PES
UNIVERSITY

PROJECT REQUIREMENTS SPECIFICATION

Layer 7 Integrity Check For Cloud Service Providers (L7ICCSP)

UE17CS490A – Capstone Project Phase – 1

Submitted by:

R Siva Girish	**PES1201700159**
Gaurav C.G	**PES1201700989**
Sangam Kedilaya	**PES1201701139**
Zenkar Srinivas	**PES1201701532**

Under the guidance of

Prof. Prasad B Honnavalli
Professor, Computer Science and Engineering
Director, Centre for Information Security,
Forensics and Cyber Resilience (C-ISFCR)
Director, Centre for Internet of Things (C-IoT)

August - December 2020

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

TABLE OF CONTENTS

1. Introduction

Our product does not need any explicit data to analyse but only the cloud credentials of the user to login to their cloud account.The product deals with verifying the integrity of the data downloaded from the user’s cloud.A User uploads a file to the cloud and a certain amount of metadata is generated for each file while uploading and stored on the user’s cloud and the root hash value of the uploaded file is stored on the local file system. This data stored on the local file system is vital for the integrity check.

1.1. Project Scope

This work aims to make it easier for layman users to verify the integrity of files they download from the cloud service providers. The project seeks to provide an interface for users to upload and download files and run integrity checks on the files they download from the cloud.

2. Product Perspective

The product aims to provide users with a means to verify the integrity of the files they download with a minimal overhead to the existing procedure coupled with an easy to use interactive interface.

2.1. Product Features

2.1.1. The user is able to upload and download files through our application.

2.1.2. The user is able to verify the integrity of the files downloaded.

2.1.3. The user is able to check which file has been tampered with.

2.1.4. The user is authenticated before giving access to download and upload files.

2.1.5. The application generates hashes and creates node entries for all user uploads.

2.1.6. Provide an easy-to-use interface for the user.

2.2. User Classes and Characteristics

End User: Upload/download files from cloud storage platform

2.3. Operating Environment

The software can be built and run on any OS that supports Python 3.8 and above.

2.4. General Constraints, Assumptions and Dependencies

2.4.1 Constraints

1. Single user interface
2. Single cloud provider
3. Requires cloud credentials of the user
4. User initiates the integrity check

2.4.2 Assumption

1. Cloud Storage Platforms do not provide complete transparency.

2.4.3 Risks

1. Novelty of the problem chosen. No extensive work has been done on integrity verification till now.

3. Functional Requirements

- The user is able to upload and download files through our application.
- The user is able to verify the integrity of the files downloaded.
- The user is able to check which file has been tampered with.
- The user is authenticated before giving access to download and upload files.
- The application generates hashes and creates node entries for all user uploads.

4. External Interface Requirements

4.1. User Interfaces

Login Portal

- A Login Portal used to authenticate a user.
- Authenticated users will be able to make use of the application.

File Upload

- The Upload portal will allow the user to select a file 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.

4.2. Hardware Requirements

Standard x86-64 machine.

4.3. Software Requirements

Name	Description	Version	Type
Cloud Services	Cloud storage Provider	NA	Cloud Storage Platform
Operating System	Platform Independent	NA	Operating System
Python	Programming Language	>= 3.6	Programming Language Runtime
MongoDB	Database	>=3.6	Database

Table 1 : Software Requirements

4.4. Communication Interfaces

All communications occur over HTTP/HTTPs requests.

5. Non-Functional Requirements

5.1. Performance Requirements:

- Performance : time taken for verification, file upload and download should be optimum
- Reliability : detection of breach in integrity
- Interoperability : between cloud platforms and the software
- Portability : usability of the software on different OS

5.2. Safety Requirements:

- Access tokens of the user needs to be safely cached and needs to be removed after a certain duration to prevent non-users from accessing previous user's data.

5.3. Security Requirement:

- Confidentiality : only authorized users can access their respective files
- Availability : the user should be able to perform the integrity check at any given time
- Integrity: Files required by the application should be encrypted and stored securely and should not be easily accessible for common users.

Appendix A: Definitions, Acronyms and Abbreviations

Merkle Tree : Merkle tree is a tree in which every leaf node is labelled with the cryptographic hash of a data block, and every non-leaf node is labelled with the cryptographic hash of the labels of its child nodes.

Sha256 : Sha 256 is a cryptographic hash function whose size is exactly that of 32 bytes.

Appendix B: References

1. Joannas, Msands, Brishima, "(File) System Support Solution".
2. Dalia Attas, Omar Batrafi, "Efficient integrity checking technique for securing client data in cloud computing".
3. Emil Stefanov, Marten Van Dijk, Alina, Ari Juels, "Iris: A Scalable Cloud File System with Efficient Integrity Checks"
4. FileZilla Pro : FileZilla Pro
5. Kevin D. Bowers, Ari Juels, Alina Oprea ,"HAIL: A High-Availability and Integrity Layer for Cloud Storage"
6. Boyang Wang, Baochun Li, Hui Li,"Panda: Public Auditing for Shared Data with Efficient User Revocation in the Cloud".
7. Packaging Diagram Tutorial
8. Deployment Diagram Tutorial
9. Sequence Diagram Tutorial
10. Cryptographic Hashing Functions -SHA-256

11. Cryptographic Hashing Functions - MD5
12. RSA (cryptosystem)
13. Symmetric key encryption