Third-Party based Data Auditing Service (TP-DAS)

Progress Report

Written by: Maher Alharby
Supervised by: Dr. Ning Zhang

May 2015
4.2 Implementation Methodology.................................................................28
  4.2.1 Selection of Programming Languages ..............................................28
  4.2.2 Selection of Development Environments .........................................28

4.3 Testing and Evaluation Methodology ....................................................29

4.4 Project Plan .........................................................................................29

4.5 Chapter Summary ..............................................................................30

References................................................................................................31
List of Figures

FIGURE 1.1: SYSTEM STRUCTURE ........................................................................................................ 3
FIGURE 2.1: DIGITAL SIGNATURE METHOD ..................................................................................... 6
FIGURE 2.2: MESSAGE AUTHENTICATION CODE (MAC) METHOD ............................................... 7
FIGURE 2.3: HASH METHOD ............................................................................................................. 8
FIGURE 2.4: SENTINEL-BASED PROOF OF RETRIEVABILITY ....................................................... 11
FIGURE 2.5: PROVABLE DATA POSSESSION PROTOCOL .......................................................... 13
FIGURE 2.6: PUBLIC AUDITING SCHEME USING THIRD PARTY AUDITOR (TPA) ...................... 15
FIGURE 3.1: TP-DAS MODEL OVERVIEW ...................................................................................... 18
FIGURE 3.2: SYSTEM ARCHITECTURE ......................................................................................... 20
FIGURE 3.3: ALGORITHMS USED IN THE PROTOCOL .............................................................. 22
FIGURE 3.4: THE INITIALISATION PHASE PROTOCOL .............................................................. 24
FIGURE 3.5: THE VERIFICATION PHASE PROTOCOL ..................................................................... 25
FIGURE 3.6: THE DYNAMIC DATA OPERATIONS PHASE PROTOCOL ....................................... 27
FIGURE 4.1: PROJECT PLAN ........................................................................................................... 30

List of Tables

TABLE 2.1: A COMPARISON BETWEEN DIFFERENT REMOTE DATA INTEGRITY CHECKING SCHEMES. .......... 17
TABLE 3.1: THE NOTATIONS USED IN THE SYSTEM ......................................................................... 21
Abstract

Cloud computing has emerged to provide various hosted services to the end user. One of these services is the storage service that allows users to shift their data into the cloud in a cost-effective manner. By shifting data into the cloud, users will be alleviated from the burden of storing data locally, and the maintenance costs associated with this. However, as data is stored in the cloud, users will no longer have the ultimate control over their data. Thus, data integrity is the biggest challenge concerning users. Users’ data needs to be checked regularly while it is in the cloud. Yet, users might have limited resources and capabilities, so they can rely on a third party auditor (TPA) to check the integrity of the data on their behalf.

The aim of this project is to design and implement an efficient solution that enables data users to check the integrity of their data while it is in the cloud by resorting to a TPA. The solution should enable users to detect any data integrity drift such as data alteration or data loss. Also, the solution should preserve the privacy of users’ data by prohibiting unauthorised people, malicious parties or even the TPA from accessing the content of the users’ data.
Chapter 1. Introduction

1.1 Introduction and Motivations

Cloud computing has gained a wide popularity due to the services that it provides. One of the services offered by the cloud is the storage service; this allows cloud users to store their data in the cloud in a cost-effective manner, without worrying about the management of the underlying infrastructure. The storage service also allows users to access their data remotely at any time.

Although cloud computing has various appealing advantages, it brings various security challenges towards users’ remote data. As users’ outsourced data is managed by a separate administrative party, users will no longer have the ultimate control over their data [5]. Therefore, one of the biggest challenges concerning data users is how to ensure the integrity of their data while it is in the cloud. Users may be concerned about this issue for various reasons. First of all, data could be lost due to hardware or software failures at the Cloud Service Providers’ (CSP’s) side. Secondly, the CSP could hide data loss incidents from data users in order to maintain their reputations. Another reason is that the CSP may deliberately delete rarely accessed data for saving more storage capacity [3].

Users’ data needs to be checked regularly while it is in the cloud. However, as users may have many outsourced data files, they will find it difficult to check the integrity of their data files themselves due to their limited computing resources [5]. In addition, data integrity check could be expensive at the users’ side in terms of computation and storage [6]. To tackle these issues, users can resort to a Third Party Auditor (TPA) to check the correctness of the remote data on their behalf. Resorting to a TPA can have various benefits. The first benefit is to eliminate the burden of checking the data integrity at the client’s side, resulting in saving users’ computation resources. Another benefit is that TPA is usually a specialist who has more capability and computational resources than ordinary users [5].

In this project, an efficient solution will be implemented to enable data users to check the correctness of their data while it is in the cloud by resorting to a TPA. The solution should enable users to detect any data integrity drift such as data alteration or data loss. Also, the solution should preserve the privacy of users’ data by prohibiting unauthorised people or even the TPA from accessing the content of the users’ data.
1.2 Project Aim and Objectives

The main aim of this project is to design and implement a system that allows users to check the integrity of their data while it is in the cloud by resorting to a third party auditor (TPA). The TPA will perform the task of checking the integrity of users’ data on behalf of the users. Yet, the TPA should not be able to access the content of users’ data during the auditing process, and therefore, the privacy of users’ data will be maintained. The system will also maintain data confidentiality by preventing malicious parties from accessing the content of users’ data while it is in the cloud.

The system consists of three different entities, namely, users, Cloud Service Provider and Third Party Auditor. The entire system will work as in Figure 1.

1- **Users** who have data files to be stored into the cloud computing server.
2- **Cloud Service Provider (CSP)** that provides data storage services as it has huge storage capacity as well as huge computing resources.
3- **Third Party Auditor (TPA)** who is responsible for checking the integrity of the remote data on behalf of the users as he has more expertise and capability that users do not have.

Several specific objectives need to be achieved in order to end up successfully with the entire system. These objectives are as follows:

- Understand the concept of cloud computing in terms of storing users’ data. Thus, a database will be created to store users’ data files.
- Read and investigate in the existing data integrity check methods such as MAC and Hash methods. Then, more reading is required to fully understand the existing protocols such as Provable Data Possession that can be used to detect data integrity drifts.
- Create and design a new protocol that can detect remote data integrity drifts as well as preserving the privacy of users’ data.
Create a graphical-based application that allows users to perform the following tasks:

1. Upload their data files to be stored in the database.
2. View or download their remote data files at any time via the application.
3. Check the integrity of their remote data files by resorting to a third party auditor for this task.
4. Perform data operations on their data files such as insertion, deletion or alteration while it is in the cloud when needed.

Analyse, test and evaluate the new protocol.

1.3 Report Structure

In addition to the introduction chapter, the progress report will cover three other chapters. Chapter 2 will start with an overview about cloud computing and its security issues (i.e. data integrity). Then, it will study and critically analyse the current data integrity checking methods and schemes. Chapter 3 will explain the preliminary design of the proposed system (TP-DAS). Chapter 4 will provide a plan for the implementation and the evaluation of the proposed system. The timeline for achieving this plan is also provided.
Chapter 2. Background and Literature Review

2.1 Chapter Overview

This chapter will first provide a brief introduction to cloud computing and its security issues, followed by different methods such as digital signature that can be used to detect data integrity drifts. Then, a discussion regarding the current data integrity checking schemes is stated. From the provided discussion, a vision to a new solution that can address unsolved issues will be provided.

2.2 Cloud Computing Overview

Due to the unprecedented revolutions in technology, cloud computing paradigm has emerged as an alternative solution to the in-house architecture. Cloud computing can be defined as “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” [1]. Cloud computing delivers various services to the end users over the Internet [18]. Cloud services can fall into three types: Infrastructure as a service (IaaS), Platform as a service (PaaS) and Software as a service (SaaS) [1].

The storage service is one of the cloud services that allows users to shift their data into the cloud to gain a multitude of significant advantages, namely, “on-demand services, usage-based pricing, ubiquitous network access and location-independent resource pooling” [2][3]. In addition, users will also benefit from moving their data into the cloud as they do not have to worry about the burden of storage management as well as they will avoid the expenditure of purchasing or managing hardware and software [4].

2.3 Security Issues in Cloud Computing

Data security must be taken into account as shifting data into the cloud could result in many security breaches such as data leakage, data loss, data alteration and unauthorised data access. Data security here refers to data confidentiality, integrity and availability. Data confidentiality means data must not be accessed except by authorised users [20]. Data confidentiality can be achieved by the adoption of encryption techniques. Data availability means data must be always accessed even in the case of unpleasant disasters such as power outages. Data integrity means data must not be intentionally or accidentally altered or tampered. Data integrity can be achieved by the adoption of hash function and digital signature techniques. However, those techniques cannot be used directly to detect data integrity drifts for the following reason. When users shift their data into the cloud, they delete their local copy of the data files. Thus, the challenge is how to ensure the integrity of the data
without having a local copy of the data. Downloading data files for the purpose of checking their integrity is not an efficient solution due to the high network bandwidth required [5].

The focus of this project will be on achieving data integrity and data confidentiality. Specifically, the project will propose a solution to ensure that users’ data is maintained in its original state without alteration or corruption, and without requiring the local copy of the data. The solution will also ensure that users’ data cannot be accessed except by authorised users.

2.4 Methods for Detecting Data Integrity Drifts

There are various methods that can be used to detect data integrity drifts. These methods can be used to build an efficient system that requires less communication and computation overhead. There are three methods that will be discussed in this section, which are digital signature, Message Authentication Code (MAC) and hash function.

2.4.1 Digital Signature

Digital signature is basically an electronic signature that authenticates the sender of the messages to the recipient. So the recipient will ensure that the message has been sent by a known sender [21]. In addition, digital signature can be used to assure that the data has not been tampered while it is in transit [22]. Thus, the integrity of the data is checked.

The digital signature method consists of three algorithms, namely, keyGen, Signature and Validation [21]:

- KeyGen: this algorithm produces a private key and its corresponding public key.
- Signature: this algorithm takes the message and the private key, and produces the digital signature.
- Validation: this algorithm takes the signature and the public key to recover the message.

If the recovered message matches the original one, the signature is valid (the message is origin).

![Figure 2.1: Digital Signature Method [23]](image-url)
However, to prevent the verifier from accessing the content of the data during the verification process, the data can be encrypted or hashed prior signing it. Therefore, the verifier will only access the data in an encrypted format.

Advantages:

- This method can detect data integrity drifts.
- This method can preserve the privacy of the data if the data is encrypted prior generating the signature.

Drawbacks:

- Signatures have an expiry date. So data needs to be re-signed after that date.

2.4.2 Message Authentication Code (MAC)

Message Authentication Code (MAC) is a portion of information that can be used to provide an assurance regarding the authenticity and integrity of the data [24]. MAC method with the help of the secret key can be used to detect data alteration.

The MAC method takes a secret key and a message as input, and generates a MAC. To verify the integrity of the data, the verifier has to use the secret key to generate a new MAC and then compare it with the received one. If the two MACs are the same, then the data is origin.

![Figure 2.2: Message Authentication Code (MAC) Method [24]](image)

Advantages:

- As the generation and verification of MACs are done by the same secret key, the MAC method is suitable for encrypting large files unlike the digital signature method.
Drawbacks:

- The secret key needs to be shared between the sender and the receiver. Thus, the distribution of the key might impose a risk towards the data if the key is compromised.

2.4.3 Hash Function

A cryptographic hash function can be used to detect data integrity drifts. It takes a message of a variable length as input, and produces a fixed size message digest [25]. It does not require any key for generating the message digest unlike the MAC method.

![Figure 2.3: Hash Method [25]](image)

To verify the integrity of the message, the verifier has to compute a fresh hash value of the message and then compare it with the received one. If the two hash values are the same, the message is origin.

Advantages:

- This method can protect the privacy of the data as the verifier can only access the hashed value of the message instead of the original message.
- This method does not require any key.
2.5 Remote Data Integrity Checking Schemes

There are various schemes that can be used to detect data integrity drifts while data is stored in the cloud. A description of these schemes including their advantages and drawbacks will be provided in this section.

2.5.1 Basic Schemes

2.5.1.1 MAC based scheme

A straightforward scheme to check the integrity of remote data files can be implemented using Message Authentication Code (MAC) [7]. Before uploading the file \( F \) to the CSP, the user needs to pre-compute a specific number of MACs depending on the number of auditing attempts required using a different secret key for each MAC. For example, if the file needs to be audited ten times, the user has to compute ten MACs using ten different keys. The user then has to store the MACs along with the secret keys at his side. To verify the integrity of data, the user can each time challenge the CSP by releasing one of the keys. The CSP then has to respond by computing the MAC using the received key. If the returned MAC matches the stored one, the data is not tampered. However, once all keys have been used for auditing, the user has to retrieve the data and compute new MACs.

Advantages:

- This scheme provides a simple deterministic data integrity assurance for remote data.

Drawbacks:

- This scheme supports only limited numbers of data auditing attempts depending on the number of the used secret keys. Users have to retrieve the data to compute new MACs if all keys are used.
- This scheme does not support public auditability due to the use of secret keys. Thus, only data owner who has the keys can audit the data.

2.5.1.2 Digital signature based scheme

MAC scheme can neither support public auditability nor unlimited number of data auditing attempts as discussed earlier. To tackle these issues, digital signature can be used alternatively [7]. In this scheme, users pre-compute the signature for each data block and then send both the blocks and their corresponding signatures to the CSP. To verify the integrity of data, users can challenge the CSP by requesting a number of arbitrarily selected blocks along with their corresponding signatures. Users then use the public key to generate a signature for
each block received. The generated signatures will be compared with the received ones. If they are equal, the data is origin. Thus, this scheme provides a probabilistic assurance for remote data.

**Advantages:**

- This scheme provides a simple probabilistic data integrity assurance for remote data.
- This scheme supports public auditability because everyone who has the public key can audit the data.
- This scheme supports unlimited number of data auditing attempts because user can verify the data integrity many times by using the public key.

**Drawbacks:**

- This scheme requires retrieving a number of data blocks which might lead to a communication overhead.

2.5.2 Proof of Retrievability (PoR)

2.5.2.1 Basic Proof of Retrievability

The previous scheme (Digital signature) requires retrieving the data for every auditing process. However, this is not efficient as it requires high network bandwidth for transferring the data. To tackle this issue, the basic PoR scheme [8] has been proposed to enable data users to check the integrity of their remote data files without the need of retrieving those files. The basic PoR is based on a keyed hash function \( H_k(F) \), where \( F \) is the file and \( k \) is a secret key. Prior uploading the file \( F \) to the CSP, the user first computes the hash value of the file \( F \) using the secret key \( k \), and then stores \( H_k(F) \) along with the used key \( k \) at his side. The data file \( F \) then can be uploaded to the CSP. To check that the CSP possesses the correct file, the verifier (the user in this case) releases the key \( k \) to the CSP and asks it to compute and return \( H_k(F) \). If the returned \( H_k(F) \) matches the stored one, then the file is not tampered. To support multiple auditing attempts, the user can compute and store multiple hash values of the file \( F \) using different secret keys. For each auditing request, the verifier can release one of the keys to the CSP and ask it to compute and return \( H_k(F) \).

**Advantages:**

- This scheme allows verifiers to check the integrity of remote files without the need of retrieving those files.
- This scheme allows verifiers to do multiple data integrity checks over their files by computing and storing multiple hash values using different keys.

**Drawbacks:**

- The verifier has to compute and store a certain amount of hash values for each file depending on the number of the required checks. Therefore, this would result in
storage burden at the verifier’s side (i.e. the verifier storage is O(c) where c is the number of the required checks), especially when the number of checks is relatively large.

- For each auditing request, the CSP has to access and process the whole data to generate the hash value of the data that will be sent as a response to the verifier. This might be computationally expensive (i.e. the server computation is O(n) where n is the number of the file blocks), despite the fact that hashing is a lightweight operation.

### 2.5.2.2 Sentinel-based Proof of Retrievability

As discussed above, the basic POR imposes a storage and computational cost at the verifier’s and the CSP’s side respectively. These issues have been addressed by Juels and Kaliski[8] who have proposed a developed POR scheme that is based on sentinels. In their scheme, the file F is encrypted and randomly embedded with a set of check blocks (sentinels). The purpose of encrypting the file F is to ensure that the sentinels cannot be distinguished from other file blocks. The use of sentinels here is to verify the integrity of users’ data files. The sentinels and their corresponding positions will be stored at the verifier’s side (the user in the case), while the encrypted file will be sent to the CSP. To verify the data integrity, the verifier challenges the CSP by identifying the positions of a set of sentinels and requesting the CSP to return the sentinel values. Thus, the CSP will only need to access a portion of the file to return the sentinel values instead of the whole file. In their scheme, data files can be encoded by error-correction codes prior the encryption process to protect against corruption at the CSP side.

![Figure 2.4: Sentinel-based Proof of Retrievability](image)
Advantages:

- In this scheme, only a portion of the file F will be accessed by the CSP to generate the response. This is not the case in the basic POR where the CSP has to access the entire file as discussed earlier.

Drawbacks:

- This scheme requires pre-processing the file F before uploading it to the CSP. Users need to encode the file with error-correction codes and then encrypt the file. Thus, this pre-processing step would impose computational overhead at the users’ side.

2.5.2.3 Compact Proof of Retrievability

Shacham and Waters [11] have later proposed two POR schemes that guarantee the shortest query and response compared to earlier POR schemes. Their schemes are with full proofs of security against arbitrary adversaries. Their first scheme can support public auditability and it is based on homomorphic authenticators constructed from the BLS signatures. It has an advantage in terms of having the shortest query (20 bytes) and response (40 bytes) compared to other POR schemes. In addition, it is secure in the random oracle model defined in [11].

Their second scheme can support private auditability and it is based on homomorphic authenticators constructed from pseudorandom functions (PRFs). It has an advantage in terms of having the shortest response (20 bytes) compared to other POR schemes. Furthermore, it is secure in the standard model defined in [11]. However, both schemes can only deal with static data.

Advantages:

- These schemes improve the challenge-response protocol by guaranteeing the shortest queries and responses compared to previous POR.

Drawbacks:

- These schemes do not support dynamic data operations such as modification and insertion.

2.5.3 Provable Data Possession (PDP)

2.5.3.1 Basic PDP

All the above POR schemes can neither support public auditability nor dynamic data operations. To support public auditability, Ateniese et al. [12] have proposed a provable data possession (PDP) scheme that allows users, and not only the data owner, to check the integrity of their data without retrieving it. In addition, their scheme can support partial
dynamic data operations (appended only). That means, users can add new blocks only at the end of the file. Their scheme consists of two main phases:

1- Pre-processing phase

Initially, the user has to pre-process the file F prior storing it in the CSP. The user splits the file F into a number of blocks, so the file $F = \{\text{block1}, \text{block2}, \text{block3}, \ldots, \text{blockn}\}$ where n is the total number of the file blocks. After that, the user has to generate a unique metadata (called tag) for each block. The generated tags will be stored at the verifier side (the user in this case) while the modified data file F will be uploaded to the CSP.

2- Verification phase

To verify that the CSP possesses the file F, the verifier sends a data possession challenge to the CSP by specifying a set of data blocks and asking the CSP to generate and send the corresponding tags. The verifier then can verify the response by using its local tags. If the returned tags match the stored ones, the data file F is origin (not tampered).

![Figure 2.5: Provable Data Possession protocol [12]](image)

Advantages:

- This scheme supports public auditability because everyone who has the tags can audit the data.
- This scheme allows users to append new blocks (i.e. a blockX) to their files. Users have consequently to generate a new tag for the new block (i.e. a tagX).
Drawbacks:

- This scheme does not fully support dynamic data operations such as adding new blocks in the middle of the file because the corresponding tags will have to be retrieved and modified by the user.

2.5.3.2 Dynamic PDP (DPDP)

The basic PDP scheme does not fully support dynamic data operations as discussed above. To fully support dynamic data operations, Erway et al. [15] have extended the basic PDP in [12] to support dynamic data operations such as insertion, deletion and modification. For a file $F$ that consists of a number of blocks, users can insert a new block anywhere in the file. They can also delete any block or modify an existing block. In their scheme, rank-based authenticated skip lists are used for provable updates.

Advantages:

- This scheme supports fully dynamic data operations such as insertion, deletion and modification.

Drawbacks:

- This scheme is not efficient due to the use of authenticated skip list.

2.5.4 Third Party Auditing Schemes (TPA)

As discussed above, earlier schemes such as MAC based and PoR schemes can only support private auditability. Private auditability allows only data owners to check the integrity of their data while it is in the cloud which might impose computational burden at the data owners’ side. Later schemes such as PDP have adopted public auditability to allow everyone, who has an access right to the data, to check the integrity of the data, and not only the data owner [16].

However, there are various schemes that have adopted public auditability with the help of a Third Party Auditor (TPA). These schemes alleviate data owners and users from performing complex computation by resorting to the TPA. In addition, the TPA is usually a specialist who has more capability and computation resources that ordinary users do not have.

2.5.4.1 Trusted Third Party Auditor (TTPA)

Unlike the previous schemes, this scheme requires a Trusted Third Party Auditor (TTPA) for checking the integrity of remote data files on behalf of users [7].

This scheme consists of four algorithms, namely, $KeyGen$, $SignGen$, $GenProof$ and $VerifyProof$ [7].
1. **KeyGen:** this algorithm is run by the user to produce public and private keys.
2. **SignGen:** this algorithm is also run by the user to establish the verification metadata that will be used for data auditing.
3. **GenProof:** this algorithm is run by the CSP to generate a proof of the stored data.
4. **VerifyProof:** this algorithm is run by the TPA to verify the proof generated by the CSP.

This scheme can be divided into two different phases, **Setup** and **Audit**:

- **Setup phase:** The user executes the **KeyGen** algorithm to establish the private and public keys. The user then executes the **SignGen** algorithm that pre-processes the data file F in order to generate the verification metadata. After that, the data file F can be uploaded to the CSP, while the verification metadata can be sent to the TPA for auditing.

- **Audit phase:** The TPA could release a data possession challenge to the CSP. The CSP has to pre-process the file using the **GenProof** algorithm to generate the response and then send it to the TPA. The TPA then uses the verification metadata to verify the response by executing **VerifyProof** algorithm.

![Figure 2.6: Public auditing scheme using Third Party Auditor (TPA) [17]](image)

**Advantages:**

- This scheme reduces the cost of computational resources at the user’s side as the user will resort to a trusted TPA for data auditing.
- This scheme support public auditability as the TPA can audit users’ data on their behalf.
Drawbacks:

- This scheme does not preserve the privacy of users’ data as the CSP has to send linear combinations of file blocks to the TPA. Thus, the TPA could solve some linear equations in order to derive users’ data.

2.5.4.2 Privacy-Preserving Third Party Auditor (PP-TPA)

The above mentioned scheme might result in users’ data being leaked to the TPA. To prevent the TPA from accessing users’ data during the auditing process, Cong Wang et al [7] have developed the previous TTPA scheme to support the privacy of users’ data. In their scheme, the TPA cannot learn any knowledge regarding the content of users’ data during the auditing process. In their scheme, homomorphic authenticator technique is used. Homomorphic authenticators (verification metadata) are generated from each file block. To achieve privacy-preserving, the homomorphic authenticators can be uniquely integrated with random masking. When the CSP generates a response to the TPA challenge, the linear combination of data blocks in the response can be masked with randomness. Due to the random masking, the TPA will be unable to solve the linear equations to access the content of users’ data. However, data files at the CSP might be accessed by malicious parties as data is not encrypted.

Advantages:

- This scheme preserves the privacy of users’ data due to the random masking technique. So, the TPA cannot derive the content of users’ data.

Drawbacks:

- This scheme may result in users’ data being accessed by malicious parties.

2.5.5 Further Discussions

In this section, we have discussed various schemes that can be used to detect data integrity drifts. The following table represents a brief comparison between these schemes in terms of the supported features:
Table 2.1: A comparison between different remote data integrity checking schemes.

<table>
<thead>
<tr>
<th>Schemes</th>
<th>Basic schemes</th>
<th>POR schemes</th>
<th>PDP schemes</th>
<th>TPA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features</td>
<td>MAC</td>
<td>Digital</td>
<td>Basic</td>
<td>Compact</td>
</tr>
<tr>
<td>Unbounded number of data</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>auditing attempts</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Auditing without</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>retrieving data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dynamic data operations</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Public auditability</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>The use of TPA</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Privacy preserving</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Confidentiality</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

All the schemes, which have been discussed in this section, can be used to detect data integrity drifts, although they vary in terms of the supported features. Interestingly, none of these schemes can achieve data confidentiality by protecting users’ data from malicious parties. Malicious parties may access the content of users’ data while it is in the cloud. Therefore, in Chapter 3, we are going to propose an efficient solution that achieves both data integrity and data confidentiality. To achieve confidentially, data will be encrypted prior uploading it to the CSP.

2.6 Chapter Summary

This chapter begins with a brief overview of cloud computing and its security issues, followed by different methods that can be used to detect data integrity drifts. A detailed discussion regarding the current data integrity checking schemes is also provided. An insight into the proposed solution that will be explained in chapter 3 is also included.
Chapter 3. Third-Party based Data Auditing Service (TP-DAS) Design

3.1 Chapter Overview

This chapter provides a preliminary design of the proposed system. The system will be called Third-Party based Data Auditing Service (TP-DAS). The system model overview is first provided to describe the general concept of the system’s design. This is followed by the design requirements. After that, a deep description of the system’s design will be explained embracing system architecture, assumptions, notations and the protocol design.

3.2 System Model Overview

The system consists of three different entities, namely, users, Cloud Service Provider (CSP) and Third Party Auditor (TPA). Users can rely on the CSP to store their data and then later they can access, edit or audit their data. To audit the stored data, users can resort to a TPA to check the integrity of the data on their behalf. However, the TPA should not be able to access the content of users’ data during the auditing process.

The general system model will be as follow. Users firstly encrypt the data file (F) and then pre-process the encrypted file (F’) to generate the verification metadata. The F’ and the verification metadata will be uploaded to the CSP and the TPA respectively. To verify the integrity of users’ data files, users can ask the TPA to verify the integrity of their data. The TPA will then issue a verification challenge to the CSP. Upon receiving the challenge, the CSP has to respond correctly to the TPA by returning the required proof. The TPA will then verify the response and inform the users about whether their data has been tampered or not.

![Figure 3.1: TP-DAS model overview](image-url)
3.3 Design Requirements

The following requirements need to be met in the proposed system:

1. The system should check the integrity of remote data files without having the local copy of those files.
2. The system should maintain the confidentiality of users’ data by prohibiting malicious parties (i.e. a malicious CSP) from accessing the data.
3. The system should prohibit the TPA from learning any knowledge regarding the content of users’ data during the auditing process.
4. The system should allow the TPA to perform auditing tasks with minimum communication and computation overhead.
5. The system should allow unbounded number of data auditing attempts.
6. The system should support dynamic data operations, i.e. insertion, deletion or alteration.
7. The system should support public auditability by allowing everyone, not only the data owner, to verify the integrity of the data.

3.4 TP-DAS Design

3.4.1 System Architecture

There are three parties involved in the system, which are user, Cloud Service Provider (CSP) and Third Party Auditor (TPA).

The design of the system requires creating a graphical-based application. To use the application, the user has to log into the system through a login page using their name and password. In case of new users, the registration page can be used to setup new accounts. Upon a successful login, the user can view their existing files or even download them if required. To upload a new file to the CSP, the user has first to encrypt the file and then generate the file’s verification metadata. The encrypted file will be sent to the CSP to be stored in its database, while the verification metadata will be sent to the TPA for auditing purposes. The user can also use the application to update their existing files. To update existing files, the user can send a request including the related data to the CSP and the TPA. Upon receiving the request, the CSP will update the existing file based on the received request. Similarly, the TPA has to update the stored verification metadata.

To audit the integrity of data, the user could use the application to send a verification request to the TPA. The request includes the file name that needs to be verified. Upon receiving the request, the TPA will issue a verification challenge and send it to the CSP. With the use of the received challenge, the CSP then pre-processes the user’s encrypted file to generate the proof. The generated proof then will be sent to the TPA. After that, the TPA has to use the stored verification metadata to verify the CSP’s proof. Finally, the verification result will be sent to the user.
3.4.2 Assumptions

There are various assumptions that must be taken into account in the proposed system, which are:

1- The CSP is not trusted, and therefore, a trusted TPA is needed to audit users’ data.
2- The TPA is trusted but curious. That means, the TPA might access the content of users’ data.
3- It is assumed that the communication between entities is done via a secure channel.
4- It is assumed that the user has logged into the system successfully using their name and password.
3.4.3 Notations and Preliminaries

This table describes the notations used in the system.

<table>
<thead>
<tr>
<th>Notation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User</td>
<td>Who has data to be stored in the cloud.</td>
</tr>
<tr>
<td>CSP</td>
<td>The Cloud Service Provider.</td>
</tr>
<tr>
<td>TPA</td>
<td>Third Party Auditor.</td>
</tr>
<tr>
<td>$F, F'$</td>
<td>The user’s file, the encrypted file.</td>
</tr>
<tr>
<td>$D$</td>
<td>The set of verification metadata that used for auditing purposes.</td>
</tr>
<tr>
<td>$m_i, M_i$</td>
<td>The $i$th data block as the file is divided into a number of blocks, the encrypted $i$th block.</td>
</tr>
<tr>
<td>$d_i$</td>
<td>The verification metadata for the $i$th block.</td>
</tr>
<tr>
<td>$u_p, u_p$</td>
<td>The user’s public key, the user’s private key.</td>
</tr>
<tr>
<td>$f_k(.)$</td>
<td>A keyed Pseudo Random Function (PRF).</td>
</tr>
</tbody>
</table>

| $\sum_{i=1}^{n} x_i$ | $x_1 + x_2 + x_3 + \ldots + x_n$ |
| $\prod_{i=1}^{n} x_i$ | $x_1 * x_2 * x_3 * \ldots * x_n$ |

Table 3.1: The notations used in the system

There are two main cryptographic primitives used in the design of the system, which are as follows:

**Elliptic Curve Cryptography (ECC)**

The design of the system will make use of elliptic curve over the ring $\mathbb{Z}_n$. Let assume $n$ is an integer. Two integers in $\mathbb{Z}_n$ (a and b) are chosen to satisfy $\gcd(4a^3+27b^2, n)=1$. $\text{En}(a,b)$ can be defined as the set of pairs $(x,y) \in (\mathbb{Z}_n)^2$ satisfying $y^2+ax+b \pmod{n}$ along with the point $O_n$ at infinity [26].

**Pseudo Random Function (PRF)**

The design of the system will also make use of Pseudo Random Function (PRF). A PRF takes a seed and some other values (i.e. the user ID or the index value of a certain block) as input, and produces a fixed length pseudorandom string as an output [27].
3.4.4 TP-DAS Protocol Design

As mentioned in 2.5.5, all existing schemes cannot maintain data confidentiality. Therefore, an efficient and secure protocol is proposed to ensure both data integrity and data confidentiality. To ensure data integrity, the protocol will make use of the Elliptic Curve Cryptography (ECC). For data confidentiality, data files will be encrypted prior uploading them to the cloud by using Pseudo Random Function. Thus, users’ data cannot be leaked to the TPA or malicious parties.

The protocol consists of six algorithms, namely, KeyGen, DataEnc, MetadataGen, ChallengeGen, ProofGen and VerifyProof.

1. **KeyGen**: This algorithm is run by the user to produce a pair of keys (public key (upu) and private key (upr)).
2. **DataEnc**: This algorithm is also run by the user to encrypt the file (F) before uploading it to the CSP.
3. **MetadataGen**: This algorithm is also run by the user to pre-process the encrypted data file and generate a verification metadata that will be sent to the TPA for auditing purposes.
4. **ChallengeGen**: This algorithm is run by the TPA to send a verification challenge to the CSP.
5. **ProofGen**: This algorithm is run by the CSP to build a proof of the correctness of the data and then send it to the TPA.
6. **VerifyProof**: This algorithm is run by the TPA to determine whether the data has been tampered or not based on the received proof.

![Figure 3.3: Algorithms used in the protocol](image)

The protocol can be divided into three main phases, namely, **Initialisation, Verification and Dynamic Data Operations**.
3.4.4.1 Initialisation Phase

This phase is for pre-processing the data file (F) before uploading it to the CSP. The user first needs to run the KeyGen algorithm that takes a security parameter (k) as input and generates a pair of keys (public key (upu) and private key (upr)). The key generation process will be as follows:

User has to choose two large primes of size k (p and q), and then compute n = pq. The order of elliptic curve (Nn) is computed as Nn = lcm (p+1,q+1). P is computed as a generator of Nn. The private key = Nn and the public key = {n,P}. The public key will be sent to the TPA for auditing purposes, while the private key will be kept secret at the user’s side.

After generating the keys, the file F will be divided into a number of blocks (n) as follows:

\[ F = \{m_1, m_2, m_3, \ldots, m_n\}, \text{ where } m_i \text{ is the } i\text{th block} \]

The user then runs the DataEnc algorithm to encrypt the file blocks to maintain the confidentiality of the data. The encryption process can be done using a keyed Pseudo Random Function (fk) with a randomly selected parameter (rp). Each file block (mi) will be encrypted as follows:

\[ M_i = m_i + f_k(rp), \text{ where } M_i \text{ is the encrypted } i\text{th block} \]

So the encrypted file will be as follows: \[ F' = \{M_1, M_2, M_3, \ldots, M_n\} \]

After that, the user has to pre-process the F’ and generate its verification metadata (D) using the MetadataGen algorithm. This can be done as follows:

\[ D = \{d_1, d_2, d_3, \ldots, d_n\} \text{ where } d_i \text{ is the metadata for the } i\text{th block.} \]

The metadata for the ith block (di) will be computed using the public key, private key and the encrypted block (Mi), which is as follows:

\[ d_i = M_i P \pmod{N_n} \]

The encrypted file (F’) will be sent to the CSP, while the verification metadata (D) will be sent to the TPA for auditing purposes.
3.4.4.2 Verification Phase

This phase is for auditing users’ data. The TPA runs the ChallengeGen algorithm to issue a verification challenge that will be sent to the CSP. The challenge generation will be as follows:

The TPA selects a random key (k_{PRF}) and a random integer (c) and then compute:

\[ Q = cP \]

The challenge that will be sent to the CSP is:

\[ \text{Challenge} = \{\text{file name} \| k_{PRF} \| Q\}. \]

Upon receiving the challenge, the CSP has to respond to the challenge by using the ProofGen algorithm. The CSP will generate a proof (R) and then send it to the TPA. The generation of the proof will be as follows:

The CSP will first generate random numbers (equal to the number of the data blocks) using a keyed Pseudo Random Function (f_{k_{PRF}}), which is as follows:

\[ x_i = f_{k_{PRF}}(i) \text{ for } i \in [1,n] \]
The proof (R) then can be computed using the encrypted block (Mi) and the challenge, which is as follows:

\[
R = \sum_{i=1}^{n} x_i M_i Q \mod n \\
= \sum_{i=1}^{n} x_i M_i cP \mod n \\
= c\left(\sum_{i=1}^{n} x_i M_i P \mod n\right)
\]

After the TPA has received the proof (R), the TPA will compute R’ and then compare it with the received R by using the VerifyProof algorithm. To compute R’, the TPA has to generate random numbers as the CSP did.

\[x_i = f_{k_{PRF}}(i) \text{ for } i \in [1,n]\]

After that, it uses the public key and the verification metadata (di) to compute

\[
Z = \prod_{i=1}^{n} x_i d_i \mod n \\
R' = cZ \mod n
\]

If R = R’, the data is origin (not tampered). Finally, the verification result will be sent to the user.

Figure 3.5: The verification phase protocol
3.4.4.3 Dynamic Data Operations phase

This phase is for dynamic data operations such as insertion, modification and deletion. Users can add new blocks to their file, modify or delete existing blocks. Users have to send an update request to the CSP by specifying the type of the operation (i.e. insertion, deletion or modification). Upon receiving the request, the CSP will perform the update operation.

To insert a new block \((M_x)\) before \(M_i\) in the file, where \(1 \leq i \leq n\), the user has to perform the following tasks:

- Create a new block \(m_x\)
- Encrypt \(m_x\) using \(DataEnc\) algorithm
  \[ M_x = m_x + f_k(rp) \]
- Compute the metadata for \(M_x\) using \(MetadataGen\) algorithm
  \[ d_x = M_x P(mod\ Nn) \]
- Send an insertion request to the CSP including \((filename|| insertion || i || M_x)\)
- Send an insertion request to the TPA including \((filename|| insertion || i || d_x)\)

Upon receiving the request, the CSP will add \(M_x\) before \(M_i\) and shift the following blocks one step backward. Similarly, the TPA will also add \(d_x\) before \(d_i\).

To modify an existing block \((M_i)\) with a new block \((M_x)\), where \(1 \leq i \leq n\), the user has to perform the following tasks:

- Create a new block \(m_x\)
- Encrypt \(m_x\) using \(DataEnc\) algorithm
  \[ M_x = m_x + f_k(rp) \]
- Compute the metadata for \(M_x\) using \(MetadataGen\) algorithm
  \[ d_x = M_x P(mod\ Nn) \]
- Send a modification request to the CSP including \((filename|| modification || i || M_x)\)
- Send a modification request to the TPA including \((filename|| modification || i || d_x)\)

Upon receiving the request, the CSP will replace \(M_i\) with \(M_x\). At the TPA side, the TPA will also replace \(d_i\) with \(d_x\).

To delete an existing block \((M_i)\), where \(1 \leq i \leq n\), the user has to perform the following tasks:

- Send a deletion request to the CSP including \((filename|| deletion || i)\)
- Send a deletion request to the TPA including \((filename|| deletion|| i)\)
Upon receiving the request, the CSP will delete $M_i$ and shift the following blocks one step forward. Similarly, the TPA will also delete $d_i$.

![Diagram showing the dynamic data operations phase protocol](image)

**Figure 3.6: The dynamic data operations phase protocol**

### 3.5 Chapter Summary

This chapter describes the preliminary design of the proposed system. The system is designed using Elliptic Curve Cryptography and Pseudo Random Function to achieve data integrity and data confidentiality. Several design requirements have been taken into account during the design of the system. The system includes three parties, namely, user, CSP and TPA. The system is divided into three phases. Initialisation phase is to generate the user’s public and private keys, encrypt the file and generate its verification metadata. Verification phase is to audit the user’s data. The last phase is for updating the user’s data when required.
Chapter 4. Methodologies and Project Plan

4.1 Chapter Overview

This chapter is to provide a plan for the next stage of the project that will be held in the summer. It starts with the selection of programming languages and development environments for the implementation of the system. Then, a discussion about how the system is going to be tested and evaluated is provided. Finally, the project plan and timeline are mentioned.

4.2 Implementation Methodology

To convert the system design into a working application, the selection of programming languages and development environments is needed.

4.2.1 Selection of Programming Languages

The implementation of the system will make use of Java to build a graphical-based application. The system’s databases will be created using MySQL.

Java is an object-oriented programming language that is based on classes [28]. Java applications could be run on any Java Virtual Machine (JVM). Once the Java code has been compiled, it can run on any java-based platform without the necessity of recompilation. Also, Java has a garbage collector that clears the memory of unused objects [28]. Furthermore, Java has a multitude of libraries that can be used by developers.

MySQL is for creating and managing relational databases. It is widely used due to its simplicity and flexibility [29]. MySQL can handle a multitude of data as it supports millions of data rows.

4.2.2 Selection of Development Environments

There are various Integrated Development Environments (IDEs) for dealing with Java codes. The most common IDEs are NetBeans and Eclipse. During the implementation of the system, NetBeans will be used. NetBeans is an open source platform for developing java applications. NetBeans supports various features such as GUI builder that help developers to design their applications perfectly.

To deal with the management of MySQL, phpMyAdmin will be used. PhpMyAdmin is an open source tool that helps users to build and manage their databases through a web browser without installing any application.
4.3 Testing and Evaluation Methodology

After implementing the system, it is vital to evaluate it against the system requirements discussed in 3.3. Two main evaluation criteria that need to be considered: security and performance.

Regarding security, the system will be tested in terms of data integrity and data confidentiality. More specifically, the system will be tested to figure out whether it can detect data integrity drifts. It will also be tested to assure that the CSP cannot use previous proofs or build a proof of the correctness of data without actually storing it. In addition, it will be tested against data leakage to the TPA during the auditing process. Furthermore, it will be tested to confirm that malicious parties are not able to access users’ data while it is in the cloud.

Regarding performance, the system will be tested in terms of communication, computation and storage costs. This would measure how much storage is required by the CSP and the TPA for storing the encrypted file and the verification metadata respectively. Also, the cost of storing the keys at the user’s side will be measured. In terms of communication, this includes how many messages are sent between the TPA and the CSP during the auditing process and how much time is required for sending these messages. It also includes the time required by the user to send the encrypted file and the verification metadata to the CSP and the TPA respectively. In terms of computation, this embraces the computational complexity of the used algorithms. The system will be compared with existing systems in terms of the computational time required by each entity in the system. For example, a comparison between this system that uses Elliptic Curve Cryptography (ECC) and other systems that use RSA will be conducted.

4.4 Project Plan

During the period from March to May, a background reading about cloud computing and the issue of data integrity have been done. Then, current data integrity checking solutions have been studied and critically analysed. From this analysis, new system requirements such as data confidentiality have been taken into account for the proposed system. Having gathered the requirements, these requirements have been discussed with the supervisor. Then, the system has been successfully designed. Finally, a plan of how the system is going to be implemented, tested and evaluated has been stated.

During the period from June to September, the system design will be converted into a working application. After that, it will be tested and evaluated as discussed in section 4.3.
The timeline for the project plan is explained in the Gantt chart. The green colour represents the work that has been done so far, while the red colour represents the work that will be done in the summer.

![Gantt chart](image)

**Figure 4.1: Project plan**

### 4.5 Chapter Summary

This chapter provides a plan for the next work in the summer. It discusses how the system is going to be implemented, followed by the way that the system is going to be tested and evaluated. Finally, the project plan is provided.
References


