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

A dissertation submitted to the University of Manchester for the degree of Master in Advanced Computer Science in the Faculty of Engineering and Physical Sciences

2015

Maher Alharby

School of Computer Science

Table of Contents

Final Word Count: 19422

List of Figures

List of Tables

List of Equations

List of Abbreviations

CSP	Cloud Service Provider
DPDP	Dynamic Provable Data Possession
ECC	Elliptic Curve Cryptography
ECDLP	Elliptic Curve Discrete Logarithm Problem
GUI	Graphical User Interface
IDEs	Integrated Development Environments
JSP	JavaServer Pages
JVM	Java Virtual Machine
MAC	Message Authentication Code
MB	Megabyte
PDP	Provable Data Possession
PKC	Public Key Cryptography
PoR	Proof of Retrievability
PP-TPA	Privacy-Preserving Third Party Auditor
PRF	Pseudo Random Function
SQL	Structured Query Language
TPA	Third Party Auditor
TP-DAS	Third-Party based Data Auditing Service
TTPA	Trusted Third Party Auditor

Abstract

Cloud computing has emerged as a means of providing 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 relieved of the burden of storing data locally, and the maintenance costs associated with this. However, as data is shifted into the cloud, users will lose control of their data. Thus, data integrity and data confidentiality are the biggest challenges facing users as the data might be altered, deleted, or even accessed by unauthorised entities while it is in the cloud. The aim of this project is, therefore, to design and implement a secure and efficient method to ensure the integrity and the confidentiality of remote data.

In this project, a comprehensive literature review of related data integrity checking solutions has been conducted to identify the advantages and the drawbacks of each solution. From this literature review, it was obvious that none of these solutions can preserve the confidentiality of the data. Therefore, we have proposed a new solution called Third-Party based Data Auditing Service (TP-DAS) that can achieve both data integrity and data confidentiality. The protocol of the proposed solution has been designed using two cryptographic primitives, namely, Elliptic Curve Cryptography and Pseudo Random Function.

The proposed solution supports public auditability as it enables a Third Party Auditor (TPA) to check the integrity of the data on behalf of the users. The proposed solution has been tested and evaluated to assure its security and efficiency. Compare with related solutions, the proposed solution has a better performance in terms of the computation costs.

Declaration

No portion of the work referred to in the dissertation has been submitted in support of an application for another degree or qualification of this or any other university or other institute of learning.

Intellectual Property Statement

i. The author of this dissertation (including any appendices and/or schedules to this dissertation) owns certain copyright or related rights in it (the "Copyright") and s/he has given The University of Manchester certain rights to use such Copyright, including for administrative purposes.

ii. Copies of this dissertation, either in full or in extracts and whether in hard or electronic copy, may be made only in accordance with the Copyright, Designs and Patents Act 1988 (as amended) and regulations issued under it or, where appropriate, in accordance with licensing agreements which the University has entered into. This page must form part of any such copies made.

iii. The ownership of certain Copyright, patents, designs, trade marks and other intellectual property (the "Intellectual Property") and any reproductions of copyright works in the dissertation, for example graphs and tables ("Reproductions"), which may be described in this dissertation, may not be owned by the author and may be owned by third parties. Such Intellectual Property and Reproductions cannot and must not be made available for use without the prior written permission of the owner(s) of the relevant Intellectual Property and/or Reproductions.

iv. Further information on the conditions under which disclosure, publication and commercialisation of this dissertation, the Copyright and any Intellectual Property and/or Reproductions described in it may take place is available in the University IP Policy (see http://documents.manchester.ac.uk/display.aspx?DocID=487), in any relevant Dissertation restriction declarations deposited in the University Library, The University Library's regulations (see http://www.manchester.ac.uk/library/aboutus/regulations) and in The University's Guidance for the Presentation of Dissertations.

Acknowledgments

The most gratitude of this dissertation is to Allah, and then to my parents, for their encouragement throughout my studies. I would also like to thank my dear wife, Tagreed, for her encouragement, moral support and patience throughout my period of study. My sincere acknowledgment is to my supervisor, Dr. Ning Zhang, for her excellent supervision and beneficial advice. Her efforts and advice during my project's period are highly appreciated.

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 ultimate control over their data [5]. Therefore, one of the biggest challenges concerning data users is how to ensure the integrity and the confidentiality 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 on the Cloud Service Providers' (CSP's) side. Secondly, the CSP could hide data loss incidents from data users in order to maintain their reputations. Thirdly, the CSP may deliberately delete rarely accessed data for saving more storage capacity [3]. Another reason is that the CSP or unauthorised entities might access the content of users' data.

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 on 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 on the user'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, a secure and 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 confidentiality 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 prohibiting unauthorised 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 in 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.

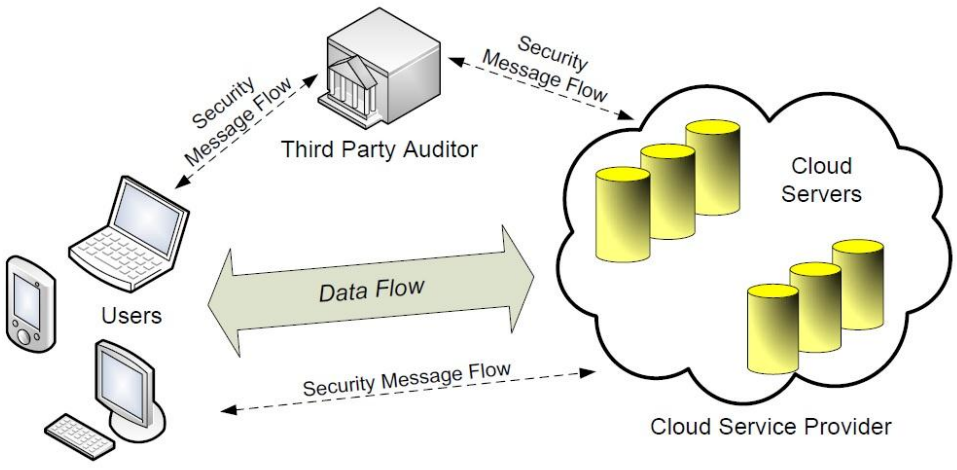


Figure 1.1: System Structure [5]

There are two main objectives that are needed to achieve the main aim of this project, which are as follows:

- Propose an efficient protocol for checking the integrity of remote data files. To ensure efficiency, the protocol must have the following features:
    1. Detect any data integrity drift such as data loss or data alteration.
    2. Support public auditability by relying on a trusted third party auditor (TPA) for checking the data integrity on behalf of the users.
    3. Support dynamic data operations such as data insertion, data alteration, and data deletion.
- Extend the data integrity checking protocol to support data confidentiality by employing encryption techniques. To ensure the confidentiality of the data, the protocol must meet the following requirements:
    1. Prohibit the CSP from accessing the content of users' data files or even learning any knowledge regarding those files.
    2. Prohibit the TPA from accessing the content of users' data files during the auditing task.
    3. Prohibit malicious parties or attackers from accessing the content of users' data files while they are in the cloud.

Several specific tasks need to be done in order to end up successfully with the entire system. These tasks 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 analyse different data integrity checking 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 confidentiality of users' data.
- Create a web-based application that allows users to perform the following tasks:
    1. Upload their data files to the cloud.
    2. View or download their remote data files at any time.
    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 dissertation report will cover five other chapters. Chapter 2 provides an overview about cloud computing and its security issues (e.g., data integrity). Then, a critical analysis regarding the current data integrity checking methods and schemes is provided. Chapter 3 describes the design of the proposed system. The system's architecture and requirements are provided in this chapter in addition to the protocol design. Chapter 4 focuses on the implementation of the proposed system by converting the system's design into a web-based application. Chapter 5 tests and evaluates the proposed system in terms of security and performance. Chapter 6 provides a summary of the whole project. It also states the main contributions, and suggests some ideas for future work.

Chapter 2. Background and Literature Review

2.1 Chapter Overview

This chapter provides a brief introduction to cloud computing and its security issues, followed by a discussion about 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 discussions, a vision to a new solution that can address unsolved issues is provided.

The structure of this chapter is as follows. Section 2.2 introduces the overview concept of cloud computing. Section 2.3 discusses the security issues of cloud computing. Section 2.4 discusses different cryptographic methods for detecting data integrity drifts. Section 2.5 discusses existing data integrity checking schemes and their drawbacks. Section 2.6 summarises the chapter.

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) [19].

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 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 not tampered).

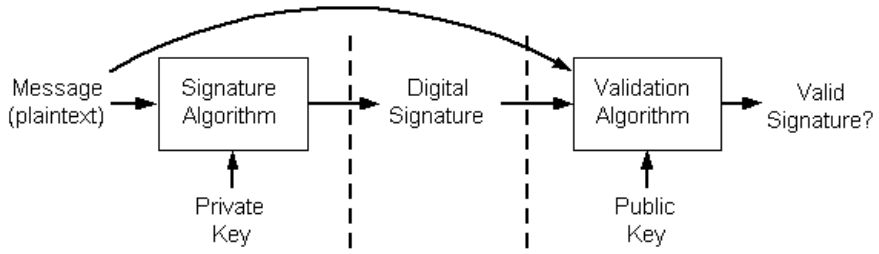


Figure 2.1: Digital Signature Method [23]

However, to prevent the verifier from accessing the content of the data during the verification process, the data can be encrypted or hashed prior to 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 to generating the signature.

Drawbacks:

- This method is based on asymmetric cryptography as it requires two keys, which are public and private keys. Thus, it is not efficient for encrypting large messages.

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 in its original state.

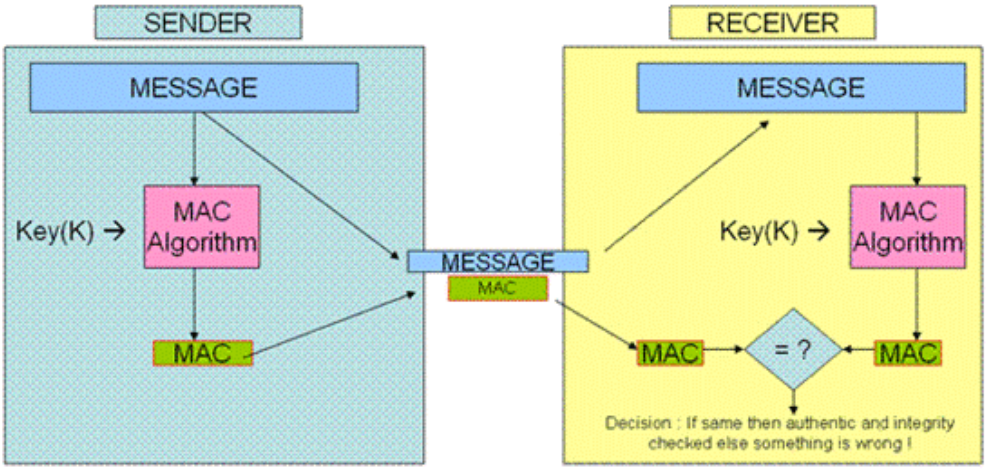


Figure 2.2: Message Authentication Code (MAC) Method [24]

Advantages:

- This method uses a single secret key to generate and verify MACs. Thus, this method is suitable for encrypting large files.

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]

To verifier 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 not tampered.

Advantages:

- 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, when all keys are 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 not tampered. Thus, this scheme provides a probabilistic assurance for remote data because the verifier checks the integrity of a certain number of data blocks instead of all data blocks.

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 to 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) can then 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 on the verifier's side (e.g., 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 (e.g., 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 on 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 on 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 on the CSP side.

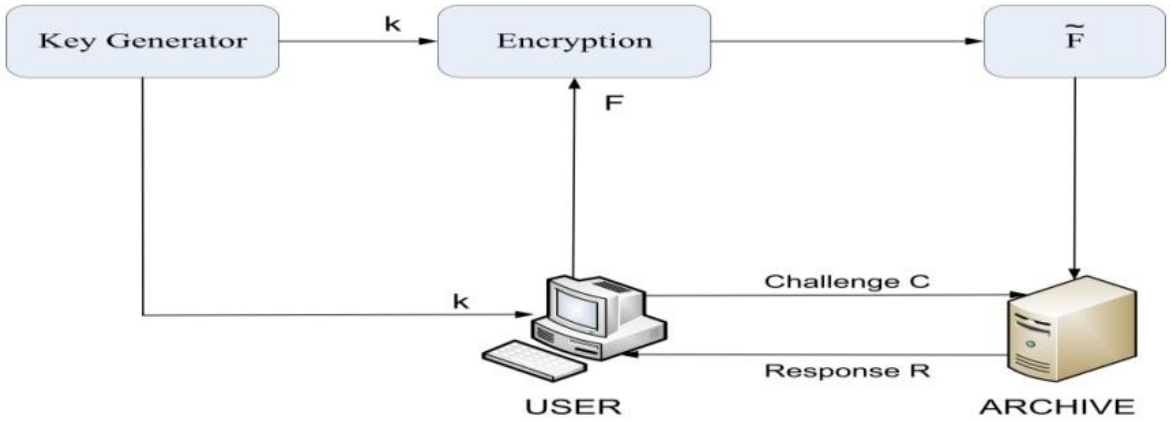


Figure 2.4: Sentinel-based Proof of Retrievability [10]

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 on the users' side.

2.5.2.3 Compact Proof of Retrievasbility

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 private auditability and it is based on homomorphic authenticators constructed from Pseudo Random Function (PRF). 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]. In this scheme, the encoded file (F) is divided into a number of data blocks $m_1, m_2, \ldots, m_n$. The user generates a private key, which is a combination of a random number and a key for the PRF function. After

generating the secret key, the user generates verification metadata for each data block using the secret key. The data blocks {mi} and the verification metadata {Ti} are then sent to the CSP. To check the integrity of the data, the verifier generates a verification challenge that will be sent to the CSP. The challenge specifies a set of data blocks, which is chosen at random, to be audited. Upon receiving the challenge, the CSP will compute a proof from the specified data blocks and then send it to the verifier. Having received the proof, the verifier can check the correctness of the received proof by comparing it with the stored metadata.

Their second 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]. In this scheme, the user generates two keys, which are public and private keys. The user then generates verification metadata for each data block using the BLS hash function, private key, and the public key. The data blocks {mi} and the verification metadata {Ti} are then sent to the CSP. To check the integrity of the data, the verifier generates a verification challenge that will be sent to the CSP. The challenge specifies a set of data blocks, which is chosen at random, to be audited. Upon receiving the challenge, the CSP will compute a proof from the specified data blocks and then send it to the verifier. Having received the proof, the verifier can verify the proof using the stored metadata and the public key.

However, Shacham and Waters did not consider dynamic data operations in their schemes. If the user wants to update an existing file, they need to retrieve that file from the CSP and then use the private key to generate new verification metadata. Thus, this may result in more communication overhead as the file needs to be retrieved in each update process.

Advantages:

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

Drawbacks:

- These schemes cannot efficiently support dynamic data operations. This is because the user has to retrieve the file that needs to be updated and then generate new metadata as discussed above.

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 to storing it in the CSP. The user splits the file F into a number of blocks, {block1, block2, block3,…, block$_n$},where n is the total number of the file blocks. After that, the user has to generate unique metadata (called tag) for each block. The generated tags will be stored on 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 can then verify the response by using its local tags. If the returned tags match the stored ones, the data file F is not tampered.

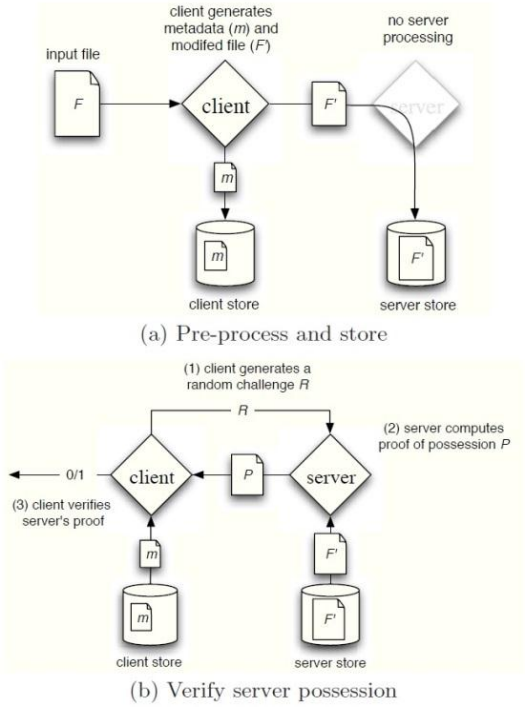


Figure 2.5: Provable Data Possession protocol [12]

Advantages:

- This scheme supports public auditability because everyone who has the tags can audit the data.
- This scheme allows users to append new blocks (e.g., a blockX) to their files. Users have consequently to generate a new tag for the new block (e.g., 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. This scheme consists of six algorithms, which are as follows:

1- KeyGen: this algorithm is run by the user to establish the secret and public keys. The public key will be sent to the CSP, while the secret key will be kept on the user's side.
2- RequestUpdate: this algorithm is also run by the user to generate an update request and then send it to the CSP. This algorithm takes the file F with the update operation required (e.g., insert a new block before block i, modify block i, or delete block i) and the verification metadata, and produces an update request that includes the file's name, the update information and new verification metadata. The request will then be sent to the CSP.
3- PerformUpdate: this algorithm is run by the CSP to respond to the user's update request. This algorithm performs the update operation stated in the request (e.g., modify an existing block) and then produces a new version of the user's file. The corresponding verification metadata will then be replaced with the new metadata received from the user.
4- Challenge: this algorithm is run by the user to assure that the CSP has done the update request successfully. This algorithm takes the secret key and the latest version of the verification metadata, and produces a challenge c. The challenge c will then be sent to the CSP.
5- Proof: this algorithm is run by the CSP to generate a proof of performing the right update process. This algorithm takes the newest version of the file and the verification metadata, and produces a proof (R) that will be sent to the user.
6- Verify: this algorithm is run by the user to verify the proof generated by the CSP. This algorithm takes the user's metadata, the challenge (c), and the proof (R), and outputs 1 if the update request has been done successfully or 0 otherwise.

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 on 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.

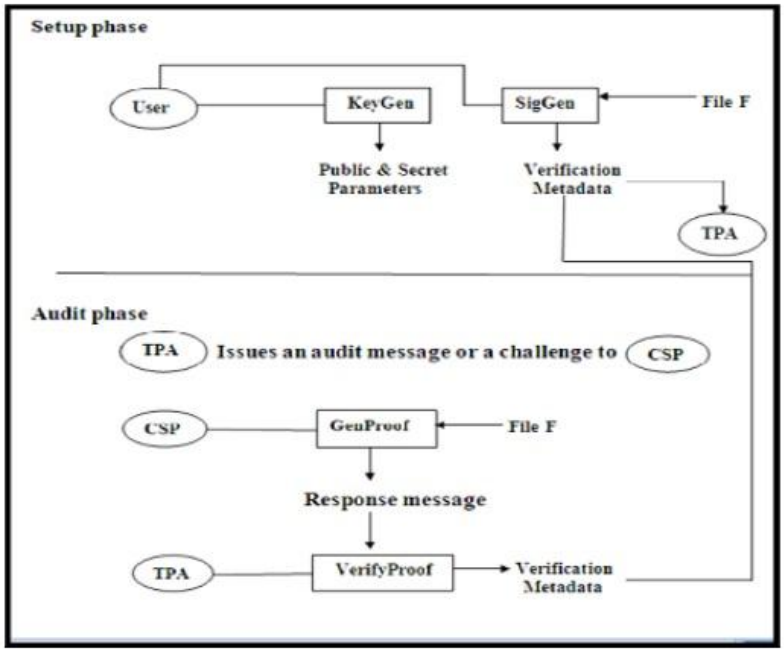


Figure 2.6: Public auditing scheme using Third Party Auditor (TPA) [17]

Advantages:

- This scheme reduces the cost of computational resources on 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:

Schemes / Features	Basic schemes		POR schemes			PDP schemes		TPA	
	MAC	**Digital signature**	**Basic POR**	**Sentinel-based**	**Compact POR**	**Basic PDP**	**DPDP**	**TTPA**	**PP-TPA**
Unbounded number of data auditing attempts	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Auditing without retrieving data	Yes	No	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Dynamic data operations	No	No	No	No	No	Partial (appended only)	Yes	Yes	Yes
Public auditability	No	No	No	No	No	Yes	Yes	Yes	Yes
The use of TPA	No	No	No	No	No	No	No	Yes	Yes
Privacy preserving	No	No	No	No	No	No	No	No	Yes
Confidentiality	No	No	No	No	No	No	No	No	No

Table 2.1: A comparison between different remote data integrity checking schemes.

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 to uploading it to the CSP.

2.6 Chapter Summary

In this chapter, we started with a brief overview of cloud computing and its security issues, followed by different methods that can be used to detect data integrity drifts. Then, the current data integrity checking schemes have been discussed to highlight their advantages and drawbacks. From this discussion, it was obvious that these schemes cannot protect the confidentiality of the users' data. Therefore, an insight into the proposed solution, which will be explained in chapter 3, has been included.

Chapter 3. Third-Party based Data Auditing Service (TP-DAS) Design

3.1 Chapter Overview

This chapter provides the design of the proposed system. The system is 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 that need to be met in the proposed system. After that, a description of the system's design is provided embracing the system's architecture, assumptions, and notations in addition to the protocol design.

The structure of this chapter is as follows. Section 3.2 provides an overview model of the TP-DAS system. Section 3.3 presents the design requirements of the proposed system. Section 3.4 describes the design of the proposed system. Section 3.5 summarises the chapter.

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 and send it 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.

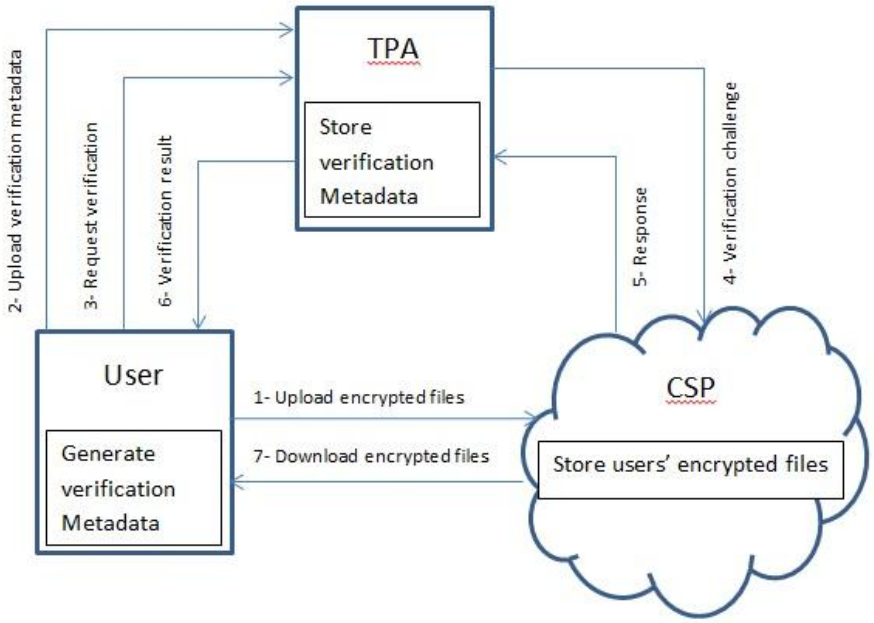


Figure 3.1: TP-DAS model overview

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 (e.g., a malicious CSP) from accessing the content of 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 reduce the computation costs required by each entity in the system.
5- The system should allow unbounded number of data auditing attempts.
6- The system should support dynamic data operations, e.g., 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 web-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 the 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 uses the user's encrypted file to generate the proof. The generated proof will then be sent to the TPA. After that, the TPA uses the stored verification metadata to verify the CSP's proof. Finally, the verification result will be sent to the user.

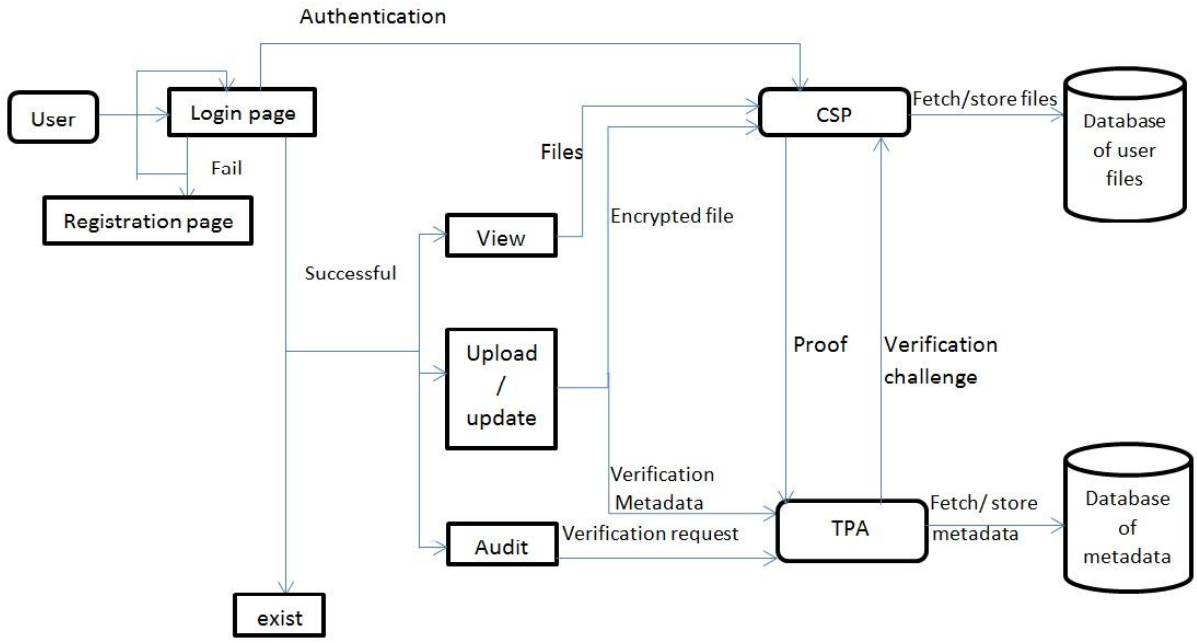


Figure 3.2: System architecture

3.4.2 Assumptions

There are different 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.

Notation	Description
User	Who has data to be stored in the cloud.
CSP	The Cloud Service Provider.
TPA	Third Party Auditor.
F, F’	The user’s file, the encrypted file.
D	The set of verification metadata that used for auditing purposes.
m_i, M_i	The *i*th data block as the file is divided into a number of blocks, the encrypted *i*th block.
D_i	The verification metadata for the ith block.
U_{pu}, U_{pr}	The user’s public key, the user’s private key.
f_k (.)	A keyed Pseudo Random Function (PRF).
$\sum_{i=1}^{n} x_i$	$x_1 + x_2 + x_3 + \ldots + x_n$
$\prod_{i=1}^{n} x_i$	$x_1 * x_2 * x_3 * \ldots * x_n$

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 Z_n. Let assume n is an integer. Two integers in Z_n (a and b) are chosen to satisfy $gcd(4a^3+27b^2, n)=1$. En(a,b) can be defined as the set of pairs $(x,y) \in (Zn)^2$ 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 (e.g., 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, none of the related solutions can 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 using Pseudo Random Function. Thus, the content of 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 (U_{pu}) and private key (U_{pr})).
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 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 proof received.

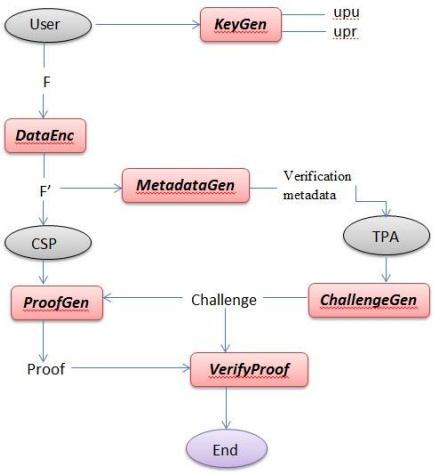


Figure 3.3: Algorithms used in the protocol

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 (U_{pu}) and private key (U_{pr})). The key generation process will be as follows.

The user has to choose two large primes of size k (p and q), and then compute n = pq. The order of elliptic curve (N_n) is computed as Nn= lcm (p+1,q+1). P is computed as a generator of N_n.

The private key = N_n 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 on the user's side.

After generating the keys, the file F will be divided into a number of data 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 (f_k) with a randomly selected parameter (r). Each file block (m_i) will be encrypted as follows.

$$M_i = m_i + f_k(r), \text{ where } M_i \text{ is the encrypted ith block}$$

Equation 3.1: Data block encryption

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

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.

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

The metadata for the i*th* block (d_i) will be computed using the public key, private key, and the encrypted block (M_i), which is as follows.

$$d_i = M_i P (\text{mod } N_n)$$

Equation 3.2: Metadata generation for data block

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

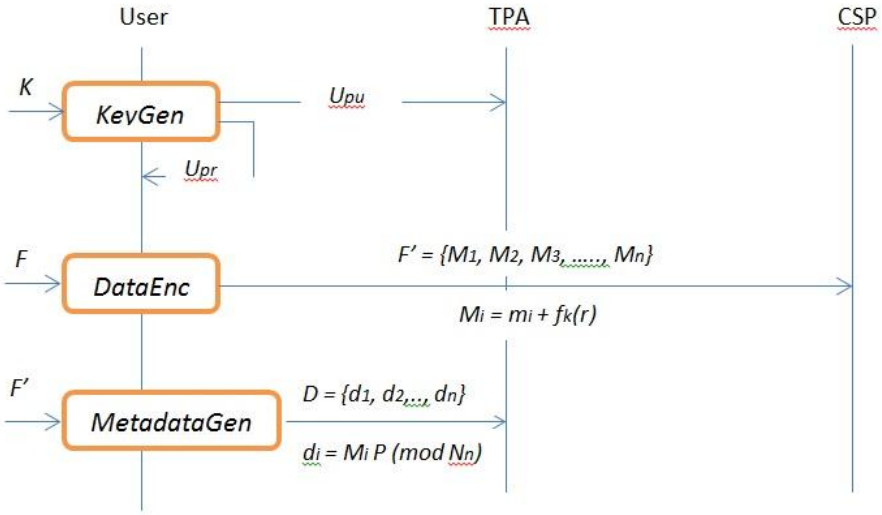


Figure 3.4: The initialisation phase protocol

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 $Challenge = \{file\ name \,||\, k_{PRF} \,||\, Q\}$.

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 (fk_{PRF}), which is as follows.

$$xi = fk_{PRF}(i) \text{ for } i \in [1,n]$$

The proof (R) can then be computed using the encrypted block (M_i) and the challenge, which is as follows.

$$R = \sum_{i=1}^{n} xi\ Mi\ Q \bmod n$$

$$= \sum_{i=1}^{n} xi\ Mi\ cP \bmod n$$

$$= c\left(\sum_{i=1}^{n} xi\ Mi\ P \bmod n\right)$$

Equation 3.3: Proof generation

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.

$$xi = fk_{PRF}\,(i)\ for\ i \in [1,n]$$

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

$$Z = \prod_{i=1}^{n} xi\ di\ mod\ n$$

$$R' = cZ\ mod\ n$$

Equation 3.4: Proof verification

If R = R', the data is in its original state (not tampered). Finally, the verification result will be sent to the user.

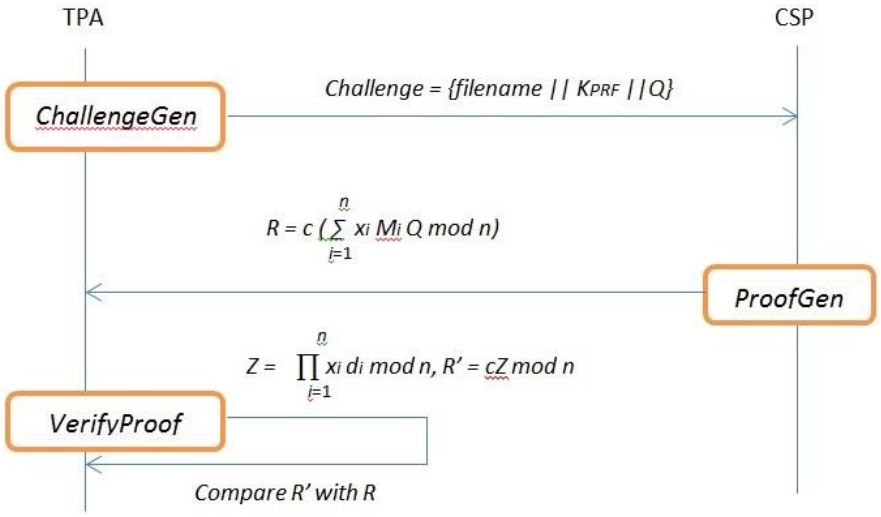


Figure 3.5: The verification phase protocol

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 existing blocks, or delete existing blocks. Users have to send an update request to the CSP by specifying the type of the operation (e.g., 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

  $M_x = m_x + f_k(r)$
- Compute the metadata for M_x using *MetadataGen* algorithm

  $d_x = M_x P \pmod{N_n}$
- 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

  $M_x = m_x + f_k(r)$
- Compute the metadata for M_x using MetadataGen algorithm

  $d_x = M_x P \pmod{N_n}$
- 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. On the TPA's 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.

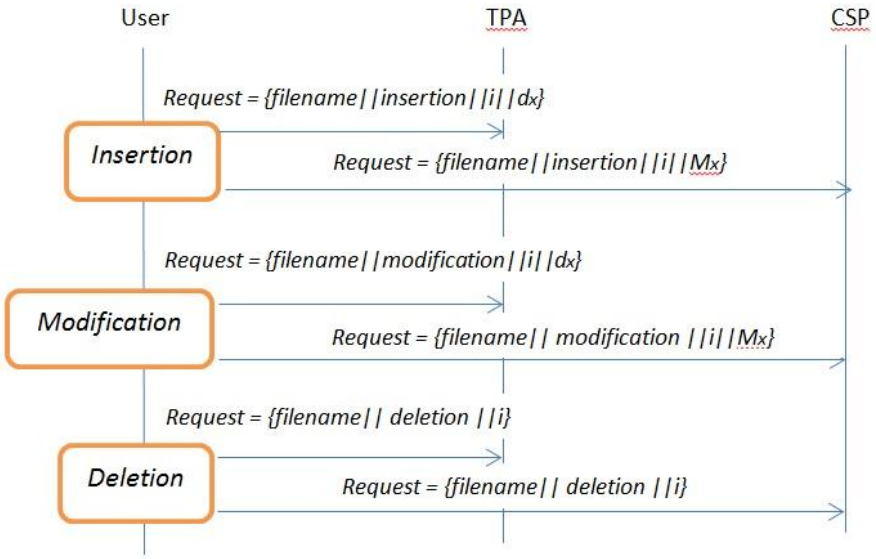


Figure 3.6: The dynamic data operations phase protocol

3.5 Chapter Summary

In this chapter, we described the design of the proposed system. The system has been designed using Elliptic Curve Cryptography (ECC) and Pseudo Random Function (PRF) 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 has been 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. TP-DAS Implementation

4.1 Chapter Overview

This chapter discusses the implementation of the TP-DAS system. Implementation development environments and programming languages, which will be used for converting the system's design into a working web application, are discussed. This chapter also discusses the implementation of each entity in the system, after which some difficulties encountered during the implementation will be provided.

The structure of this chapter is as follows. Section 4.2 discusses the Integrated Development Environments (IDEs) and programming languages used. Section 4.3 describes the implementation of the user, Third Party Auditor (TPA), and Cloud Service Provider (CSP) as well as the implementation of the system's databases. These implementations are illustrated by graphical user interfaces and code snippets. Section 4.4 describes the difficulties faced during the system's implementation. Section 4.5 summarises the chapter.

4.2 Implementation Environments and Programming Languages

To convert the system design into a working web application, the selection of programming languages and Integrated Development Environments (IDEs) is needed. In this section, we will describe the programing languages and the development environments that have been used during the implementation phase. Two programming languages have been used, which are Java and MySQL. Regarding the development environments, NetBeans, Apache Tomcat web server, and MySQL server have been used for development purposes.

4.2.1 Programming Languages

During the implementation phase, the system's design has been translated into a web-based Java application using Java language with JavaServer Pages (JSP) technology. The system's databases have been created using MySQL language.

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 [33]. The syntax of Java is simpler than that for other languages such as C++, because there are no pointers in Java [28]. In addition, Java can help developers execute different processes at the same time [33]. This is referred to as multithreading. 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.

JavaServer Pages (JSP) technology is basically HTML or XML pages that are embedded within Java code. The main purpose of JSP technology is to help developers generate dynamic web pages that contain HTML or XML code and Java code [38]. The JSP code is translated into a Java servlet and this servlet is then executed on a web server [37].

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 Development Environments

Integrated Development Environments (IDEs) can be defined as software applications that can be used by developers to build and develop their code applications. IDEs offer a Graphical User Interface (GUI) to help developers manage and edit their code application easily [34]. IDEs usually contain a debugger, an interpreter, and a compiler. An important feature of modern IDEs is the code completion feature that helps developers to speed up the task of writing their code applications [35].

There are various IDEs for dealing with Java codes. The most common IDEs are NetBeans and Eclipse. During the implementation of the system's application, NetBeans has been used. NetBeans is an open-source platform for developing java applications. NetBeans is compatible with various operating systems such as Windows, Linux, and MAC OS. NetBeans supports all different types of Java applications such as Java ME, Java SE, mobile, and web applications. In addition, it supports various features, such as GUI builder, that help developers to design their code applications perfectly.

For deploying and running JSP technology, the Apache Tomcat web server has been installed and then integrated with NetBeans. The Apache Tomcat has been selected among

other web servers because it is an open-source web server that is compatible with a servlet container. Due to the computability with Java servlet, the Apache Tomcat web server can be used to run JSP code.

For dealing with the management of MySQL, MySQL server 5.6 has been used. MySQL server 5.6 is open-source software that can help developers to build and manage their relational databases using Structured Query Language (SQL). MySQL server 5.6 is developed to improve the performance of the query optimiser [36]. MySQL server 5.6 can be installed and then integrated with NetBeans.

4.3 TP-DAS Implementation

The proposed TP-DAS system consists of three entities, namely, user, Third Party Auditor (TPA), and Cloud Service Provider (CSP). This section will describe the implementation of each entity in the system. This is illustrated by graphical user interfaces and some code snippets. The implementation of the CSP's and the TPA's databases is also embraced in this section. Figure 4.1 shows the home page of the TP-DAS system's web application.

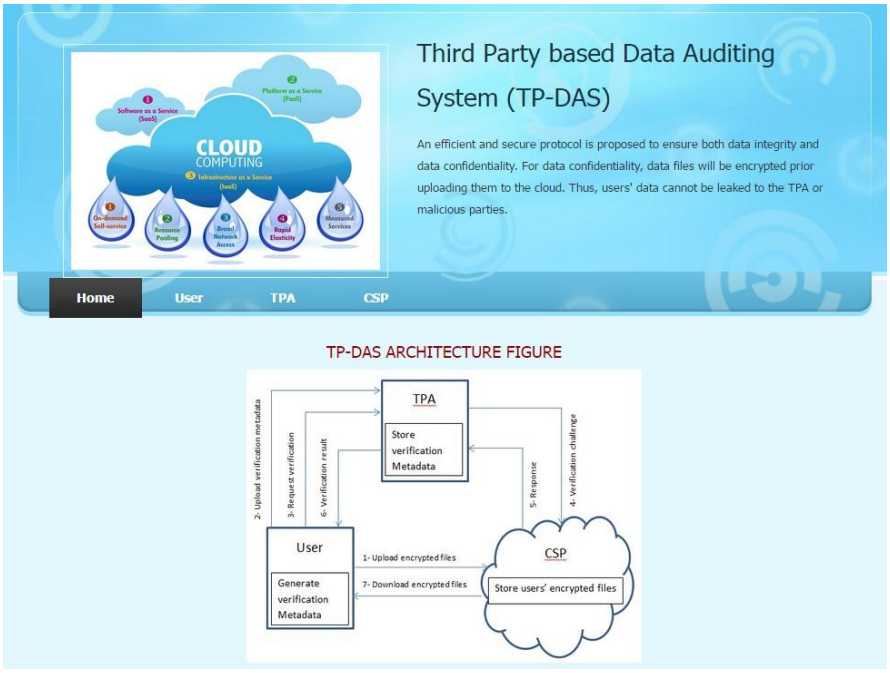


Figure 4.1: The home page of the TP-DAS web application

4.3.1 User Implementation

The user can use the graphical interfaces of the TP-DAS web application to perform the following tasks:

1- Log into the system: users can log into the system through a login page using their name and password. For new users, they can register into the system using the signup page.
2- Download files: after a successful login, the user can show all their files' information and download any file when needed.
3- Upload files: users can also upload new files to be stored in the CSP's storage server.
4- Update files: users can also update their stored files when needed.
5- Send verification requests: users can ask the TPA to check the integrity of their files.

There are five main classes for the user implementation, which are login, registration, Fileupload, fileupdate, and sendReq. Login class is to get the user's login information and then pass it to the CSP for authentication purposes. Registration class is to get the new user's information and then pass it to the CSP to set up a new account for the user. Fileupload class is to get a file from the local storage and then upload it to the CSP for storage purposes. Fileupdate class is to get the modified data blocks from the user and then pass it to the CSP for update purposes. SendReq class is to get a verification request from the user and then pass it to the TPA. Figure 4.2 shows the classes of the user implementation.

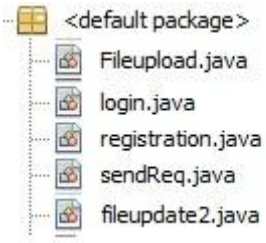
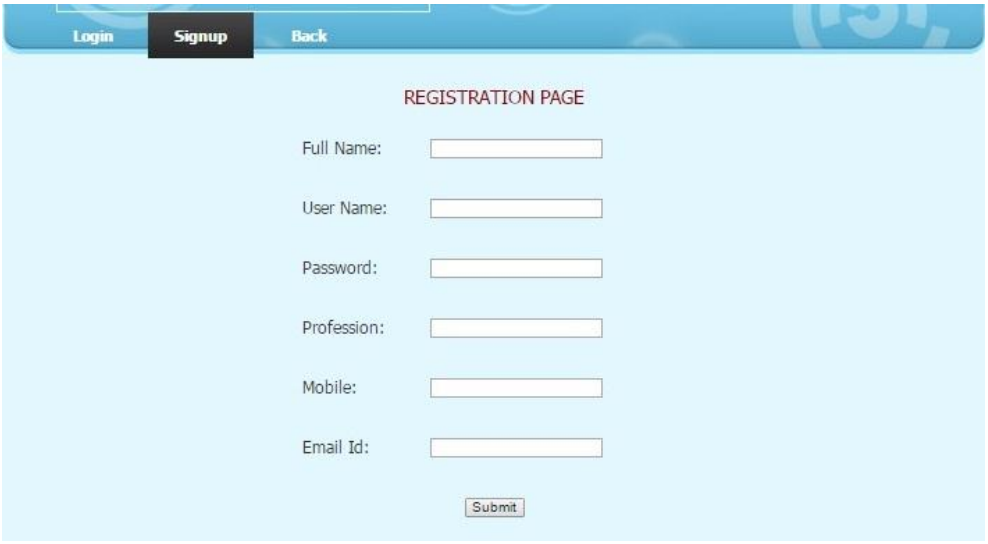


Figure 4.2: Classes of the user implementation

User Signup and Login

To be able to use the TP-DAS web application, new users have to use the signup page to setup new accounts. Figure 4.3 shows the information needed during the user's registration process. After pressing the "Submit" button, the registration information will then be passed to the CSP as string variables to be stored in its database.



4.3: The user signup page

Having set up a new account, the user can log into the system through a login page using their username and password. The username and password will be passed to the CSP to authenticate the identity of the user.

4.4: The user login page

Upon receiving the user's login information, the CSP will execute a SQL query to check if the provided username and password exist in its database. The query is as follows.

rs = st.executeQuery("select * from users where username='"+username+"' && password='"+password+"'");

If the username and password do not match the ones in the CSP's database, a decline message will be shown to the user. Otherwise, the main framework will be loaded. The main framework shows the details of all the user's files such as file id, file name, and file's

uploading date. The user can choose any file to be downloaded. Figure 4.5 shows the main framework of the user implementation.

4.5: The main framework of the user implementation

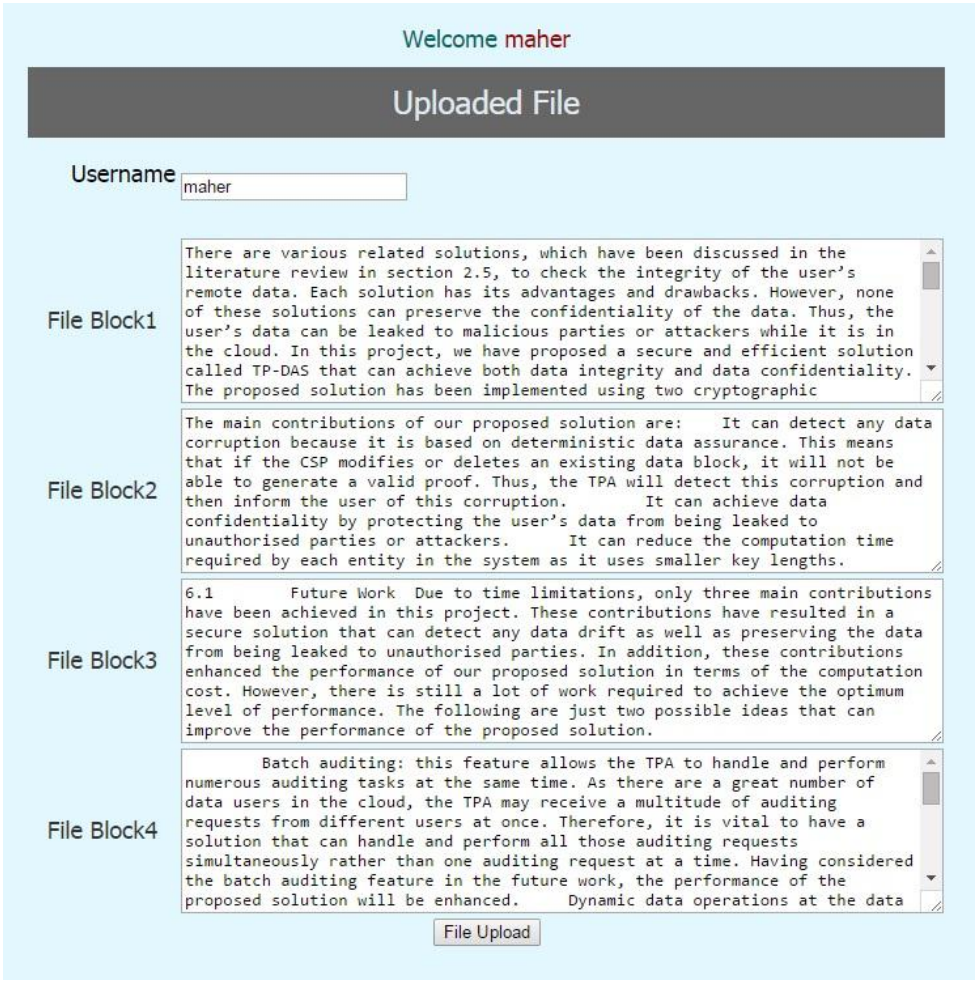
From the main framework, the user can upload new files, update existing files, or send a verification request to the TPA.

File Upload

Users can select a file from their local hard disk to be uploaded to the CSP.

4.6: File upload page

When the user presses the “Upload data” button, the chosen file will be divided into a number of data blocks (n) as discussed in section 3.4.4.1. Figure 4.7 shows the data blocks of the chosen file.



4.7: The data blocks of the file

To split the file into a number of data blocks, we have declared a split function. This function takes the file and the number of data blocks (n) as input, and produces n data blocks. In our implementation, we assumed the number of the data blocks is four. Thus, we have declared four parameters, namely, b2,b3,b4, and b5. Each parameter is to represent a single data block. The following code snippet has been used to split the file into four data blocks.

```
String line1 = reader1.readLine();
 while(line1 != null) {
     line1 = reader1.readLine();
     if(c5<=c3){b2 = b2 + line1;}
     if(c5>c3 && c5<=c4){b3 = b3 + line1;}
     if(c5>c4 && c5<=(c3+c4)){b4 = b4 + line1;}
     if(c5>(c3+c4) && c5<=c2){b5 = b5 + line1;}

     c5++;
 }
```

4.8: Code snippet of the file split function

After splitting the file into four data blocks, the user can press the "File Upload" button shown in Figure 4.7 to upload the file's data blocks to the cloud. Prior to uploading the file's data blocks to the cloud, different functions will work in the background, which are as follows.

1- Key generation function

This function is to generate a pair of elliptic curve keys (secret and public keys). The key generation process has been discussed in section 3.4.4.1. N_n, which is the least common multiple of p and q, is computed as the secret key, while the parameter n and the generator of the elliptic curve (P) are computed as the public key. Figure 4.9 shows a code snippet of the key generation function.

```
// Key Generation
public ECKey(EllipticCurve ec) {
    beta = ec;
    secret = true;

    Random rand = new Random(); // generate a random number
    p = rand.nextInt(1000) + 1;
    q = rand.nextInt(1000) + 1;
    n = p * q;
    Nn= lcm (p+1,q+1);// secret key

    P =(beta.getGenerator());
    P.fastCache();
}

public String toString() {
    if (secret) return ( "Secret key: "+ Nn);
    else return("Public key:"+ P +" "+ n);
}
```

4.9: Code snippet for the key generation function

2- Encryption function

As we discussed in section 3.4.4.1, the file is encrypted prior to uploading it to the cloud. Therefore, an encryption function is implemented to encrypt the users' files. This function takes the file's data blocks and the key as input, and produces the encrypted file. The encrypted file is then transferred to the CSP in byte streams.

Figure 4.10 shows a code snippet of the encryption function. The key for the encryption function is k. A random parameter (r) is used as the seed for the encryption function. The encrypted file is stored as byte streams in an object called out1. The out1 object is then passed to the CSP to be stored in its database.

```
InputStream in = new FileInputStream(storeFile);
OutputStream out1 = new CryptoOutputStream(new FileOutputStream(new File("D:\\enc\\"+fileName)), r, k);
int read;
System.out.print("Encrypting... ");
time = System.currentTimeMillis();

int bytes = 0;
// b1 = "";
while((read = in.read()) != -1) {
    out1.write(read);
//    b2 = b2 + read;
    bytes++;
}
out1.flush();
in.close();
out1.close();

System.out.print("done");
time = System.currentTimeMillis()-time;

System.out.println(" ("+time+" ms) ["+(bytes*1000/time)+" bytes/sec]");
```

4.10: Code snippet of the encryption function

3- Verification metadata generation function

As we discussed in section 3.4.4.1, the verification metadata is computed over the encrypted file. The verification metadata function takes the encrypted file, the secret key, and the public key as input, and produces the verification metadata. The verification metadata is then transferred to the TPA in byte streams.

Figure 4.11 shows a code snippet of the verification metadata function. The object encfile stores the encrypted file. To compute the verification metadata, the verification metadata function needs the generator P, which is a part of the public key, and the secret key (N_n). After computing the verification metadata, the verification metadata is then stored as byte

streams in an object called verification. The verification object is then passed to the TPA to be stored in its database.

```
InputStream encfile = new CryptoInputStream(new FileInputStream(new File("D:\\enc\\"+fileName)), r, k);
OutputStream verification = new CryptoOutputStream(new FileOutputStream(new File("D:\\metadata\\"+fileName)), P, Nn);

System.out.print("Generating verification metadata... ");
time = System.currentTimeMillis();

int byte1 = 0;
// b1 = "";
while((read = encfile.read()) != -1) {
    out1.write(read);
//   b2 = b2 + read;
    byte1++;
}
verification.flush();
encfile.close();
verification.close();

System.out.print("done");
time = System.currentTimeMillis()-time;

System.out.println(" ("+time+" ms) ["+(byte1*1000/time)+" bytes/sec]");
```

4.11: Code snippet of the verification metadata function

File Update

As we discussed in section 3.4.4.3, the proposed system supports dynamic data operations at the block level. Users can update their files by inserting new data blocks, modifying existing blocks, or deleting existing blocks.

When the user chooses a file to be updated, the file will be shown as data blocks. The user can then update any data block and then press the “Update” button shown in Figure 4.12 to send the updated blocks to the CSP.

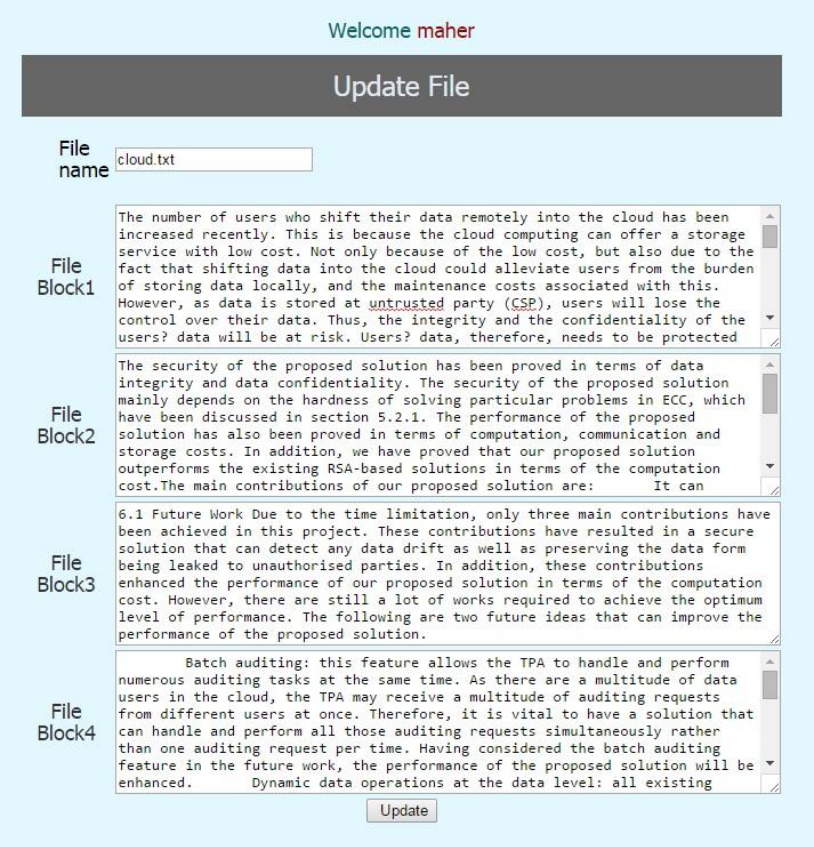


Figure 4.12: Data blocks update

Verification Request

The user can select the file that needs to be verified and then send a verification request to the TPA. Figure 4.13 shows the user interface for sending a verification request.

Figure 4.13: verification request interface

The verification request is a function that has two parameters. The first parameter is the user's id and the second one is the file's name that needs to be verified. Figure 4.14 shows the implementation code of the verification request function.

```
public  String Request (String userid, String filename)
{
     this.filename = filename;
     this.userid = userid;

RequestDispatcher rd = null;
    try {
        rs=st.executeQuery("select * from uploads where ownername='"+userid+"' and filename = '"+ filename +"' ");

    } catch (SQLException ex) {
        Logger.getLogger(sendReq.class.getName()).log(Level.SEVERE, null, ex);
    }
    return userid + " " +  filename;
}
```

Figure 4.14: The verification request function code

4.3.2 TPA Implementation

The Third Party Auditor (TPA) is implemented as a server for storing the verification metadata and checking the integrity of the users' files. There are three main classes for the TPA's implementation, which are storeMetadata, genChallenge, and verifyProof. The class storeMetadata is to receive the verification metadata from the user and then store it in its database. The class genChallenge is to generate a verification challenge and then send it to the CSP. The class verifyProof is to get the proof from the CSP and then verify it.

Receive and Store Verification Metadata

When the user uploads a file to the CSP, they generate verification metadata as discussed in section 4.3.1.2. The metadata is then sent to the TPA for auditing purposes. Upon receiving the verification metadata, the TPA will store it in its storage server, while the metadata details will be stored in its database in the audit's table. The details include file id, file name, owner name, verification metadata's uploading date, public key (pk), and the status of the file. To store the metadata details in the database, the TPA executes the following SQL command.

String sq2 = "insert into audit(filename, ownername,date,pk,status) values('" + hm.get("filename") + "','" + hm.get("ownername") + "', '"+ dateFormat.format(date)+"','"+ pk +"','"+ status+"')";

Generate Challenge

Upon receiving the verification request from the user, the TPA will generate a verification challenge and then send it to the CSP. The TPA's home page shows all the verification requests that have been received from users (see Figure 4.15).

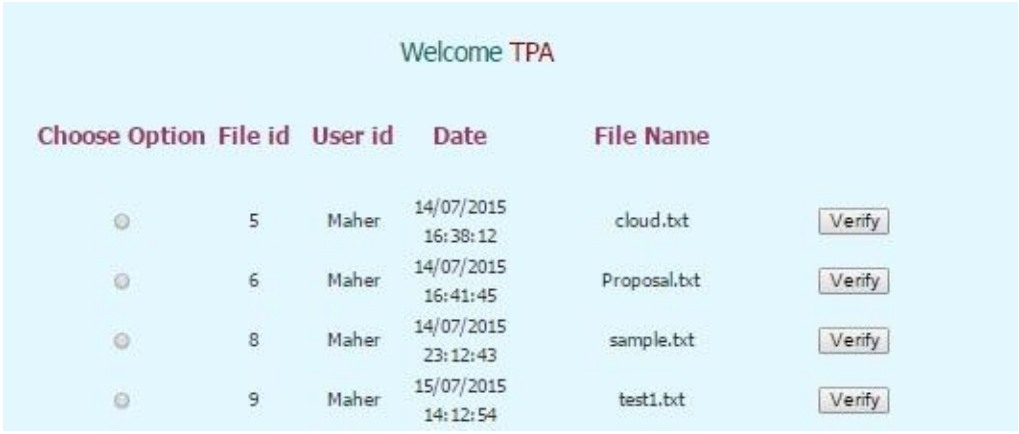


Figure 4.15: Verification requests sent by users

The TPA can choose any file to be verified and then press the "Verify" button to generate a challenge that will be sent to the CSP. The challenge generation process has been discussed in section 3.4.4.2. During the implementation of the challenge generation process, we have declared a function called genechallenge. This function takes a random key, random integer and the public key as inputs, and produces a challenge as output. To generate a random key (k_{PRF}), we have used the KeyGenerator class to generate a random AES key with a key size of 256-bits. To generate a random integer (c), we have used java.util.Random class. The challenge is then computed as the multiplication of P and c plus the random key. Figure 4.16 shows the implementation code of the challenge generation.

```
public String genechallenge () throws NoSuchAlgorithmException, InsecureCurveException
{
    // generate a random key
KeyGenerator keyGen = KeyGenerator.getInstance("AES");
keyGen.init(256); // key size
SecretKey kPRF = keyGen.generateKey();

// generate a random integer
Random c = new Random (System.currentTimeMillis());

//compute Q =cP
EllipticCurve ec = new EllipticCurve (new secp160r1());
P = new ECKey(ec);

ECKey Q = P.multiply (c);// Q= cP

String Challenge = kPRF + "" + Q;

return Challenge;
}
```

Figure 4.16: Code for the challenge generation

Verify Proof

Upon receiving the proof from the CSP, the TPA will verify the proof using the verification metadata as discussed in section 3.4.4.2. During the implementation of the verification process, we have declared a function called verifyproof. This function takes n pseudo random numbers, the metadata for all data blocks (di), and the received proof (R) as input, and produces RR. To generate n random numbers, we have defined an integer ArrayList (x) of size n, where n is the total number of the data blocks. The range of the random numbers is set to be between 0 and 65536. Z is then computed as the sum of the ArrayList (x) and the metadata d[j]. RR is then calculated as the sum of Z and the random parameter (c), which is taken from the challenge. After computing RR, the TPA then compares RR with R. If they are equal, the file is not tampered. The result of the verification process will be stored in a string variable called result. The result will then be sent to the user. Figure 4.17 shows a code snippet of the verification process.

```
public String verifyproof (String R , String filename)
{
    this.R = R;
    this.filename = filename;
    String result;
    ArrayList<Integer> x = new ArrayList<Integer>(n);
    Random rand = new Random();
    rand.setSeed(System.currentTimeMillis());

    for (int i=0; i<n; i++) // Generate n random numbers
     {
    Integer r = rand.nextInt() % 65536;
    x.add(r);
     }

     for (int j=0 ; j<n; j++) // compute R'
    {
       Z +=  mod (x.set(j, n)+ "" + d[j], n) ;
    }
     String c = rand.toString();
     RR = Z + c;

     //Compare the generated R' with the proof (R) received from the CSP
     if (R.equals(RR))
     {
         result = "The file: " + " " + filename + " is origin";
        System.out.print("The verification result is :" + " " + result);
     }
     else
     {
        result = "The file: " + " " + filename + " is not origin";
        System.out.print("The verification result is :" + " " + result);

     }

     return result;
}
```

Figure 4.17: Code snippet for verifying the proof

There are two different possible statuses for users' files, which are verified and not verified. When the TPA has verified a file, the file's status will be verified. Otherwise, the file's status will be considered as not verified. Figure 4.18 shows the status of all the users' files.

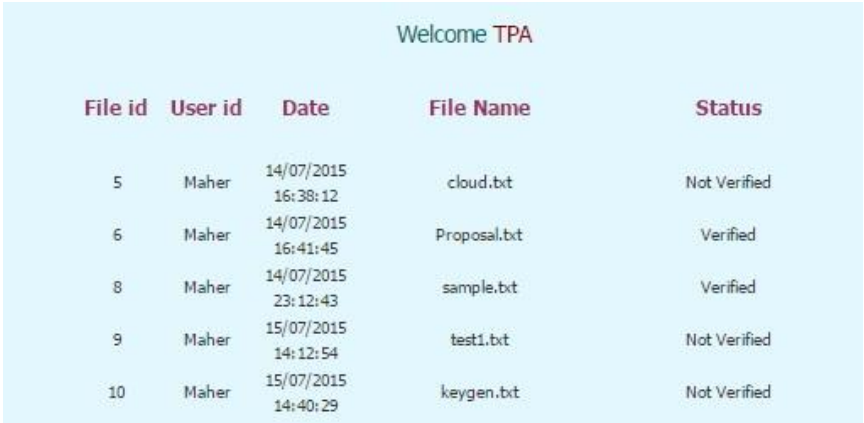


Figure 4.18: The status of all users' files

4.3.3 CSP Implementation

The Cloud Service Provider (CSP) is implemented as a server for storing users' information and files. The CSP is also implemented to respond to the TPA's verification challenges by generating and returning a proof of the data correctness. There are three main classes for the CSP's implementation, which are storeUserInfo, storeUserFiles, and genProof. The class storeUserInfo is to receive the user's registration information and then store it in its database. The class storeUserFiles is to receive the user's files and then store them in its database. The class genProof is to generate a proof for the TPA's challenge and then send it to the TPA.

Store Users Information

The CSP receives the user's information during the user's registration process. After receiving the user's information, the CSP stores this information in its database in the users' table using the following SQL command.

```
int add=st1.executeUpdate("insert into users values ('"+fullname+"','"+username+"',
'"+password+"','"+profession+"','"+mobile1+"','"+emailid+"')");
```

Store Users Files

The CSP receives all data files from users in an encrypted form as discussed in section 4.3.1.2. The CSP then stores those encrypted files in its storage server. Thus, the CSP can manage the storage of the users' files without accessing the content of those files. The CSP also stores the details of all the users' files in its database in the files' table. The details include file id, user id, file name, owner name, and file's uploading date. To store those details, the CSP executes the following SQL command.

```
String sql = "insert into files (userid, filename, ownername, date) values ('" +
hm.get("userid") + "','" + hm.get("filename") + "','" + hm.get("ownername") + "','"+
dateFormat.format(date)+"')";
```

Figure 4.19 shows the details of all the users' files that are stored in the CSP's side.

Figure 4.19: Users files stored in the CSP

When the CSP tries to show the file's content by pressing the "View" button shown in Figure 4.19, the file will be shown as encrypted data blocks. Thus, the confidentiality of the users' data is protected. Figure 4.20 shows how the CSP views the file's content.

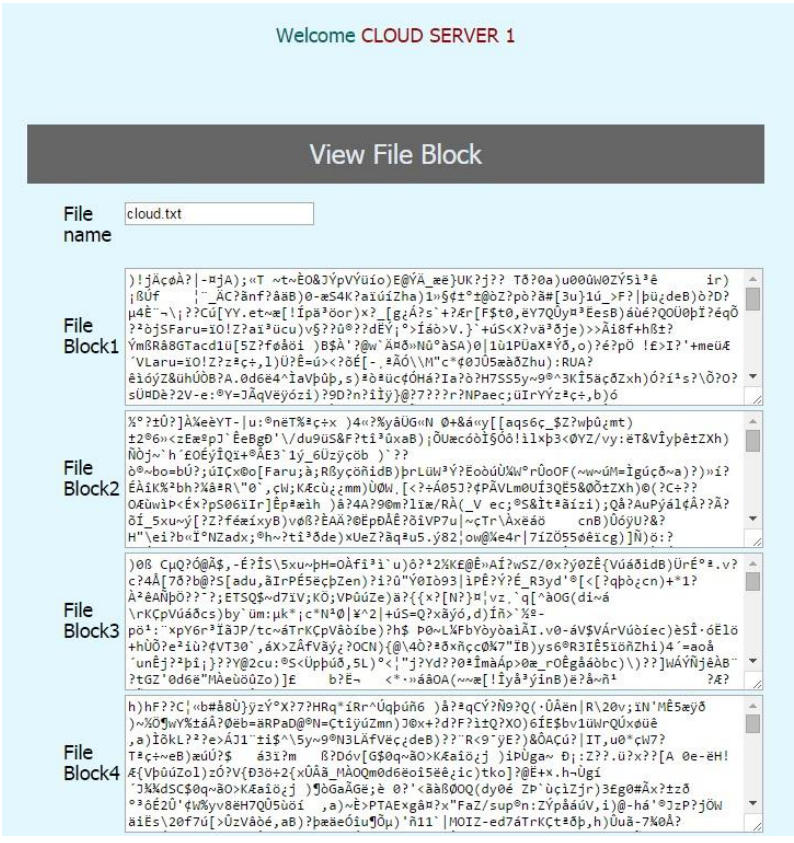


Figure 4.20: File content on the CSP's side

Generate Proof

Upon receiving the verification challenge from the TPA, the CSP will generate a proof of the data correctness and then pass that proof to the TPA. Figure 4.21, shows all users' files that are requested by the TPA to be verified. To generate a proof for those files, the CSP can select any file and then press the button "Generate Proof".

Figure 4.21: Generate a proof for the user's file

Figure 4.22 shows a code snippet of the proof generation process. The proof generation process has been discussed in section 3.4.4.2. During the implementation of the proof generation process, we have declared a proofGen function to generate a proof (R). This function takes n pseudo random numbers, the encrypted data blocks (Mi), and the received challenge (Q) as input, and produces a valid proof (R). To generate n random numbers, we have defined an integer ArrayList (x) of size n, where n is the total number of the data blocks. The range of the random numbers is set to be between 0 and 65536. The proof (R) is then computed as the sum of the ArrayList (x), the encrypted data block (M[j]), and the challenge (Q). After computing R, the CSP then returns R to the TPA.

```
public String proofGen (ECKey Q)
{

    ArrayList<Integer> x = new ArrayList<Integer>(n);

    Random rand = new Random();
    rand.setSeed(System.currentTimeMillis());

    for (int i=0; i<n; i++)
     {
    Integer r = rand.nextInt() % 65536;
    x.add(r);
     }

    for (int j=0 ; j<n; j++)
    {
       R +=  x.set(j, n) + "" + ""+ M[j] + Q ;
    }

    return R;
}
```

Figure 4.22: Code snippet for the proof generation process

4.3.4 Databases Implementation

There are two databases in our TP-DAS application. The first one is the CSP database that consists of two tables, namely, users and files. The second one is the TPA database that consists of a table called audit. Both databases have been implemented using MySQL server 5.6. To connect to those databases, a MySQL connector jar file has to be added to the project's libraries (The file is shown in Figure 4.23).

Figure 4.23: MySQL connector jar file

Having added the jar file to the project's libraries, we can simply use a JDBC driver to connect to the database using the code shown in Figure 4.24.

```
Class.forName("com.mysql.jdbc.Driver");
Connection con = DriverManager.getConnection("jdbc:mysql://localhost:3306/dataposession","root","admin");
con.close();
```

Figure 4.24: Database connection code

4.3.4.1 CSP Database Implementation

The CSP's database consists of two tables, namely, users and files. The users' table is for storing users' information that has been gathered during the registration process. The users' table consists of seven attributes (columns), which are user id, full name, username, password, profession, mobile, and email address. In this table, the user id is the primary key that can uniquely identify each user in the table. Figure 4.25 shows the implementation of the users' table using MySQL server 5.6 Command Line Client.

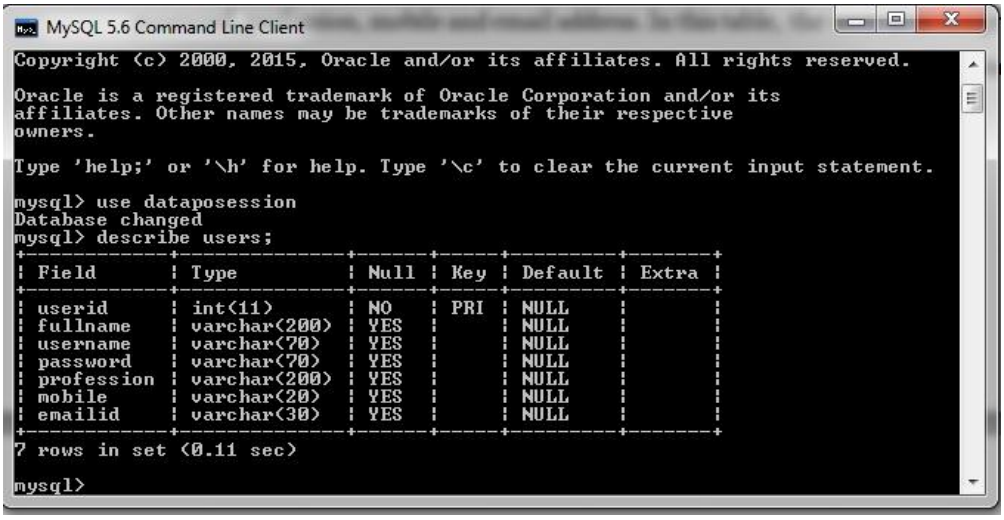


Figure 4.25: The implementation of the users' table

The second table is the files' table, which is used for storing the details of users' files. The files' table consists of five attributes, which are file id, user id, owner name, file name, and file's uploading date. In this table, the user id is a foreign key that references the primary key of the users' table to establish a link between the two tables. Figure 4.26 shows the implementation of the files' table.

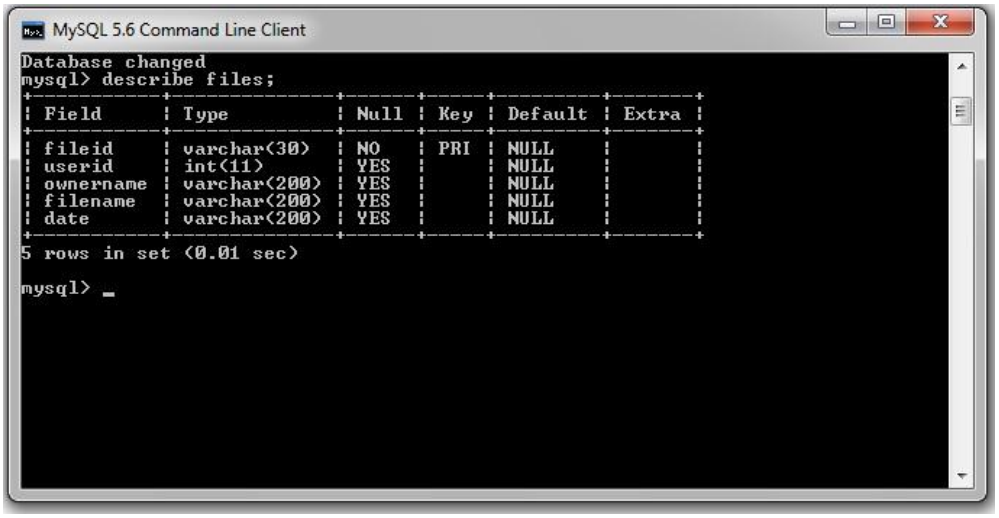


Figure 4.26: The implementation of the files' table

4.3.4.2 TPA Database Implementation

The TPA's database consists of a table called audit. The audit's table is for storing the verification metadata details. The audit's table consists of six attributes, which are file id, file name, owner name, public key (pk), metadata's uploading date, and the file's status. Figure 4.27 shows the implementation of the audit's table.

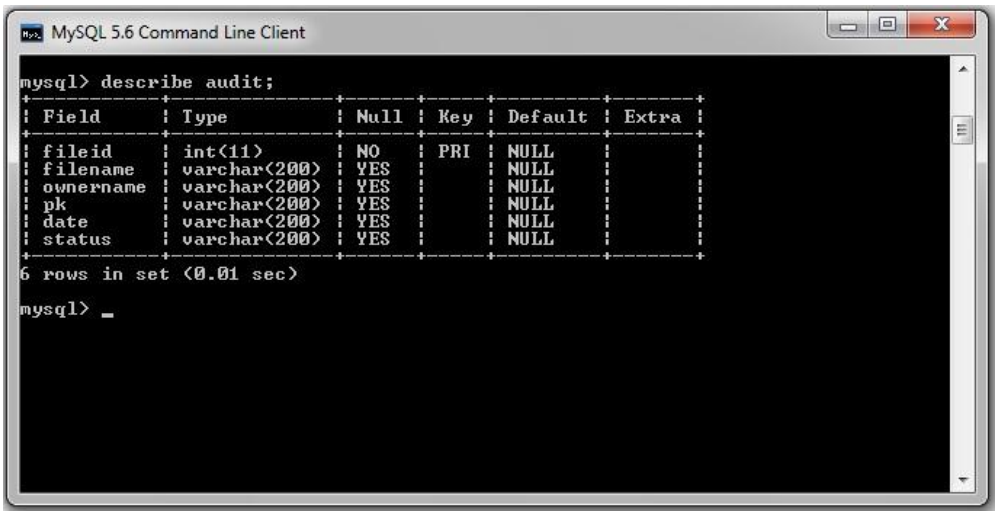


Figure 4.27: The implementation of the audit table

4.4 Challenges During the Implementation

The development of the proposed TP-DAS system involves solving two challenges to provide a better performance. The first challenge is how to reduce the communication cost required for updating a file. In basic schemes, when the user wants to update a small portion of a large file, they have to upload the whole updated file to the cloud. However, this is not efficient due to the high network bandwidth required for transferring the whole updated file each time.

To tackle this challenge, our proposed system splits the large file into a number of data blocks. When the user wants to update a portion of the file, they only need to update the data block containing that portion. Therefore, the user only needs to upload the updated data blocks instead of the whole file, resulting in a reduction of the communication cost.

The second challenge is how to reduce the computation cost required by the user to update a file. In basic schemes, when the user updates a small portion of a large file, they have to perform computations over the whole updated file to encrypt it and then generate its verification metadata. However, this may result in computation overhead on the user's side, especially when the file size is large.

To tackle this challenge, our proposed system splits the file into a number of data blocks. When the user updates a portion of the file, they only need to perform computation on the data block containing that portion. Therefore, the user only needs to perform computation on the updated data blocks instead of the whole file, resulting in a reduction of the computation cost.

4.5 Chapter Summary

In this chapter, we discussed the implementation of the TP-DAS system. NetBeans, Apache Tomcat web server, and MySQL server 5.6 have been selected as the development environments for the system's implementation. Java programming language, JavaServer Pages (JSP) technology, and MySQL language have been used to convert the system's design into a web application. The implementation of the user, TPA, and CSP has been provided, along with the implementation of the system's databases. These implementations have been illustrated by graphical user interfaces and code snippets. Two implementation difficulties regarding the computation and communication costs have been discussed in this chapter.

Chapter 5. Testing and Evaluation

5.1 Chapter Overview

This chapter tests and evaluates our proposed TP-DAS system in terms of security and performance. Regarding security, we first analyses the security of the Elliptic Curve Cryptography (ECC) used. This is followed by an analysis to prove that the system can correctly check the integrity of the data as well as maintain the confidentiality of the data. Regarding performance, the system is evaluated in terms of the computation, communication, and storage costs required by each entity in the system. Having tested the system's performance, the system is compared with an RSA-based system in terms of the key generation time and the computation time. All performance tests are carried out using the TP-DAS web application on a system with Core i5 2.5 GHz CPU and 6 GB RAM running Windows 2007.

The structure of this chapter is as follows. Section 5.2 analyses the security of Elliptic Curve Cryptography, after which an integrity and confidentiality analysis will be provided. Section 5.3 measures the performance of the proposed TP-DAS system in terms of computation, communication, and storage costs. Section 5.4 compares the proposed system with an existing RSA-based system. Section 5.5 summarises the chapter.

5.2 Security Analysis

In this section, we will discuss the security of the cryptographic primitive used, which is Elliptic Curve Cryptography (ECC). This will be followed by a security analysis of our proposed system in terms of data integrity and data confidentiality.

5.2.1 Security of Elliptic Curve Cryptography

The security of Elliptic Curve Cryptography (ECC) depends mainly on the difficulty of solving the order of the elliptic curve and the Elliptic Curve Discrete Logarithm Problem (ECDLP).

- **Difficulty of solving the order of the elliptic curve**

Let assume $E_n(a,b)$ is an elliptic curve satisfying $\gcd(4a^3+27b^2, n)=1$. Let assume p and q are two large primes and n=pq. N_n is the order of the elliptic curve over the ring Z_n, which is computed as N_n= lcm (p+1,q+1). P is the generator of $E_n(a,b)$, which is computed as P = (K· N_n+1). Solving N_n is proved to be as computationally hard as factoring the integer n [40].

In our proposed system, the order of the elliptic curve (N_n) is kept secret on the user's side. In addition, computing N_n is as difficult as factoring the integer n. For these reasons, malicious parties, such as malicious CSPs, will find it difficult to compute N_n. Thus, dishonest CSPs cannot cheat the TPA during the auditing task, as we will discuss in section 5.2.2.

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**

This problem can be defined as the difficulty of finding the random element c from the equation Q= cP, where Q and P are points on the elliptic curve such that $Q,P \in E_n(a,b)$ and c < P. It is somewhat easy to find Q given P and c, but it is difficult to find c given Q and P [32].

In our proposed system, the TPA challenges the CSP by sending Q. P is a public parameter that is known to the CSP. Having Q and P, the CSP will find it hard to determine c. Therefore, this would assure that the CSP cannot cheat the TPA during the auditing process, as we will discuss in section 5.2.2.

5.2.2 Data Integrity Analysis

The proposed system allows users to check the integrity of their remote data files by relying on the TPA. Here, in this section, we will show how the proposed system can correctly check the integrity of the data, assuming that both the TPA and the CSP are honest. Then, we will show how the proposed system is secured against dishonest CSPs.

Theorem 5.1: If the CSP honestly computes the required proof (R) using the challenge received from the TPA, the TPA can then consider the proof as valid.

Proof: this theorem can be proved by showing that the proof (R) matches R'. When the TPA sends a verification challenge to the CSP, the CSP will generate a proof (R) and then send it

to the TPA. Upon receiving R, the TPA will compute R' and then compare it with the received R. If R = R', the data is in its original status, and therefore, the proof is valid.

With the help of the commutative property of point multiplication that exists in an elliptic curve [41], we can successfully prove R = R' if the CSP honestly computes R. The proof is as follows.

$$R' = cZ \bmod n$$

$$Z = \prod_{i=1}^{n} xi\ di \bmod n, \text{ where } xi = fk_{PRF}(i) \text{ for } i \in [1,n]$$

$$= \prod_{i=1}^{n} (xi\ M_i\ P (\bmod N_n)) \bmod n$$

$$= \sum_{i=1}^{n} xi\ M_i\ P \bmod n$$

$$R' = cZ \bmod n$$

$$= c\ (\prod_{i=1}^{n} (xi\ M_i\ P \bmod n))$$

$$= c\ (\sum_{i=1}^{n} xi\ M_i\ P \bmod n)$$

$$= R$$

As R = R', the proof (R) is valid. Thus, the TPA can assure that the CSP still holds all the data blocks in their original status.

Theorem 5.2: The system is secured against dishonest CSPs. The CSP cannot dishonestly compute the required proof.

Proof: the CSP might be dishonest, and try to cheat the TPA during the data integrity checking process. Here, we will show different possible ways of cheating the TPA during the auditing process. Having stated these ways, we will show how the proposed system is secured against potential threats.

1- The CSP might generate a proof (R) without storing the data. However, this is not possible in our proposed system, because the generation of the proof requires accessing all the encrypted data blocks (M_i) in order to compute the proof correctly. Thus, if the CSP does not store the encrypted file (F'), it cannot generate a valid proof during the auditing process.

2- The CSP might use previous proofs as a response to the TPA challenges. However, this is not feasible, because the CSP has to find the random integer (c) from the new challenge to use the previous proof. To clarify this, assume that the TPA has challenged the CSP about a file A and the CSP then responded to the TPA by generating the required proof R. After that, the CSP stored the R in its database for future challenges and then it modified or deleted file A.

   If the TPA challenges the CSP again about file A, the CSP will not be able to use the stored R unless it can find the random integer c, which is different each time, from the new challenge. However, due to the ECDLP that was discussed in section 5.2.1, it is hard to find c given Q and P. Therefore, the CSP cannot respond correctly to the TPA challenges using previous proofs.

3- The CSP might corrupt some data blocks and then try to generate a proof only from uncorrupted blocks. However, this is not feasible, because the proposed system requires the CSP to perform computation over all the data blocks to successfully generate the proof. Even if the CSP has only modified a small portion of a block and then tried to generate the proof, the generated proof will be different from the TPA computation, and therefore, the verification result will be negative.

   This is not the case for the majority of existing systems that rely on probabilistic data integrity assurance. In these systems, the CSP has only to perform computation over a specific number of data blocks to generate the proof. Therefore, the generated proof might be accepted as valid proof even though the CSP has corrupted some data blocks. For instance, assume that the TPA challenges the CSP to generate proof from

file A, which consists of five blocks, by specifying three blocks (1, 3, and 5). The CSP can successfully generate the proof even if it has modified or deleted blocks 2 and 4.

4- The CSP might not store the encrypted file, but, instead, it has an algorithm to compute $M_i \ mod \ N_n$ for $i \in [1,n]$. So, during the auditing task, the CSP will run the algorithm and then generate the proof. However, this is not possible in our proposed system, because the CSP has to compute N_n to build this algorithm. Computing N_n is computationally difficult due to the difficulty of solving the order of the elliptic curve, as we have discussed in section 5.2.1.

From this discussion, it is clear that our proposed system cannot accept the CSP's proof unless the CSP honestly computes the proof. In addition, our proposed system is secured against dishonest CSPs. In other words, any cheating attempt from the CSP during the auditing process will be detected by the TPA.

5.2.3 Data Confidentiality Analysis

As the user's data is stored at an untrusted entity (CSP), the confidentiality of the data should be maintained. Malicious CSPs or attackers should not be able to access the content of the user's data while it is in the cloud. Also, the TPA should not acquire any knowledge regarding the content of the user's data during the auditing process. Here, in this section, we will show how the proposed system is secured against data leakage to unauthorised entities.

Theorem 5.3: The proposed system prevents unauthorised entities (the CSP, the TPA, or attackers) from accessing the content of the user's data.

Proof: this theorem will be proved against the following attacks.

1- If malicious parties (e.g., the CSP) try to access the content of the user's data, they will not be able to acquire any knowledge regarding the data, because the file is encrypted prior to uploading it to the cloud using a secret parameter. These malicious parties then need to know the secret key, which is kept secret on the user's side, in order to decrypt the file and then access its content. If these parties try to do different combinations of the public key to get the secret key, they will not succeed because of

the ECDLP that was explained in section 5.2.1. Thus, malicious parties cannot access the content of the user's data.

2- If the TPA tries to access the content of the user's data from the stored metadata (D), it will not succeed for the following reason. The user computes the verification metadata over the encrypted file using the secret key and then sends the verification metadata to the TPA along with the public key. To access the content of the user's data from the metadata, the TPA has to get the secret key. Having the metadata and the public key will not help the TPA to find the secret key. In addition, during the auditing process, the TPA will not be able to derive any knowledge regarding the content of the user's data from the proof received. This is because the proof is generated from the encrypted file, so, the secret key is needed to decrypt the data.

From this discussion, it is clear that the proposed system is secure from data leakage to the TPA, malicious parties, or attackers. This is because the data file is encrypted prior to uploading it to the cloud.

5.3 Performance Analysis

In this section, the performance of the proposed system will be tested and evaluated in terms of computation, communication, and storage costs. All performance tests have been conducted using the TP-DAS web application on a system with Core i5 2.5 GHz CPU and 6 GB RAM running Windows 2007.

5.3.1 Computation Cost

The computation cost on the user's, the TPA's, and the CSP's side will be measured. The computation cost and time required by the user to encrypt the file and generate its verification metadata will be calculated. After that, the computation cost and time required by the TPA to generate a verification challenge and then verify the CSP's proof will be considered. Finally, the computation cost and time required by the CSP to generate the required proof will be measured.

User's side: during the initialisation phase, the user needs to encrypt the file and then generate verification metadata for the encrypted file. As the file is divided into n data blocks,

the user needs to perform an integer addition of two bits to encrypt each data block. So, the computation cost for encrypting the whole file is n integer additions, where n is the total number of data blocks. For instance, if the file consists of ten data blocks, the computation cost of encrypting the file is ten integer additions.

After encrypting the file, the user computes the verification metadata for each encrypted data block by performing a modular multiplication of two bits. So, the computation cost for generating the verification metadata is n modular multiplications. Therefore, the total computation cost required by the user is n integer additions + n modular multiplications.

Different file sizes have been used to measure the computation time required by the user to encrypt the file and generate its verification metadata. In this report, MB means Megabyte and ms means milliseconds. The following table represents the computation time required by the user.

File Size (MB)	Computation Time (ms)
10	192.15
20	228.90
30	265.41
40	302.45
50	338.15

Table 5.1: Computation time required by the user

TPA's side: during the verification phase, the TPA issues a challenge and then sends it to the CSP. To generate the challenge, the TPA needs to generate two random numbers, which are k_{PRF} and c, and then compute $Q = cP$. The cost of computing Q is a point multiplication of two bits. Upon receiving the proof from the CSP, the TPA computes $xi = fk_{PRF}(i)$ *for* $i \in [1,n]$, whose cost is n pseudorandom number generations. The TPA then computes Z, whose cost is the sum of n modular multiplications of two bits. Finally, the TPA computes R', whose cost is a modular multiplication of two bits. Therefore, the total computation cost on the TPA's side is 1 point multiplication + n pseudorandom number generations + the sum of n modular multiplications + 1 modular multiplication.

Different file sizes have been used to measure the computation time required by the TPA to generate a challenge and then verify the CSP's proof. The following table represents the computation time required by the TPA.

File Size (MB)	Computation Time (ms)
10	395.15
20	427.5
30	470.0
40	518.75
50	581.25

Table 5.2: Computation time required by the TPA

CSP's side: during the verification phase, the CSP generates a proof (R) and then sends it to the TPA. To generate the proof, the CSP computes $xi = fk_{PRF}\ (i)\ for\ i \in [1,n]$, whose cost is n pseudorandom number generations. The CSP then computes *R*, whose cost is the sum of n modular multiplications of three bits. Therefore, the total computation cost on the CSP's side is n pseudorandom number generations + the sum of n modular multiplications.

Different file sizes have been used to measure the computation time required by the CSP to generate the proof. The following table represents the computation time required by the CSP.

File Size (MB)	Computation Time (ms)
10	308.96
20	350.53
30	390.98
40	428.05
50	468.76

Table 5.3: Computation time required by the CSP

The following table summarises the total computation cost required by each entity in the system. It is clear that the computation cost on the TPA's side exceeded that for the user and the CSP.

Entity	Computation Cost
User	n integer additions + n modular multiplications
TPA	1 point multiplication + n pseudorandom number generations + the sum of n modular multiplications + 1 modular multiplication.
CSP	n pseudorandom number generations + the sum of n modular multiplications.

Table 5.4: Computation cost required by each entity in the system

5.3.2 Communication Cost

This section will measure the communication costs incurred by the TPA and the CSP during the auditing process.

During the verification phase, the TPA sends a challenge to the CSP. The challenge consists of a key (k_{PRF}) and $Q = cP$. The communication cost of the challenge is k + |n| bits (k is the key length, e.g., 128-bit, and |n| is an elliptic curve group order of 160-bit). The CSP then sends a proof (R) to the TPA. The communication cost of sending R is |n| bits. So, the total communication cost required to audit the user's data is k + 2 |n| bits.

5.3.3 Storage Cost

This section will measure the storage costs required by the user, the TPA and the CSP.

User's side: the user needs to store the private key. The cost of storing the private key is |p+1|+|q+1| bits, where |p| and |q| are the length of two random primes.

TPA's side: the TPA needs to store the user's public key and the verification metadata. The cost of storing the public key, which is {n,P}, is 2|n| bits (|n| = 160 bits). The cost of storing the verification metadata is linear to the number of data blocks, which is $(|F'| / l) \cdot |n|$ bits. |F'| denotes the length of the encrypted file, while *l* denotes the length of each encrypted data block. Therefore, the total storage cost on the TPA's side is $(2 + (|F'| / l)) \cdot |n|$ bits.

CSP's side: the CSP needs to store the whole encrypted file (F'). Therefore, the storage cost on the CSP's side is |F'| bits.

The following table summarises the storage cost required by each entity in the system.

Entity	Storage	Storage Cost
User	Private key.	|p+1|+|q+1| bits.
TPA	Public key and verification metadata.	$(2 + (\|F'\| / l)) \cdot \|n\|$ bits.
CSP	Encrypted file.	|F'| bits.

Table 5.5: Storage cost required by each entity in the system

5.4 Comparison with Existing Systems

The majority of existing systems rely on RSA for checking the integrity of the remote data. These systems adopted RSA because the key length for RSA has been increasing recently to offer a high level of security. However, the increase of the key length would result in more computation overhead. To avoid this issue, we proposed an ECC-based system because ECC can offer the same level of security as the RSA, but with a smaller key length [30].Having a smaller key length, the computation overhead is reduced [30].

In this section, the performance characteristics of existing systems, that use RSA, will be compared with our proposed system which uses ECC. This would embrace a comparison between RSA and ECC in terms of the time needed to generate the keys. In addition, another comparison will be conducted to compare our ECC-based system with an RSA-based system in terms of the computation time required by each entity in the system.

5.4.1 Key Generation Time

In this section, we will compare ECC with RSA in terms of the key generation time. Jansma and Arrendondo have estimated the ECC key lengths that provide the same security level as the RSA key lengths [39]. The following table shows the ECC key lengths and the equivalent RSA key lengths [39].

ECC key length (bits)	RSA key length (bits)
160	1024
233	2240
283	3072
409	7680
571	15360

Table 5.6: RSA and ECC equivalent key lengths [39]

From table 5.6, it is clear that ECC uses a smaller key length in comparison to RSA, although both the RSA and the ECC keys are of the same security level. For instance, a 233-bit ECC key can offer the same security level as a 2240-bit RSA key.

Jansma and Arrendondo have also done a performance test to measure the key generation time for both ECC and RSA [39]. They have found that the use of ECC can result in generating the public and private keys in a short time compared to the RSA. A big difference between ECC and RSA in the key generation time has been recorded, especially with the increase of the key length. The following table represents the key generation time for both ECC and RSA using equivalent key lengths [39].

Key length (bits)		Key generation time (ms)	
ECC	**RSA**	**ECC**	**RSA**
160	1024	80	160
233	2240	180	7470
283	3072	270	9800
409	7680	640	133900
571	15360	1440	679060

Table 5.7: A comparison between ECC and RSA in the key generation time [39]

It is clear that our ECC-based system uses smaller key lengths compared to the RSA-based systems, and therefore, the key generation time in our system is less than that for the existing RSA-based systems.

5.4.2 Computation Time

The computation time required by each entity in our ECC-based system has been measured in section 5.3.1. In this section, we will compare our ECC-based system with an RSA-based-system, which is defined in [31], in terms of the computation time required by each entity in the system. Generally, our ECC-based system achieved a better performance compared to the RSA-based system. This is because we have chosen 160-bits elliptic curve group order in our proposed system instead of 1024-bits RSA modulus in the RSA-based system [31]. Thus, the computation time required by each entity in our proposed system is reduced.

Figure 5.1 compares our ECC-based system with the existing RSA-based system [31] in terms of the computation time required by the user to generate the file's verification metadata. It is clear that our ECC-based system outperforms the RSA-based system.

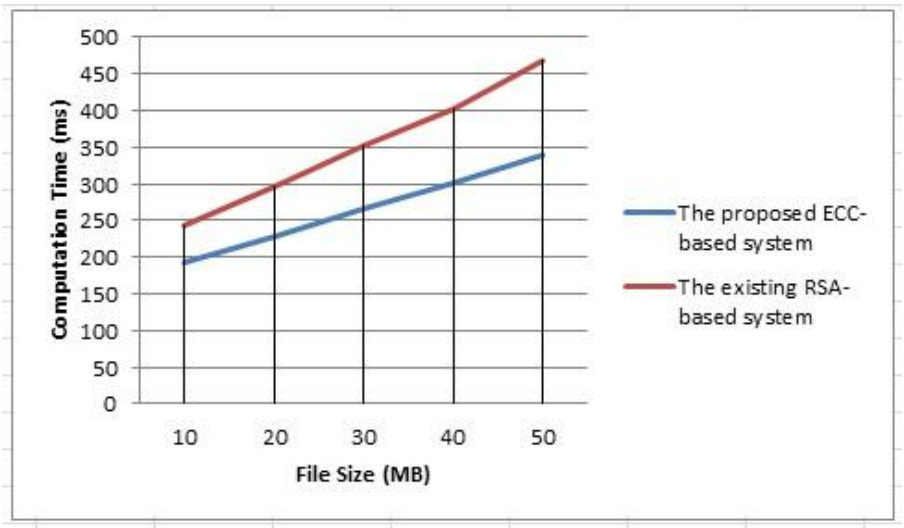


Figure 5.1: A comparison regarding the computation time required by the user.

Figure 5.2 compares our ECC-based system with the existing RSA-based system [31] in terms of the computation time required by the TPA to generate a challenge and then verify the CSP's proof. It is clear that our ECC-based system outperforms the RSA-based system.

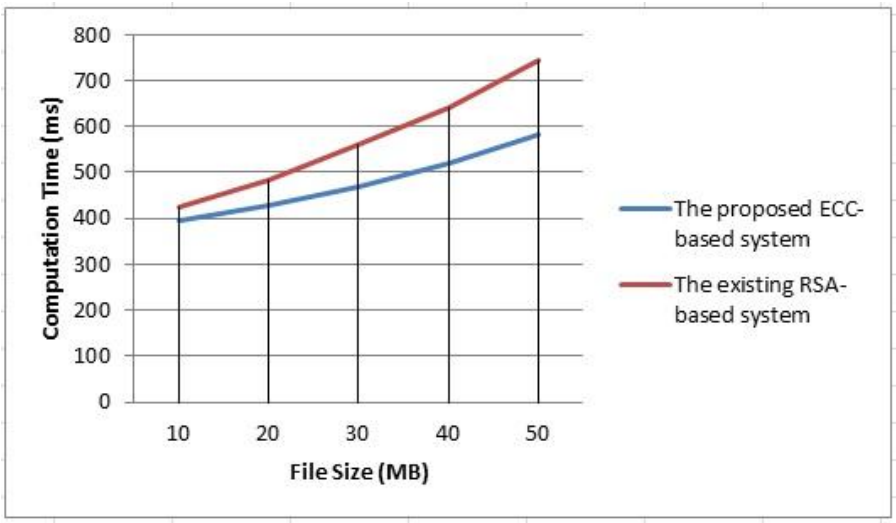


Figure 5.2: A comparison regarding the computation time required by the TPA.

Figure 5.3 compares our ECC-based system with the existing RSA-based system [31] in terms of the computation time required by the CSP to generate a proof from the stored data. It is clear that our ECC-based system outperforms the RSA-based system.

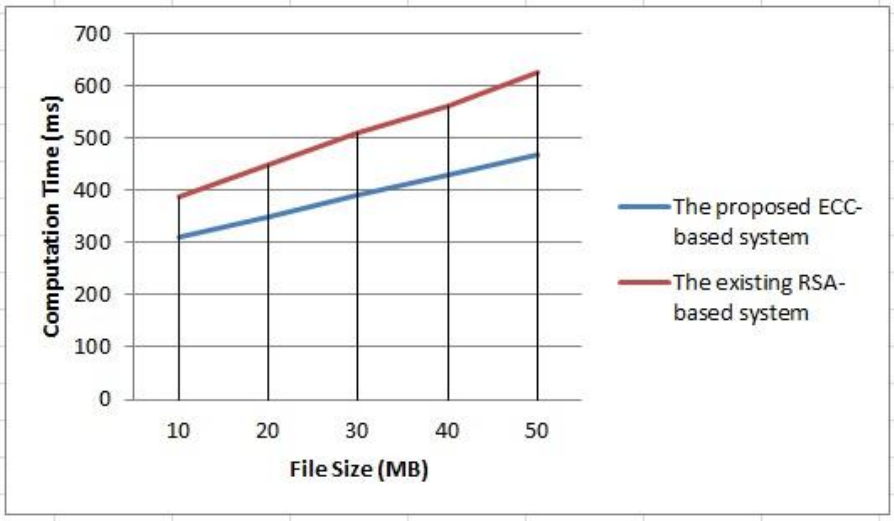


Figure 5.3: A comparison regarding the computation time required by the CSP.

5.5 Chapter Summary

In this chapter, we tested and evaluated the security and performance of the proposed system. This chapter showed the security of the cryptographic primitive (ECC) used, followed by an analysis to prove the validation of the proof generated by an honest CSP. In addition, it showed how our system is secured against dishonest CSPs. The confidentiality of the user's data has also been proved under different attacks. After that, the performance of the proposed system has been measured in terms of the computation, communication, and storage costs required by each entity in the system. Finally, the performance of the proposed system has been compared with that of an RSA-based system.

Chapter 6. Conclusion and Future Work

The number of users who shift their data remotely into the cloud has been increasing recently. This is because cloud computing can offer a storage service with low costs and the fact that shifting data into the cloud could relieve users of the burden of storing data locally, and the maintenance costs associated with this. However, as data is stored at an untrusted party (CSP), users will lose control of their data. Thus, the integrity and the confidentiality of the user's data will be at risk. The user's data, therefore, needs to be checked on a regular basis while it is in the cloud. In addition, the content of the user's data must not be accessed except by authorised users.

There are various related solutions, which have been discussed in the literature review in section 2.5, to check the integrity of the user's remote data. Each solution has its advantages and drawbacks. However, none of these solutions can preserve the confidentiality of the data. Thus, the user's data can be leaked to malicious parties or attackers while it is in the cloud.

In this project, we have proposed a secure and efficient solution called TP-DAS that can achieve both data integrity and data confidentiality. The proposed solution has been implemented using two cryptographic primitives, namely, ECC and PRF. The ECC has been used because it can provide the same security level as the RSA and other Public Key Cryptography (PKC), but with smaller keys. Thus, the computation costs are reduced in our solution.

In our proposed solution, the user first encrypts the data file prior to uploading it to the cloud to ensure the confidentiality of the data, and then computes the verification metadata over the encrypted file. Later, the user can ask the TPA to check the integrity of the data on their behalf. The TPA should be able to detect any data integrity drift during the auditing process and then inform the user of this drift.

Our proposed solution supports public auditability as it enables the TPA to check the integrity of the data without downloading the data from the CSP. In addition, it supports dynamic data operations such as inserting new data blocks, modifying existing data blocks, or deleting existing data blocks.

The security of the proposed solution has been proved in terms of data integrity and data confidentiality. The security of the proposed solution depends mainly on the difficulty of

solving particular problems in ECC, which have been discussed in section 5.2.1. The performance of the proposed solution has also been proved in terms of computation, communication, and storage costs. In addition, we have proved that our proposed solution outperforms the existing RSA-based solutions in terms of the computation cost.

The main contributions of our proposed solution are:

- It can detect any data corruption because it is based on deterministic data assurance. This means that if the CSP modifies or deletes an existing data block, it will not be able to generate a valid proof. Thus, the TPA will detect this corruption and then inform the user of this corruption.
- It can achieve data confidentiality by protecting the user's data from being leaked to unauthorised parties or attackers.
- It can reduce the computation time required by each entity in the system as it uses smaller key lengths.

6.1 Future Work

Due to time limitations, only three main contributions have been achieved in this project. These contributions have resulted in a secure solution that can detect any data drift as well as preserving the data from being leaked to unauthorised parties. In addition, these contributions enhanced the performance of our proposed solution in terms of the computation cost. However, there is still a lot of work required to achieve the optimum level of performance. The following are just two possible ideas that can improve the performance of the proposed solution.

- **Batch auditing:** this feature allows the TPA to handle and perform numerous auditing tasks at the same time. As there are a great number of data users in the cloud, the TPA may receive a multitude of auditing requests from different users at once. Therefore, it is vital to have a solution that can handle and perform all those auditing requests simultaneously rather than one auditing request at a time. Having considered the batch auditing feature in the future work, the performance of the proposed solution will be enhanced.

- **Dynamic data operations at the data level:** all related solutions embracing our proposed solution can support dynamic data operations at the block level. That means users can add a new block, modify an existing block, or delete an existing block. However, this can result in unnecessary communication costs for the following reason. Suppose a user wants to modify an existing block by adding a few bits. They have to update the whole corresponding block and the verification metadata for that block. The updated block will then be sent to the CSP, while the updated metadata will be sent to the TPA. Yet, sending the whole block to the CSP, rather than a few bits results in more communication costs.

  For the future work, our proposed solution can be extended to support dynamic data operations at the data level. That means when users want to add a few bits to an existing block, they only need to send these few bits to the CSP, rather than sending the entire block. By doing so, the communication costs can be reduced, resulting in a better performance. Although this idea reduces the communication costs, it might result in more computational effort on the CSP's side for placing the modified bits in the right place in the block.

References

[1] Mell, Peter, and Tim Grance. "The NIST definition of cloud computing."*National Institute of Standards and Technology* 53.6 (2009): 50.

[2] Mell, Peter, and Tim Grance. "Draft NIST working definition of cloud computing." *Referenced on June. 3rd* 15 (2009): 32.

[3] Li, Jin, et al. "An efficient proof of retrievability with public auditing in cloud computing." *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*. IEEE, 2013.

[4] Fox, Armando, et al. "Above the clouds: A Berkeley view of cloud computing."*Dept. Electrical Eng. and Comput. Sciences, University of California, Berkeley, Rep. UCB/EECS* 28 (2009): 13.

[5] Wang, Cong, et al. "Privacy-preserving public auditing for data storage security in cloud computing." *INFOCOM, 2010 Proceedings IEEE*. Ieee, 2010.

[6] Wang, Qian, et al. "Enabling public verifiability and data dynamics for storage security in cloud computing." *Computer Security–ESORICS 2009*. Springer Berlin Heidelberg, 2009. 355-370.

[7] Wang, Qian, et al. "Enabling public auditability and data dynamics for storage security in cloud computing." *Parallel and Distributed Systems, IEEE Transactions on* 22.5 (2011): 847-859.

[8] Juels, Ari, and Burton S. Kaliski Jr. "PORs: Proofs of retrievability for large files." *Proceedings of the 14th ACM conference on Computer and communications security*. Acm, 2007.

[9] Li, Jin, et al. "An efficient proof of retrievability with public auditing in cloud computing." *Intelligent Networking and Collaborative Systems (INCoS), 2013 5th International Conference on*. IEEE, 2013.

[10] Gohel, Mihir R., and Bhavesh N. Gohil. "A new data integrity checking protocol with public verifiability in cloud storage." *Trust Management VI*. Springer Berlin Heidelberg, 2012. 240-246.

[11] Shacham, Hovav, and Brent Waters. "Compact proofs of retrievability."*Advances in Cryptology ASIACRYPT 2008*. Springer Berlin Heidelberg, 2008. 90-107.

[12] Ateniese, Giuseppe, et al. "Provable data possession at untrusted stores."*Proceedings of the 14th ACM conference on Computer and communications security*. Acm, 2007.

[13] Singh, Rampal, Sawan Kumar, and Shani Kumar Agrahari. "Ensuring Data Storage Security in Cloud Computing." *International Journal Of Engineering And Computer Science ISSN* (2012): 2319-7242.

[14] Li, Ling, et al. "Study on the third-party audit in cloud storage service." *Cloud and Service Computing (CSC), 2011 International Conference on*. IEEE, 2011.

[15] Erway, Chris, et al. "Dynamic provable data possession." *Proceedings of the 16th ACM conference on Computer and communications security*. Acm, 2009.

[16] Priyadharshini, B., and P. Parvathi. "Data integrity in cloud storage." *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*. IEEE, 2012.

[17] Goyal, Renuka, and Navjot Sidhu. "Third Party Auditor: An Integrity Checking Technique for Client Data Security in Cloud Computing." *International Journal of Computer Science & Information Technologies* 5.3 (2014).

[18] Zhang, Qi, Lu Cheng, and Raouf Boutaba. "Cloud computing: state-of-the-art and research challenges." *Journal of internet services and applications* 1.1 (2010): 7-18.

[19] Innocent, A. "Cloud infrastructure service management-a review." *arXiv preprint arXiv:1206.6016* (2012).

[20] Ramgovind, Sumant, Mariki M. Eloff, and Elme Smith. "The management of security in cloud computing." *Information Security for South Africa (ISSA), 2010*. IEEE, 2010.

[21] Advantages and Disadvantages of Digital Signatures. [Online] Available at: http://lerablog.org/technology/data-security/advantages-and-disadvantages-of-digital-signatures/ [Accessed 19 April,2015].

[22] Vaudenay, Serge. *A classical introduction to cryptography: Applications for communications security*. Springer Science & Business Media, 2006.

[23] Digital Signautres. [online] Available at: https://msdn.microsoft.com/en-us/library/windows/desktop/aa381977(v=vs.85).aspx [Accessed 19 April,2015].

[24] Message Authentication Code (MAC). [online] Available at: http://www.sqa.org.uk/e-learning/WebTech04CD/page_12.htm [Accessed 19 April,2015].

[25] System Security. [online] Available at: http://alchetron.com/System-Security-867-W [Accessed 20 April, 2015].

[26] Koyama, Kenji, et al. "New public-key schemes based on elliptic curves over the ring Zn." *Advances in Cryptology—CRYPTO'91*. Springer Berlin Heidelberg, 1992.

[27] William Stallings. *Cryptography and Network Security: Principles and Practice, 5/E*., 2010.

[28] Java Application Servers Report. [online] Available at: http://www.fscript.org/prof/javapassport.pdf [Accessed 19 May,2015].

[29] Welling, Luke, and Laura Thomson. *PHP and MySQL Web development*. Sams Publishing, 2003.

[30] Gupta, Kamlesh, and Sanjay Silakari. "ECC over RSA for Asymmetric Encryption: A review." *IJCSI International Journal of Computer Science Issues* 8.3 (2011).

[31] Hao, Zhuo, Sheng Zhong, and Nenghai Yu. "A privacy-preserving remote data integrity checking protocol with data dynamics and public verifiability." *Knowledge and Data Engineering, IEEE transactions on* 23.9 (2011): 1432-1437.

[32] Lauter, Kristin E., and Katherine E. Stange. "The elliptic curve discrete logarithm problem and equivalent hard problems for elliptic divisibility sequences." *Selected Areas in Cryptography*. Springer Berlin Heidelberg, 2009.

[33] Java Programming Language Handbook. [online] Available at http://portal.aauj.edu/portal_resources/downloads/programming/javap_programming_language_handbook.pdf [Accessed 16 July, 2015].

[34] Integrated Development Environments for Natural Language Processing. [online] Available at http://www.textanalysis.com/TAI-IDE-WP.pdf [Accessed 17 July, 2015].

[35] It's all about intelligent code completion. [online] Available at http://code-recommenders.blogspot.co.uk/2010/05/its-all-about-intelligent-code.html [Accessed 17 July, 2015].

[36] What's New in MySQL 5.6. [online] Available at http://dev.mysql.com/tech-resources/articles/whats-new-in-mysql-5.6.html#optimizer [Accessed 19 July, 2015].

[37] JavaServer Pages Overview. [online] Available at http://www.oracle.com/technetwork/java/overview-138580.html [Accessed 20 July, 2015].

[38] Introduction to JavaServer Pages. [online] Available at http://www.scribd.com/doc/54581336/j-Introjsp-PDF [Accessed 21 July, 2015].

[39] Jansma, Nicholas, and Brandon Arrendondo. "Performance comparison of elliptic curve and rsa digital signatures." *nicj. net/files* (2004).

[40] Koyama, Kenji, et al. "New public-key schemes based on elliptic curves over the ring Zn." *Advances in Cryptology—CRYPTO'91*. Springer Berlin Heidelberg, 1992.

[41] Oualha, Nouha, Melek Önen, and Yves Roudier. "A security protocol for self-organizing data storage." *Proceedings of The Ifip Tc 11 23rd International Information Security Conference*. Springer US, 2008.