PES UNIVERSITY

UE17CS490A – Capstone Project Phase – 1

End Semester Assessment

Project Title : Layer 7 Integrity Check On Cloud Storage Platforms(L7ICCSP)
Project ID : PW21HBP02
Project Guide : Prof. Prasad B Honnavalli
Project Team : R Siva Girish Gaurav C.G
Sangam Kedilaya Zenkar S

Agenda

- Problem Statement
- Abstract and Scope
- Literature Survey
- Suggestions from Review – 3
- Design Approach
- Design Constraints, Assumptions & Dependencies
- Proposed Methodology / Approach
- Architecture
- Design Description
- Technologies Used
- Project Progress
- References

Problem Statement

- Cloud service providers are deemed secure by default. Most users are oblivious to any tampering of their files if it occurs.
- With such high progression in algorithms and data structures it is a minimal requirement for all software users to expect their applications to be extremely fast and smooth to handle. Therefore many cloud providers are silent on the integrity of the files uploaded.
- There is no means to check for the integrity of the files we download from the cloud.
- Therefore it is the goal of our product to serve as an application that verifies the integrity of the files stored on untrusted cloud servers.

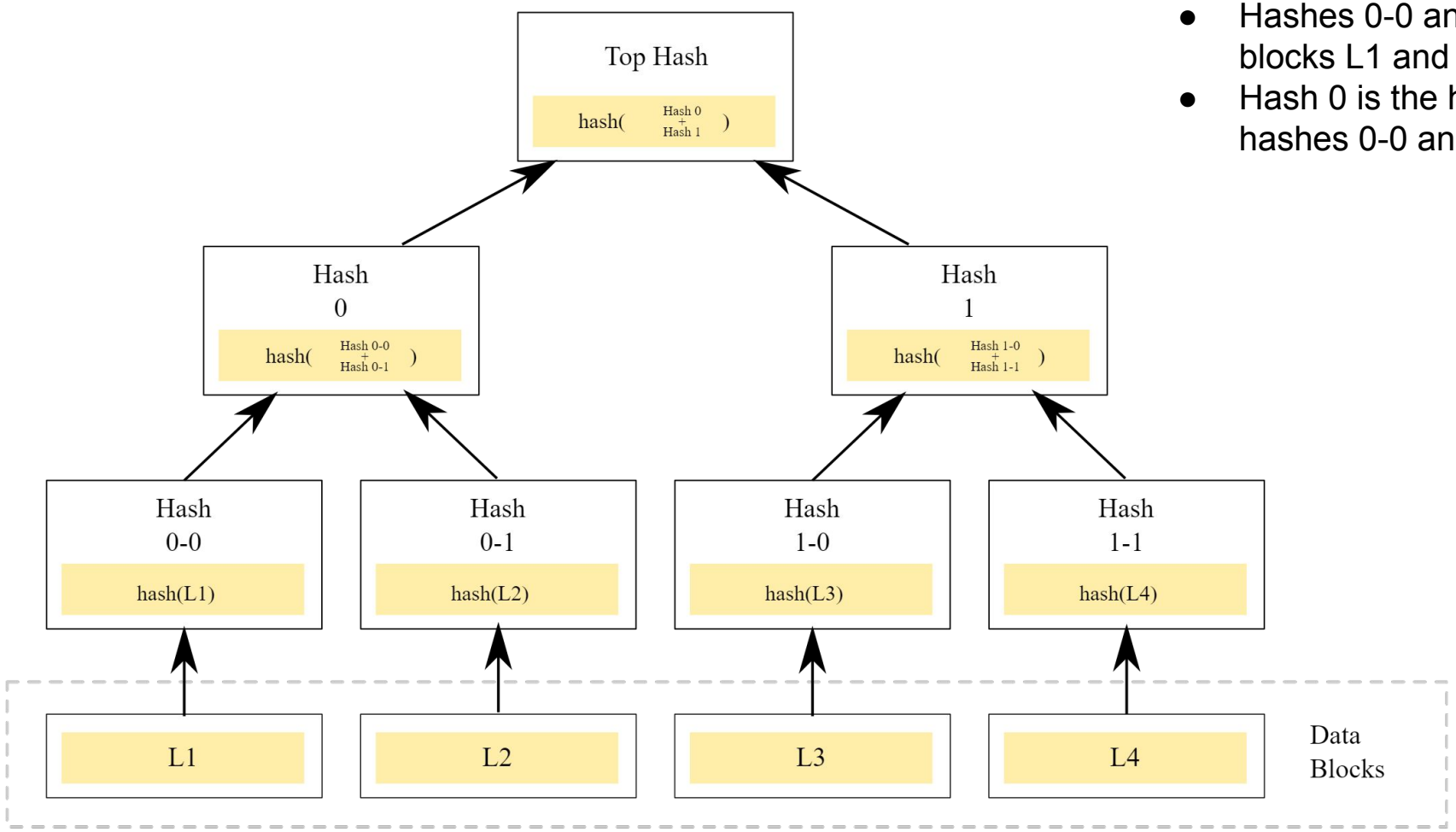
Abstract and Scope

- To create an application that verifies the integrity of the files stored on untrusted cloud servers.
- The application works on top of cloud servers. Files are uploaded and downloaded to the cloud through the application.
- Integrity checks are initiated by the user upon downloading the file.
- Limited to integrity verification, no retrieval of tampered files.
- Single user application.

Research Paper I : File System Support Solution

- Abstract
  - The main focus of this paper was to create a system that maintains the integrity of files hosted on the cloud by untrusted servers.
  - Looking deeper into their implementation-:
    - A B+ tree which serves as a Merkle Tree is constructed to keep track of files while maintaining a low tree depth.
    - The verification of files uploaded happens by means of 32 byte hash.
  - They also managed to integrate their implementation with dropbox.

❑ Merkle Hash Tree



- Hashes 0-0 and 0-1 are the hash values of data blocks L1 and L2, respectively.
- Hash 0 is the hash of the concatenation of hashes 0-0 and 0-1.

Research Paper I : File System Support Solution

- Analysis
  - The implementation described in this paper served as file store on top of DropBox(Cloud Store Application) where dropbox does not provide complete transparency.
  - DropBox was augmented with a merkle tree to keep track of file and intermediate node hashes in the tree making it the only component that had to be kept persistent on the client's local file system.
  - This system will be able to detect file tampering but will not be able to reproduce the original files from an adversarial server.

Research Paper I : File System Support Solution

- Analysis Continued
  - The paper lays deeper emphasis to file integrity and does not account for confidentiality and availability components of security in cloud systems.
  - This paper focuses on building a single user Dropbox client which builds a Merkle tree as metadata
  - The client is able to verify that files downloaded are the original files that were uploaded to Dropbox.
  - Usage of Merkle tree,ensures that the client only keeps the root hash in persistent memory, everything else is stored in the cloud.

Research Paper I : File System Support Solution

- Analysis Continued

  Operations:

  - Add/Upload
  - Delete
  - Download
  - Edit
  - Verify
  - List

  All operations support **O(nlogn)** time complexity.

Research Paper II : Design and Implementation of an Efficient Tool to Verify Integrity of Files Uploaded to Cloud Storage

❑ Analysis

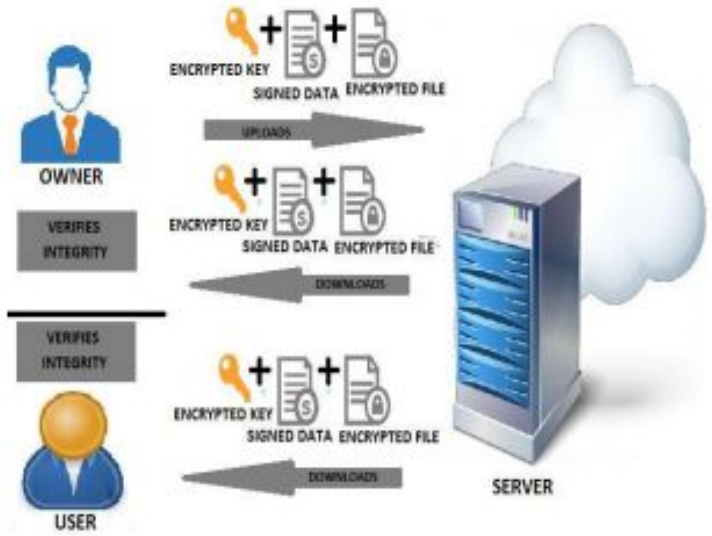


Fig. 2: System Overview

- The public key and private key are generated for digitally signing, using RSA based digital signature algorithm.
- The owner of the file performs encryption of the plaintext file using AES-256 algorithm using a symmetric key K.
- Key K is encrypted using RLWE encryption and it is saved in a file.
- MHT is generated for the encrypted file and hashed nodes is converted to hex format and is written to a file. This file is then signed using a private key.

Research Paper II : Design and Implementation of an Efficient Tool to Verify Integrity of Files Uploaded to Cloud Storage

❑ Analysis continued

- The owner uploads the following to the cloud :
  a. Encrypted file
  b. Encrypted file key K
  c. Signed MHT file.

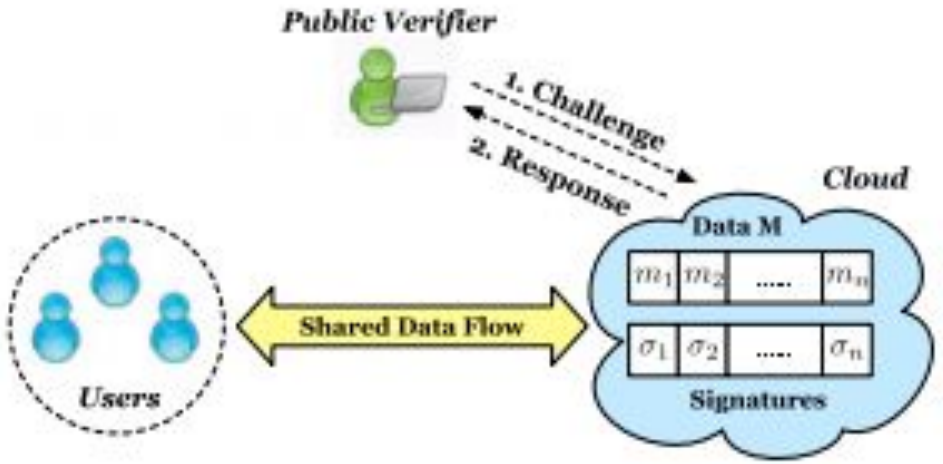
- Signature verification is done on MHT file and if it succeeds, it means MHT file hasn't been modified.

- Then user proceeds further to compare the uploaded MHT file with the calculated MHT file. If there is a match, the user ensures that the encrypted file has not been modified and proceeds further for decryption.

Research Paper III : PANDA : Public Auditing for Shared Data with Efficient User Revocation in the Cloud (2015)

- Abstract
  - The main focus of this paper:
    - To create a auditing mechanism for shared data.
    - Very efficient user revocation mechanism.
  - They have used the following concepts:
    - Digital signature.
    - Proxy re-signature scheme.
  - Users of the group can access, download and modify shared data.
  - By utilizing the idea of proxy re-signatures, cloud re-signs blocks that were previously signed by the revoked user.
  - This saves users from significant amount of computation and communication resources during user revocation.

- Analysis
  - Each block in shared data is attached with a signature, which is computed by one of the users in the group. Initially, original user computes the signatures on shared data.
  - Once a user modifies a block, this user needs to sign the modified block with his/her own private key. Thus different blocks maybe signed by different users.

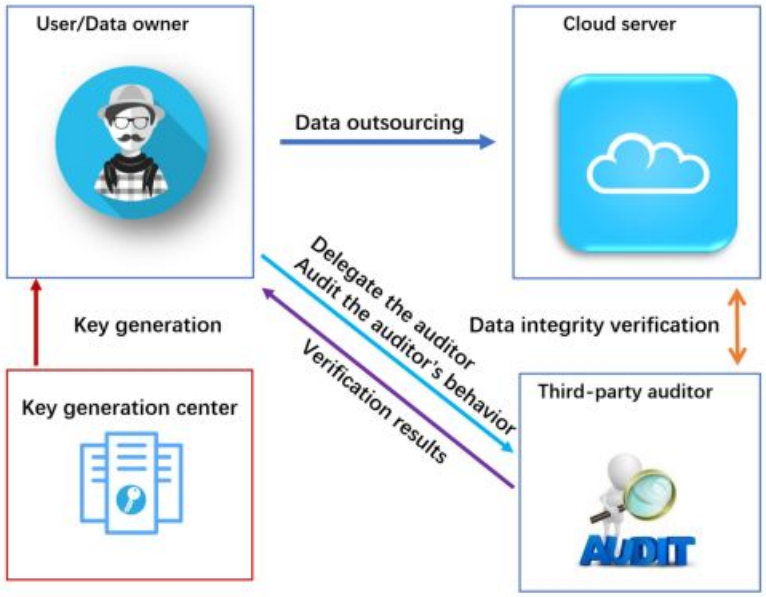


- Analysis continued [Panda]
  - ***Proxy Re-Signatures***
    - It allows a proxy to act as a translator of signatures between two users. It basically converts a signature of one user into signature of another user on the same data.
    - The proxy does not learn any signing key and also cannot sign arbitrary messages on behalf of users.
  - ***Efficient and Secure User Revocation***
    - The straightforward method is to allow an existing user to download the corresponding part of shared data and re-sign it. This is highly inefficient due to the large size of shared data in the cloud.
    - When a user is revoked, blocks that were signed by this user are resigned by the cloud using re-signing key. As a result, the efficiency of user revocation is significantly improved, and computation and communication resources of existing users is saved.

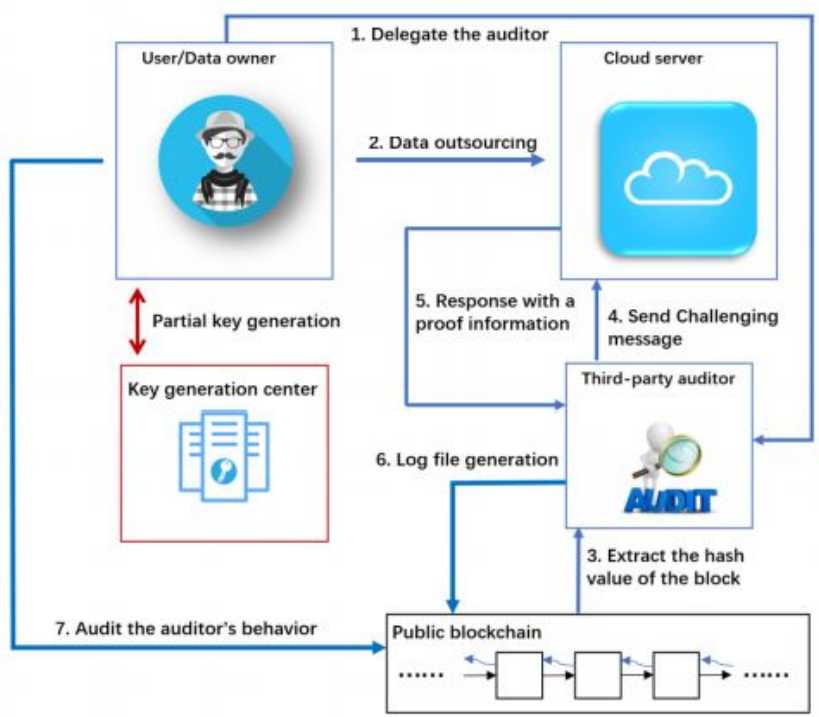
Research Paper IV : Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors

- Abstract
  - This paper proposes a certificateless public verification scheme against procrastinating auditors (CPVPA) by using blockchain technology.
  - Since transactions on the blockchain are time-sensitive, the verification can be time-stamped after the corresponding transaction is recorded into the blockchain, which enables users to check whether auditors perform the verifications at the prescribed time.
  - Moreover, CPVPA is built on certificateless cryptography, and is free from the certificate management problem.

Research Paper IV : Blockchain-Based Public Integrity Verification for Cloud Storage against Procrastinating Auditors



System model



Proposed solution

Suggestions from Review - 3

- All Good. Keep up the Good Work.

- Prof. Prasad B Honnavalli

Design Approach

- Hash values of the files are stored in the leaf nodes of the **merkle tree** and the internal nodes are calculated from the leaf. The user is only required to keep the hash value of the root.
- The upload and download operation time depends on the height of the tree, hence use of **B+ trees** decreases the time taken
- Software application runs on the **local system**, rather being hosted as a web-platform, this eliminates the possibility of integrity breach from our end, also prevents upload to our cloud application and then to the cloud storage opted by the user.

Design Constraints, Assumptions & Dependencies

Constraints:

- Single user interface
- Single cloud provider
- Requires cloud credentials of the user
- User initiates the integrity check
- Portability

Assumptions:

- Cloud Storage Platforms do not provide complete transparency.
- Users always make uploads only through their personal computer.

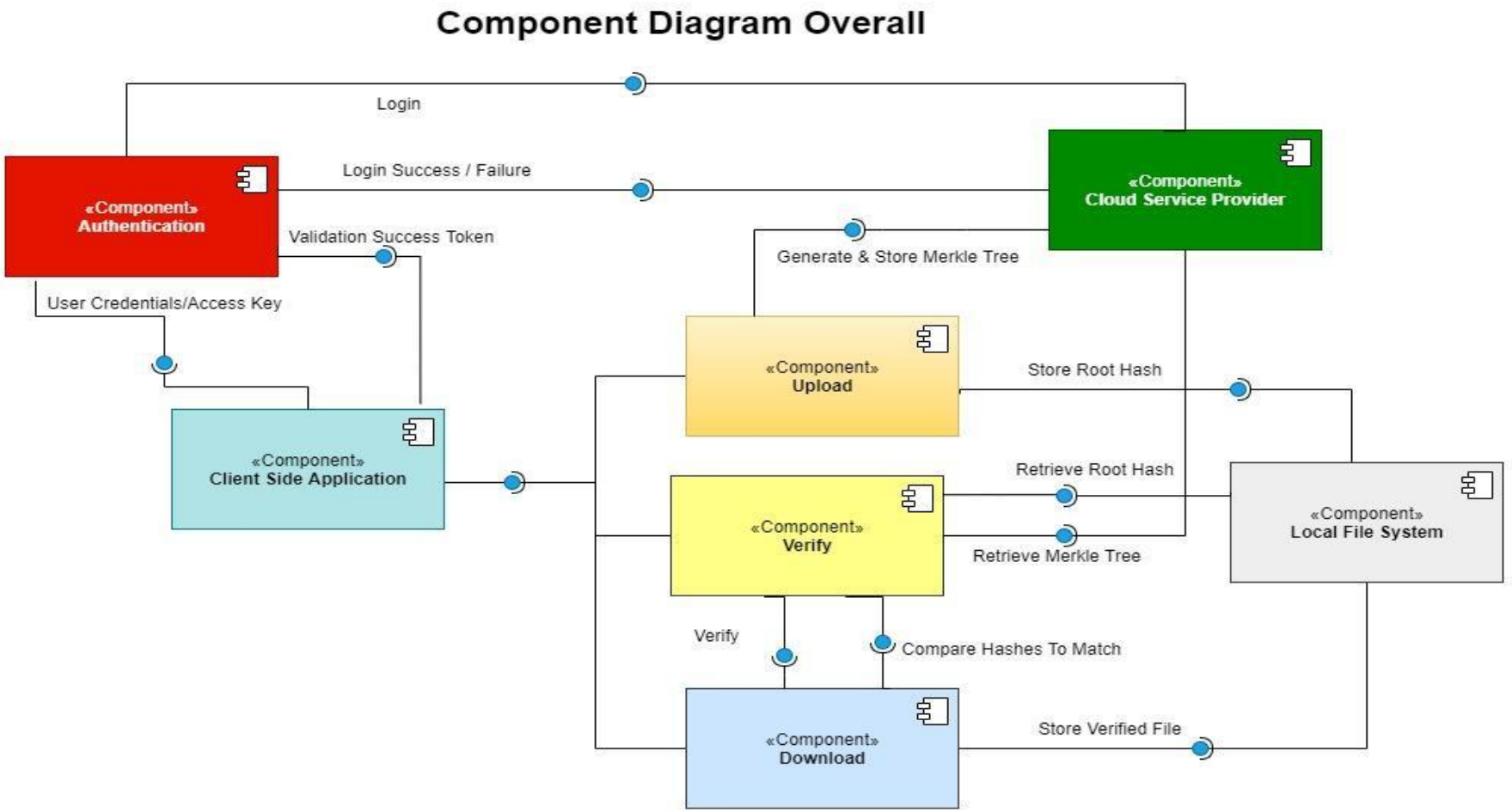
Dependencies:

- Open APIs of Cloud Storage Provider.
- User must provide his/her credentials.

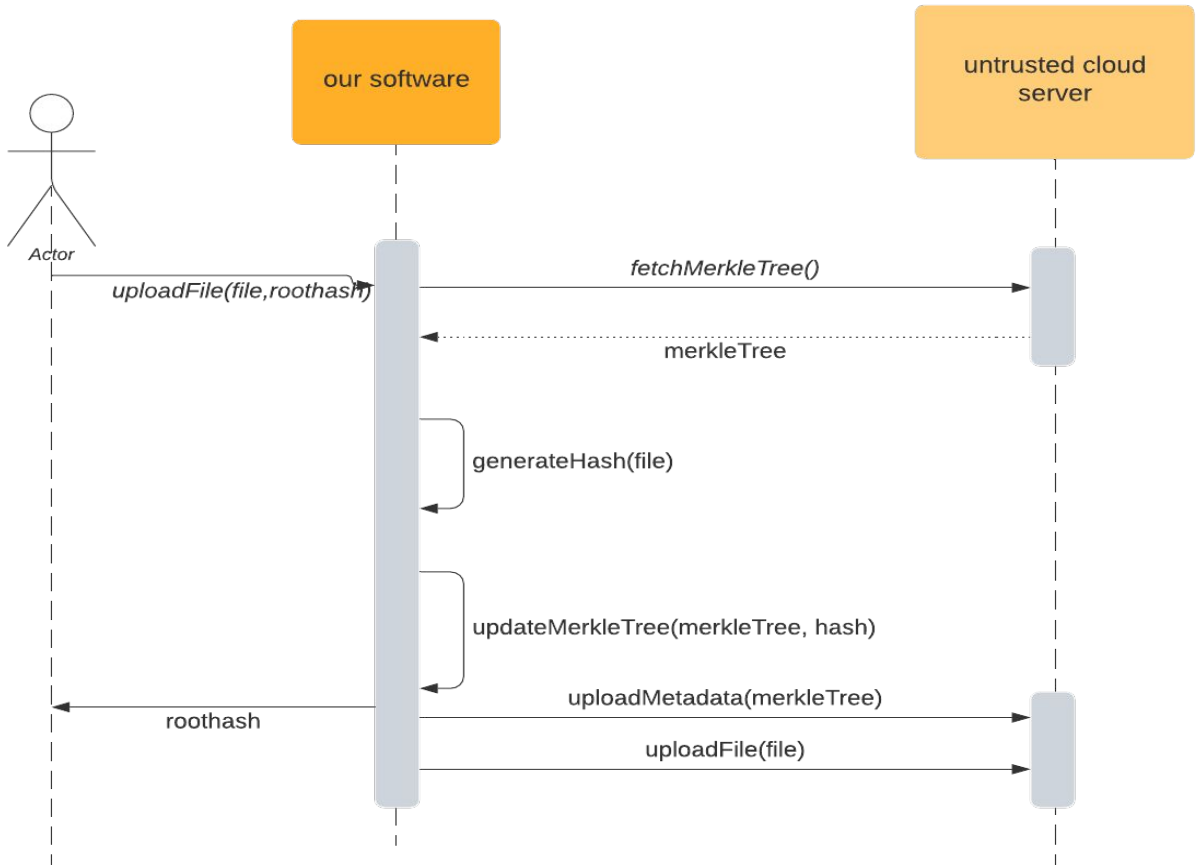
Proposed Methodology / Approach

- We intend to implement a software solution that ensures the integrity of the files hosted on the cloud by untrusted servers and support a range of cloud services like Google Drive, Dropbox, OneDrive, Mediafire, Mega etc.
- We intend to start our work on cloud provides which have good documentation and effective api's for developers to use.

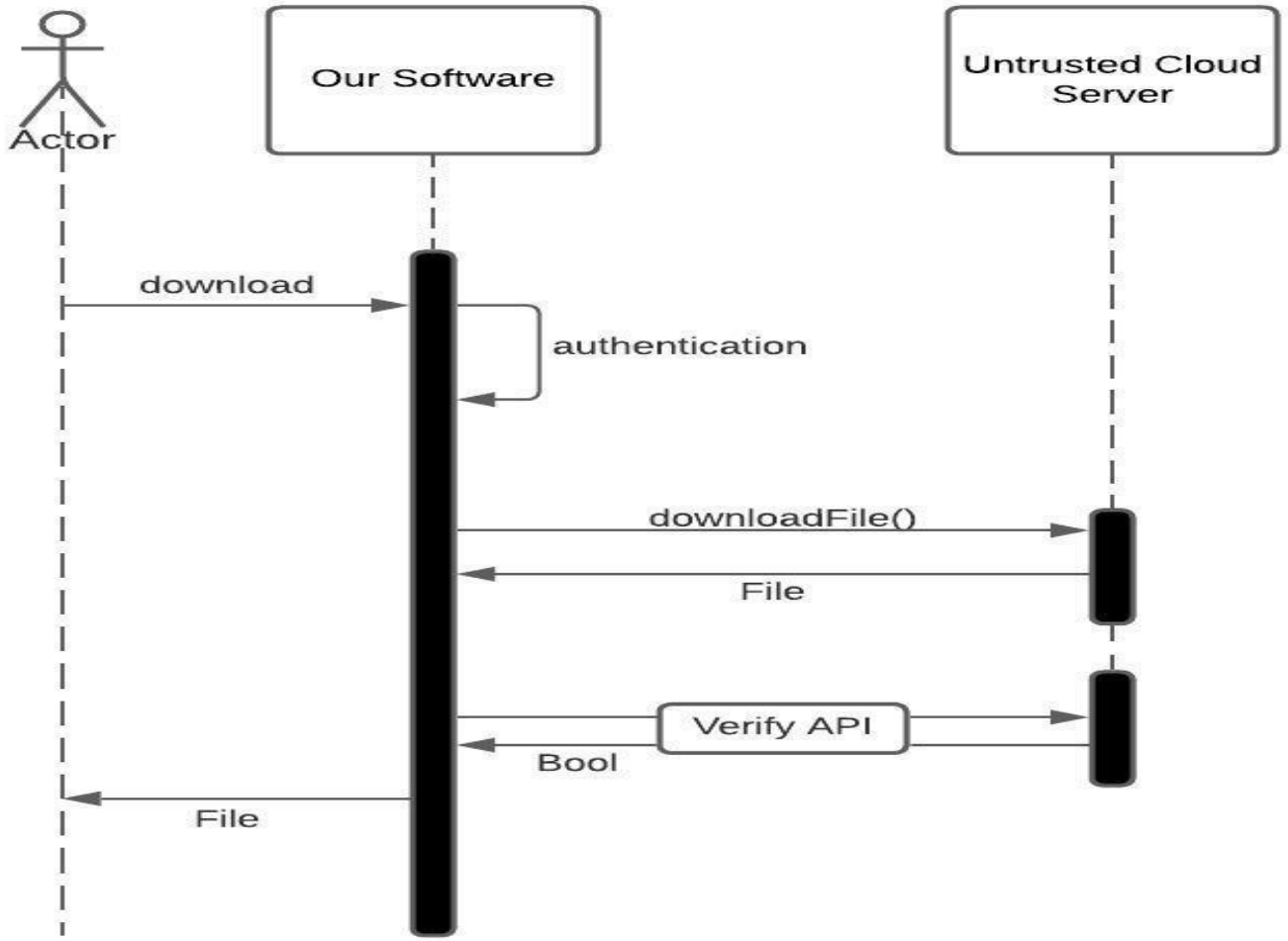
Component Diagram



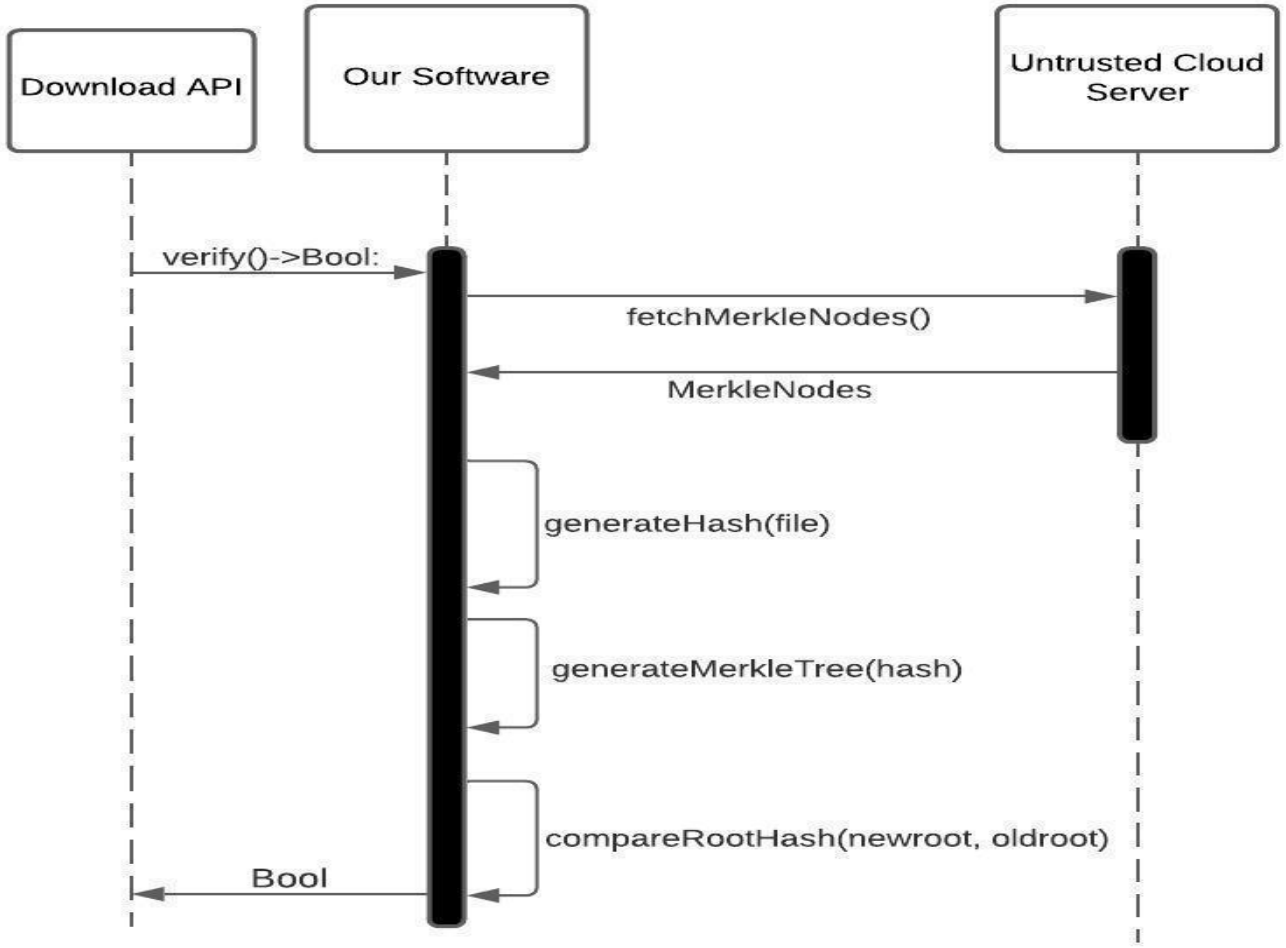
Sequence Diagram - Upload



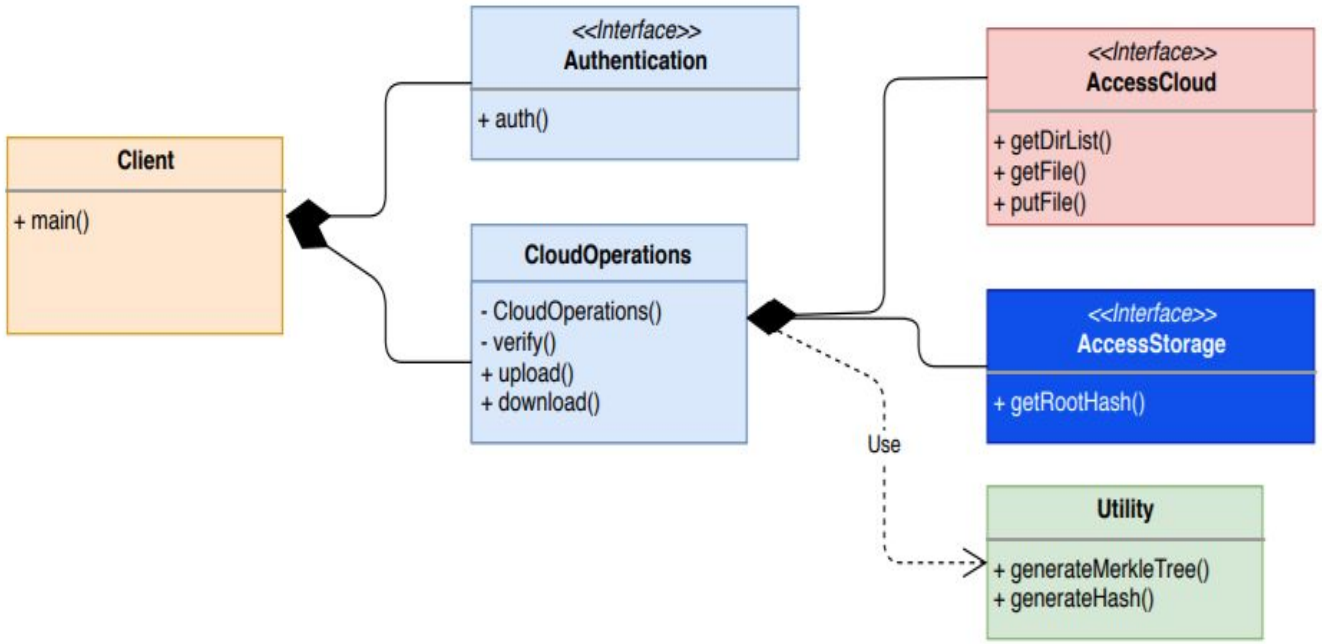
Sequence Diagram - Download



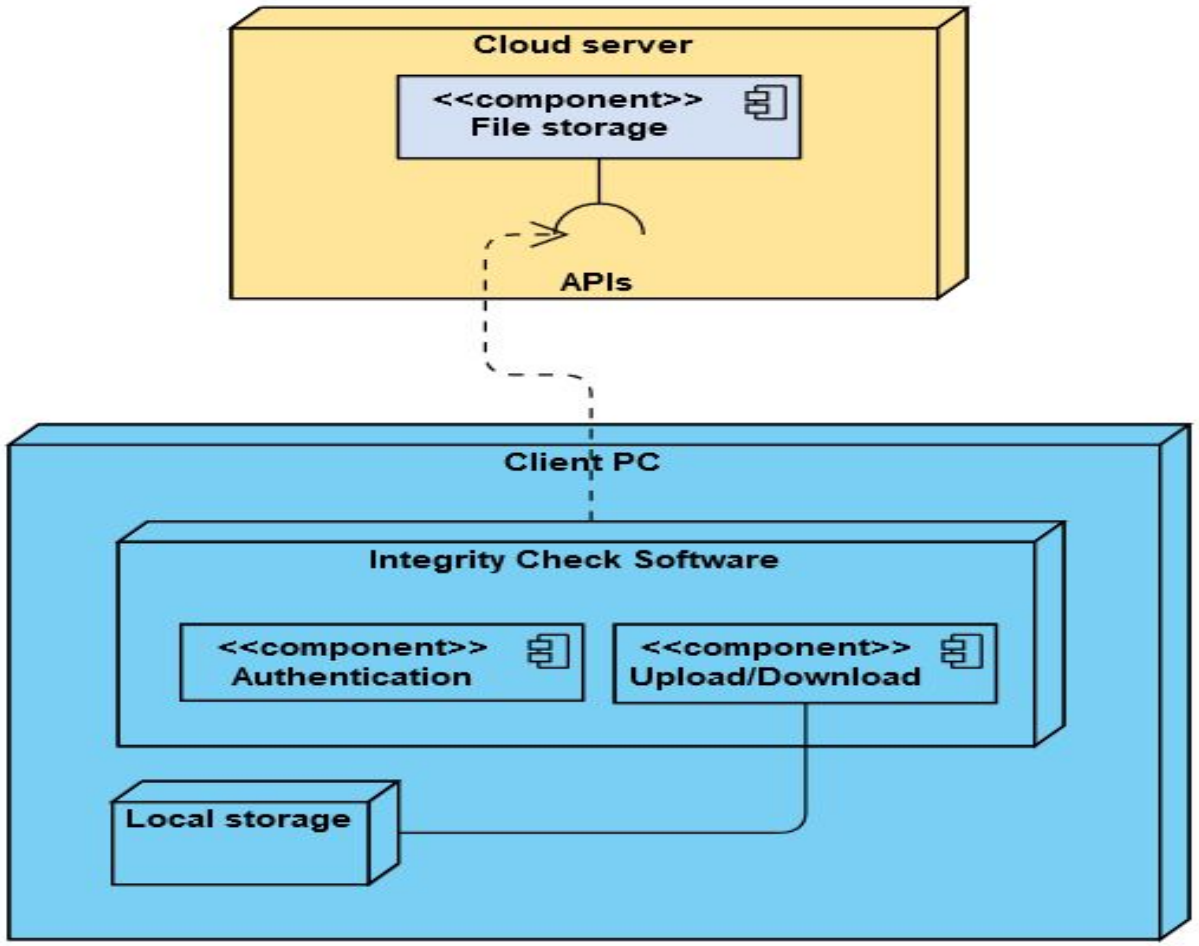
Sequence Diagram - Verify



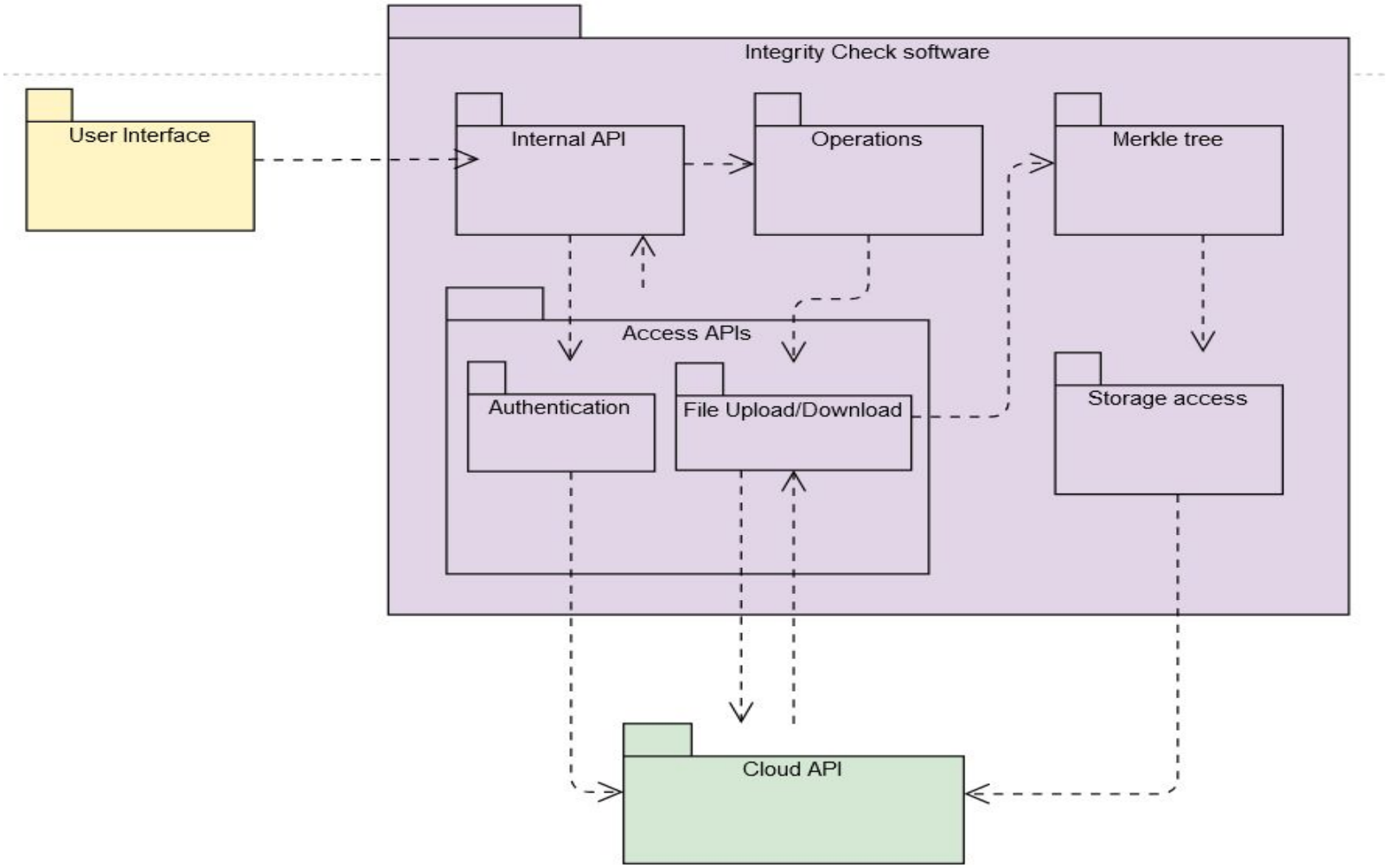
Master Class Diagram



Deployment Diagram



Packaging Diagram



ER Diagram

#	Entity	Name	Definition	Type
		ENTITIES		
1.	User	Client	Represents a client side user that interacts with the application interface.	User
#	**Attribute**	**Name**	**Definition**	**Type (size)**
		DATA ELEMENTS		
1.	Name	Name	Identifier	String
2.	Password	Auth Password	Authentication key specific to User.	Alphanumeric

#	Entity	Name	Definition	Type
		ENTITIES		
2.	File	File	File uploaded	User
#	**Attribute**	**Name**	**Definition**	**Type (size)**
		DATA ELEMENTS		
1.	RootHash	RootNodeHash	Root Node of the merkle tree which is used for verification.	Sha 256 Hash
2.	Merkle_Tree	Merkle_Tree	Merkle tree constructed to keep track of uploaded files and verify the files on download.	Merkle Tree Node

- Our Product is comprised primarily of 2 major entities the user's themselves and the files they upload.

User Interface

Login Portal

- A Login Portal used to authenticate a user.

File Upload

- The Upload portal will allow the user to select files to upload.
- A Drag and drop option will be available as well.
- Cancellation of upload will be provided to the user by means of a button.

File Download

- Users will be able to select a file to download based on the click of a button.
- The user will be alerted if in case the file has been tampered with.
- A pop up message warning the user as to whether he wants to go through with the download.

External Interface

The application relies on the open APIs provided by the cloud service provider. The application runs on top of the cloud service provider hence depends on their interface for uploading and downloading files.

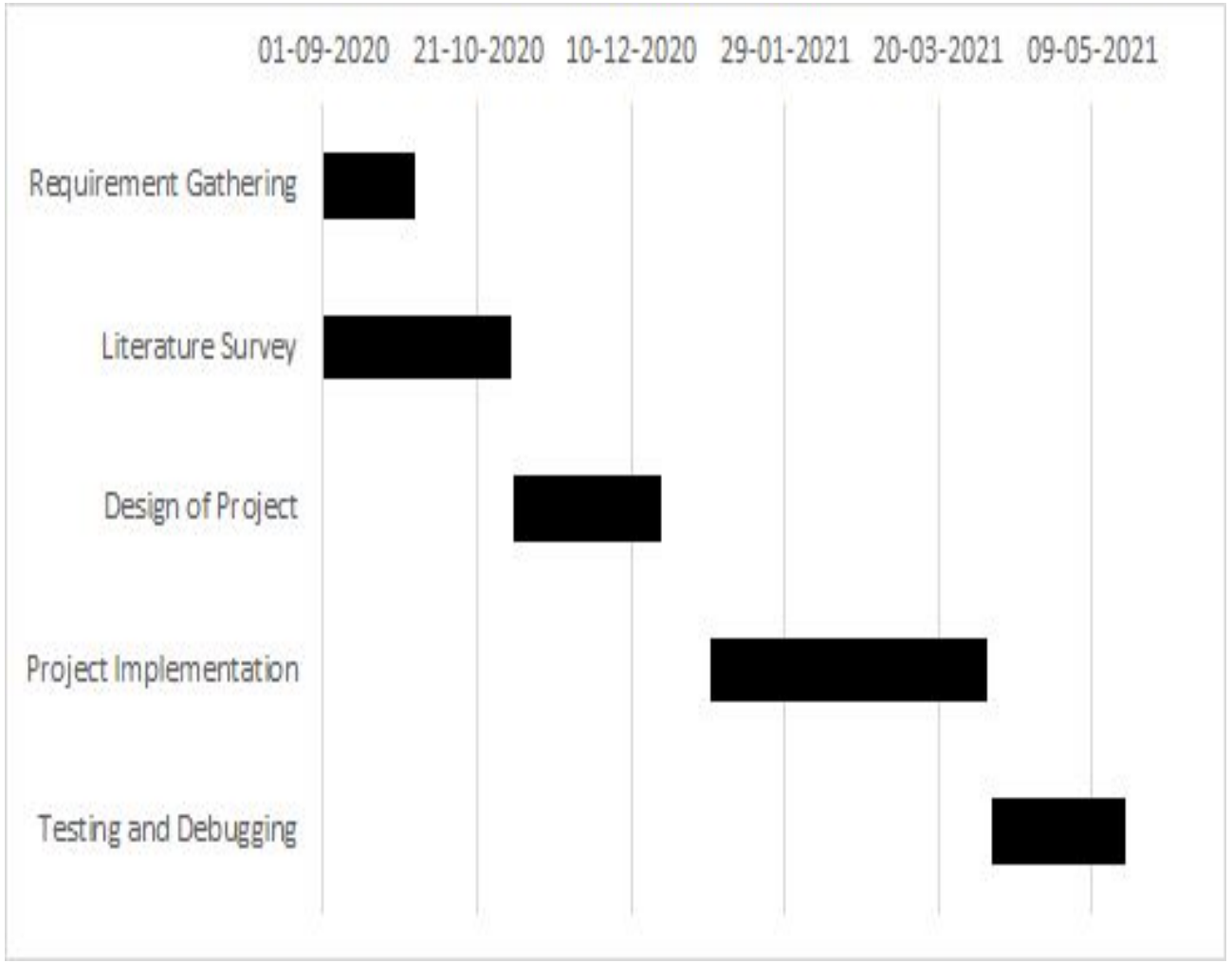
Report Layouts

- Our product does not generate any specific report but would indicate to the user that the file has been uploaded successfully or that the file to be downloaded has been tampered or not but does not account for retrieval of the tampered file if any.
- Our product does not say that integrity will be preserved but it assures the user that the file they receive is authentic but it does not add any extra features to ensure authenticity of files.

Technologies Used

- Draw.io
  - Designing high level diagrams
- Python
  - To construct command line interface
  - To construct necessary Classes and Objects
- Python-requests
  - To access Cloud Storage APIs
- Python-qt5
  - To build Graphical User Interface

Project Progress



- We have analysed all requirements and performed a thorough literature survey to determine feasibility of the project.
- We have also completed the high level design of the project.
- The Project is 40% done we need to implement all the api's mentioned in the high level design and perform testing.

References

- Joannas, Msands, Brishima(File) System Support Solution.
- Dalia Attas, Omar Batrafi, “Efficient integrity checking technique for securing client data in cloud computing”.
- Emil Stefanov, Marten Van Dijk, Alina, Ari Juels, “Iris: A Scalable Cloud File System with Efficient Integrity Checks”
- FileZilla Pro : FileZilla Pro
- Kevin D. Bowers, Ari Juels, Alina Oprea ,“HAIL: A High-Availability and Integrity Layer for Cloud Storage”
- Boyang Wang, Baochun Li, Hui Li,“Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud”

References

- Priyansi Parida, Snehalata Konhar, Bharati Mishra, Debasish Jena, “Design and Implementation of an Efficient Tool to Verify Integrity of Files Uploaded to Cloud Storage.”
- K.Kireeti, B.V. Kiranmayee, S.Nagini, “An Efficient Secure Protocol for integrity checking of data files outsourced to remote server”

Any other information

Considerations for additional Features

- Blockchain Implementation for Portability
- Multi User Interface
- Offline Features

Thank You