaminathan, and M. Baker, "Privacy-preserving audit and extraction of digital contents," *IACR Cryptology ePrint Archive*, 2008. [Online]. Available: https://eprint.iacr.org/2008/186.pdf

M. A. Shah, M. Baker, J. C. Mogul, and R. Swaminathan, "Auditing to keep online storage services honest," in *Proceedings of the 11th USENIX Workshop on Hot Topics in Operating Systems*, ser. HOTOS'07. Berkeley, CA, USA: USENIX Association, 2007, pp. 1–6. [Online]. Available: http://dl.acm.org/citation.cfm?id=1361397.1361408

Q. Zheng and S. Xu, "Secure and efficient proof of storage with deduplication," in *Proceedings of the second ACM conference on Data and Application Security and Privacy*, ser. CODASPY '12. New York, NY, USA: ACM, 2012, pp. 1–12. [Online]. Available: http://doi.acm.org/10.1145/2133601.2133603

J. Yuan and S. Yu, "Secure and constant cost public cloud storage auditing with deduplication," *IACR Cryptology ePrint Archive*, pp. 149–149, 2013.

S. Lu, R. Ostrovsky, A. Sahai, H. Shacham, and B. Waters, "Sequential aggregate signatures and multisignatures without random oracles," pp. 465–485, 2006. [Online]. Available: http://dx.doi.org/10.1007/11761679_28

B. Waters, "Efficient identity-based encryption without random oracles," in *Advances in Cryptology, EUROCRYPT 2005*, ser. Lecture Notes in Computer Science. Springer Berlin Heidelberg, 2005, pp. 114–127. [Online]. Available: http://dx.doi.org/10.1007/11426639_7

A. Boldyreva, C. Gentry, A. O'Neill, and D. H. Yum, "Ordered multisignatures and identity-based sequential aggregate signatures, with applications to secure routing," in *Proceedings of the 14th ACM Conference on Computer and Communications Security*, ser. CCS '07. New York, NY, USA: ACM, 2007, pp. 276–285. [Online]. Available: http://doi.acm.org/10.1145/1315245.1315280

Y. Wen and J. Ma, "An aggregate signature scheme with constant pairing operations," in *Proceedings of the 2008 International Conference on Computer Science and Software Engineering*, ser. CSSE '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 830–833. [Online]. Available: http://dx.doi.org/10.1109/CSSE.2008.941

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

R. F. Martin, "Group selection and key management strategies for ciphertext-policy attribute-based encryption," Master's thesis, Rochester Institute of Technology, New York, August 2013. [Online]. Available: www.cs.rit.edu/~rfm6038/Thesis.pdf