INTEGRITY AUDITING BASED ON THE KEYWORD WITH SENSITIVE INFORMATION PRIVACY FOR ENCRYPTED CLOUD DATA

A PROJECT REPORT

Submitted by

DHILIP KUMAR D	**422419205008**
DHINA P	**422419205009**
YUVARAJ S	**422419205043**

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM

ANNA UNIVERSITY:: CHENNAI 600 025

MAY 2023

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report **“INTEGRITY AUDITING BASED ON THE KEYWORD WITH SENSITIVE INFORMATION PRIVACY FOR ENCRYPTED CLOUD DATA”** is the bonafide work of **“DHILIP KUAMR D (422419205008), DHINA P (422419205009), YUVARAJ S (422419205043)”** who carried out the project work under my supervision.

SIGNATURE

Dr.S.MILTON GANESH, M.E.,Ph.D.,

HEAD OF THE DEPARTMENT & SUPERVISOR

Department of Information Technology

University College of Engineering, Melpakkam.

Tindivanam – 604 001.

Submitted for the University Examination held on ___________________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

ACKNOWLEDGMENT

We would like to thank our revered Dean, **Dr.P.Thamizhazhagan M.E., Ph.D.,** For providing infrastructure facilities and whole hearted encouragement for completing our project successfully.

We would like to express our sincere gratitude and pay our grateful acknowledgement to **Dr.S.Milton Ganesh, M.E., Ph.D.,** HoD & Assistant Professor Department of Information Technology, University College of Engineering Tindivanam. We are extremely thankful for his unwavering support and for extending the facilities of the department towards our project. Throughout the project, Dr. Ganesh served as our guide, providing guidance and assistance in every phase. We greatly appreciate his thoroughness, tolerance, and willingness to share his knowledge with us. His approachability and thoughtfulness have been invaluable, and we are indebted to him for harnessing our potential and bringing out the best in us. Without his immense support at every step of the way, we would not have been able to achieve what we have accomplished.

We thank all our teaching and non-teaching faculty members, our class advisor **Dr.R.Shymala, M.E., Ph.D.,** Department of Information Technology, University College of Engineering Tindivanam. And also our fellow friends helping us in providing valuable suggestions and timely ideas for the successful completion of the project.

We extends our thanks to our family members, who have been a great source of inspiration and strength to us during the course of this project work. We sincerely thank all of them.

ABSTRACT

The public cloud data integrity auditing technique is used to check the integrity of cloud data through the Third Party Auditor (TPA). In order to make it more practical, we propose a new paradigm called integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. This paradigm is designed for one of the most common scenario, that is, the user concerns the integrity of a portion of encrypted cloud files that contain his/her interested keywords. In our proposed scheme, the TPA who is only provided with the encrypted keyword, can audit the integrity of all encrypted cloud files that contain the user's interested keyword. Meanwhile, the TPA cannot deduce the sensitive information about which files contain the keyword and how many files contain this keyword. These salient features are realized by leveraging a newly proposed Relation Authentication Label (RAL). The RAL can not only authenticate the relation that files contain the queried keyword, but also be used to generate the auditing proof without sensitive information exposure. We give concrete security analysis showing that the proposed scheme satisfies correctness, auditing soundness and sensitive information privacy. We also conduct the detailed experiments to show the efficiency of our scheme.

Dhilip kuamr D

Dhina P

Yuvaraj S

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF ABBREVATIONS

ABBREVATIONS	FULL FORMS
PBC	Pairing Based Cryptography
EEC	Elliptic Curve Cryptography
RAL	Relational Authentication Label
BM	Bilinear Mapping
CSP	Cloud Services Provider
TPA	Third Party Auditor

CHAPTER 1

INTRODUCTION

1.1 PAIRING BASED CRYPTOGRAPHY

Pairing-Based cryptography is based on pairing function that map pairs of points on an elliptic curve into finite field. It has been recently discovered that some cyclic groups that cloud be used in Cryptography admit a special bilinear pairing map that introduces extra structure to the group. Bilinear pairing maps were first used to break cryptosystems and later it was realized that the extra structure could be exploited to build cryptosystems with extra properties. Boneh and Franklins identity – based encryption scheme is the most famous early example of what could be achieved using bilinear maps. After that, a plethora of cryptosystems have been designed using bilinear maps. No full and freely available implementation of pairing based cryptography was available until this work.

Recent proposals fall short of this goal as either their source code is not available or curve. Moreover, neither one of implements preprocessing that is crucial to reduce the computation time. In this work, we present JPBC a java port of the PBC library written in C.

JPBC provides a full ecosystem of interfaces and classes to simplify the use of the bilinear maps even for a non-cryptographer. JPBC supports different types of elliptic curves, preprocessing which can speed up the computation significantly and it is ready for the mobile world. More over a benchmark comparison between JPBC and PBC has been performed to measure the gap between the two libraries. Further more JPBC has been benchmark on different Android mobile platforms.

A Port of the Pairing-Based Cryptography Library(PBC), library developed by Ben Lynn, to performs the mathematical operations underlying – based cryptosystems directly in java.

A Wrapper that enables the delegation of the pairing computation of multilinear maps over the integers by Coron Lepoint, and Tibouchi. The implementation supports multithreading and uses memory mapped files to save in primary memory requirements.

1.2 ELLIPTIC CURVE CRYPTOGRAPHY

1.2.1 Elliptic Curve

An elliptic curve is a plane algebraic curve defined by an equation of the form, $y^2 = x^3 + ax + b$,

which is non-singular (has no self-intersections or cusps). The curve is non-singular if the discriminant $4a^3 + 27b^2 \neq 0$. This is called the Weierstrassequation for an elliptic curve. We always take a, b, x and y to be elements of afield such as R, Q, C or finite field Fq where q = p n with p prime and $n \geq 1$. IfK is a field with $a, b \in K$, then the elliptic curve is said to be defined over K.The point (x, y) on the elliptic curve with $x, y \in K$ is called a K-rational point. The elliptic curve $E : y^2 = x^3 + ax + b$ over R has the following general forms:

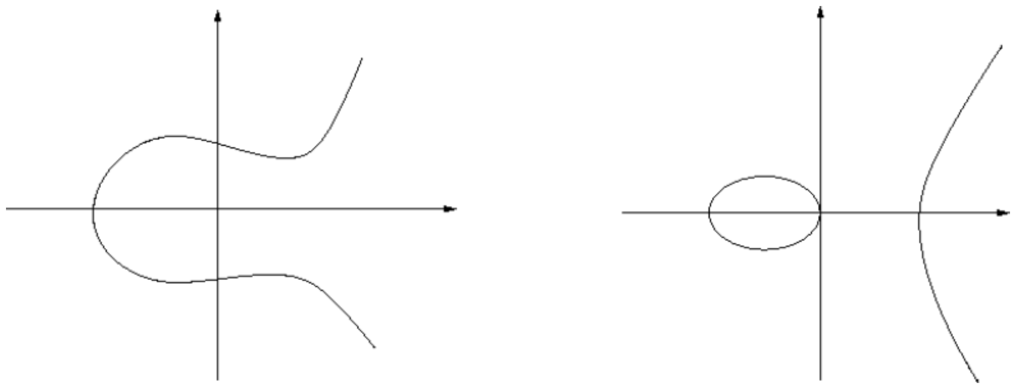


Figure 1.2.1 : Elliptic Curve General Forms

1.2.2 Group Law

One of the most important properties of elliptic curves is the existence of agroup law for adding points on the curve. The group law on the rational points on an elliptic curve is defined using the intersection of straight lines with the curve.

Consider two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$, $P_1 \neq P_2$ on the elliptic curve E given by $y^2 = x^3 + ax + b$. Draw a line L through P_1 and P_2 which intersects E in a third point P'_3 . Reflect P'_3 across the x-axis to obtain $P_3 = (x_3, y_3)$. Define $P_1 + P_2 = P_3$.

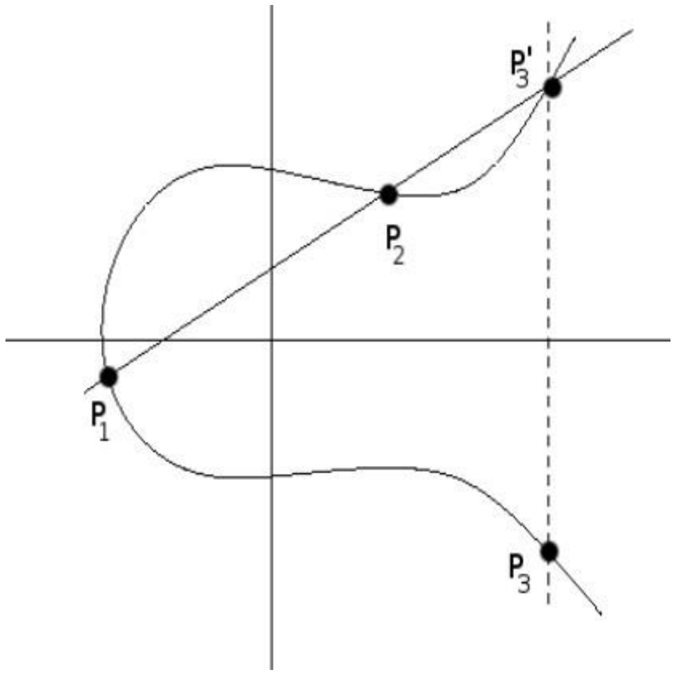


Figure 1.2.2 Group Law in Elliptic Curve

1.2.3 Public Key Cryptography

In private key cryptography, since the same key is used for both encryption and decryption, the two parties must take great care in exchanging the key so that an eavesdropper does not obtain it. The problem of key exchange is one of the most difficult in symmetric cryptosystems. Symmetric cryptosystems create problems of trust, as the same key is held by both users in any sender/receiver pair. In addition to being cryptanalytically secure, a cryptosystemshould satisfy the following:

1. Authentication: The recipient of a message should be able to ascertain itsorigin.
2. Integrity: The recipient of a message should be able to determine that themessage has not been modified during transit.
3. Non-Repudiability: A sender of a message should not be able to deny laterthat he sent the message.

These requirements are impossible to achieve in classical one-key cryptosystems. The breakthrough came in 1976 with the invention of public keycryptography by Stanford University Professor Martin Hellman and graduate student W. Diffie. By definition, a public key cryptosystem has the property thatthe enciphering function $f : P \rightarrow C$ is easy to compute once the enciphering key,KE, is known, but it is very hard in practice to compute the inverse function $f^{-1}: C \rightarrow P$, that is, the function f is not invertible (without some additional information - the deciphering key KD). Such a function f is called a trapdoor function.

1.2.4 Discrete Logarithm Problem

If G is a finite group, $b \in G$ and y is an element of G which is a power ofb, then the discrete logarithm of y to the base b is any integer x such that $b^x = y$.

The discrete logarithm of y is not unique as it can only be found modulothe order of b in G. If G is a finite cyclic group and b is a generator of G, then $|G| = |b|$. Let N be the order of G, then

$$\log_b y \equiv x \pmod{N}.$$

All methods designed for computing discrete logarithms requireexponential time, with the fastest requiring $O(\sqrt{N})$ time where N is the order of the group. The discrete logarithm problem is important because of its wide use incryptography. The problem of computing the discrete logarithm was just a mathematical curiosity until 1976 when Diffie and Hellman prescribed a methodof exchanging cryptographic keys which relies on the difficulty of the discrete logarithm problem for its security.

1.2.5 Elliptic Curve Cryptography

Elliptic Curve Cryptography (ECC) was discovered in 1985 by VictorMiller (IBM) and Neal Koblitz (University of Washington) as an alternative mechanism for implementing public key cryptography. ECC is emerging as an attractive public key cryptosystem for mobile/wireless environments. Comparedto traditional cryptosystems like RSA, ECC offers equivalent security with smaller key sizes, which results in faster computations, lower power consumption, as well as memory and bandwidth savings. This is specially usefulfor mobile devices which are typically limited in terms of their CPU, power and network connectivity.

1.2.6 Elliptic Curve Discrete Logarithm Problem

The elliptic curve discrete logarithm problem is the foundation of muchof present-day ECC. It relies on the natural group law on a non-singular ellipticcurve which allows one to add points on the curve together.

Given an elliptic curve E over a finite field Fq, and a point Q on E otherthan O, the discrete logarithm problem on E to the base Q is the following: givena point P in E (Fq) \ {O}, find an integer n such that nQ = P, if such an integer exists. Let E be an elliptic curve given by y 2 = x 3 + x + 1 over F7. We can show (as in Chapter 4 Example 4.1.1) that, E (F7) = {O,(0, 1),(0, 6),(2, 2),(2, 5)}

If Q = (2, 2) and P = (0, 6), it can be shown that 3Q = P. Hence n = 3 isa solution to the discrete logarithm problem.

#E (Fq). The order of E (Fq) needs not to be smooth because any abelian groupcan be decomposed as a direct sum of cyclic subgroups. The discrete logarithmin the group is then a problem in each of the cyclic subgroups. If all the cyclic subgroups are small ("smooth"), then the discrete logarithm is easy to handle. Hence we always choose the elliptic curve such that the order of E (Fq) is not "smooth". It is because of these reasons that the subject of counting points on an elliptic curve is very important in the study of elliptic curves.

1.3 BILINEAR MAPPING

A Bilinear operator is a function combining elements of two vector spaces to yield an element of a third vector space that is linear in each of its arguments.

Assume G1 and G2 are the two cyclic groups of the same prime order q. A map e : G1 × G1 → G2 is said to be a bilinear pairing. Bilinear e: G1*G1 →G2. G1, G2 is a two generators

Computability: It is efficient to compute this map.

Non-degeneracy : e(g,g) $\neq$1 for a generator g € G1.

Bilinearity: Given a; b € Zq* and u; v € G1, $e(u^a, v^b) = e(u,v)^{ab}$

1.4 INTEGRITY AUDITING

Integrity audits explore the moral and ethical standards of a company’s practice. Identify flags that incite such audits, the criteria of their investigations, and streps in the process demonstrated through real- world examples.

1.5 OBJECTIVE

The proposed scheme aims to address the challenges of auditing the integrity of specific encrypted cloud files that contain a user's interested keyword while maintaining sensitive information privacy, such as not disclosing which files contain the queried keyword or the number of files containing the queried keyword.

1.6 SCOPE

Scope of our project is to address the challenges of auditing the integrity of specific files in the cloud, without the need to audit all files, and without exposing sensitive information to the Third Party Auditor (TPA). The proposed scheme utilizes a new label called Relation Authentication Label (RAL) to achieve auditing of encrypted cloud files containing specific keywords while maintaining sensitive information privacy.

1.7 FEASIBILITY STUDY

A feasibility analysis usually involves a through assessment of the operational needs, financial and technical aspects of a proposal. Feasibility study is the test made whether the user needs may be satisfied using the current software and hardware technologies, whether the system will be budgetary constraints.

A feasibility study should be relatively cheap and done at the earlier possible time. Depending on the study, the decision can be made whether to go ahead with detailed analysis.

When a new project is proposed, it normally goes through feasibility assessment. Feasibility study is carried out to determine whether the proposed system is possible to develop with available resource and what would be the cost consideration. Fact's considerations in the feasibility analysis ware,

- Technical Feasibility
- Economic Feasibility
- Operational Feasibility

1.7.1 Technical Feasibility

Technical feasibility is intended to see that technical requirements of a project are met. Any system developed must not have a high demand on available technical resources. This will lead to high demands being placed on the client.

The technical requirements are the compared to the technical capability of the organization. The project is considered feasible if the internal capability is sufficient to support the project requirements.

1.7.2 Economic Feasibility

This study is carried out to check the economic impact that the system will have on the organization. The amount of the fund that the company can pour into the research and development of the system as well within the budget and this was achieved because most of the technologies used are freely available.

- Improved resulting over the existing method in terms of accuracy timeliness.
- Cost consumption.
- Estimate on the life expectancy of the hardware.
- Overall objective.

1.7.3 Operational Feasibility

Since, java and Intellij Idea are used as a base platform to develop the project, interoperability is high and hence operational feasibility is achievable. Java and Intellij Idea run efficiently in windows platform. It can be implemented in all kinds of environments and so the constraints are found to be less.

CHAPTER 2

LITERATURE SURVEY

Xiang Gao, Jia Yu, Yan Chang, Huaqun Wang and Jianxi Fan “Integrity Auditing Based On The Keyword With Sensitive Information Privacy For Encrypted Cloud Data” The public cloud data integrity auditing technique is used to check the integrity of cloud data through the Third Party Auditor (TPA). In order to make it more practical, we propose a new paradigm called integrity auditing based on the keyword with sensitive information privacy for encrypted cloud data. This paradigm is designed for one of the most common scenario, that is, the user concerns the integrity of a portion of encrypted cloud files that contain his/her interested keywords.

Jia Yu proposed the scheme in which TPA who is only provided with the encrypted keyword, can audit the integrity of all encrypted cloud files that contain the user’s interested keyword. Meanwhile, the TPA cannot deduce the sensitive information about which files contain the keyword and how many files contain this keyword. These salient features are realized by leveraging a newly proposed Relation Authentication Label (RAL). The RAL can not only authenticate the relation that files contain the queried keyword, but also be used to generate the auditing proof without sensitive information exposure.

Huaqun Wang and Jianxi Fan they give a concrete security analysis showing that the proposed scheme satisfies correctness, auditing soundness and sensitive information privacy. We also conduct the detailed experiments to show the efficiency of our scheme.

CHAPTER 3

REQUIREMENT SPECIFICATION

3.1 SYSTEM REQUIREMENTS

The software requirement specification is acquired at the completion of analysis phase. The function and performance allocated to software in system engineering is elaborated by complete information description of software, performance evaluation, design constraints and appropriate validation criteria.

3.1.1 Functional Requirements

The requirements specification is a technical specification of requirements for the software products. It lists the requirements of a particular software system including functional, performance and optional and an administrative perspective.

3.1.2 Hardware Requirements

Hard Disk	:	10 GB and Above
RAM	:	4 GB and above
Processor	:	Intel i3 and above

3.1.3 Software Requirements

IDE	:	InteliJ IDEA
Operating System	:	Windows 10 or Above
Coding Language	:	JAVA

CHAPTER 4

SYSTEM DESIGN

4.1 EXISTING SYSTEM

Existing system is the traditional cloud data integrity auditing technique, which involves checking the integrity of all cloud files periodically through the Third Party Auditor (TPA). The TPA charges users according to the workload of auditing services it provides, and the more cloud files are audited, the more money the user needs to pay. This can bring a heavy economic burden on the user, especially as data held by each user is expected to be up to 5200 GB in 2020.

4.2 ADVANTAGE AND DISADVANTAGE

Advantages

- Cost-effective auditing of specific files
- User's sensitive information is kept private
- Uses Relation Authentication Label (RAL) for auditing proof
- Efficient auditing for large-scale cloud files
- Intact storage of queried files

Limitations

- TPA cannot know which files contain the keyword
- Security may be compromised if RAL is not designed properly
- Malicious cloud could provide a valid proof with incorrect files

4.3 PROPOSED SYSTEM

The proposed system leverages a newly proposed Relation Authentication Label (RAL) to authenticate the relation between files containing the queried keyword and generate auditing proof without exposing sensitive information.

The user provides the Third Party Auditor (TPA) with only the encrypted search trapdoor (keyword), and the TPA can audit the integrity of all encrypted cloud files containing that specific keyword.

4.4 SYSTEM ARCHITECTURE

The proposed scheme is designed to allow users to check the integrity of specific encrypted files in the cloud without revealing sensitive information about the files' content or location. The scheme involves a third-party auditor (TPA) that verifies the integrity of the files without knowing which files contain the queried keyword or how many files contain it.

The proposed scheme uses a novel form of label called a Relation Authentication Label (RAL) to authenticate the relationship between files containing the queried keyword and generate an auditing proof without revealing sensitive information. The RAL is generated by the user and stored in the cloud along with the encrypted files. When the user wants to audit the integrity of files containing a specific keyword, they provide the TPA with a search trapdoor that allows the TPA to retrieve the RALs for files containing the keyword.

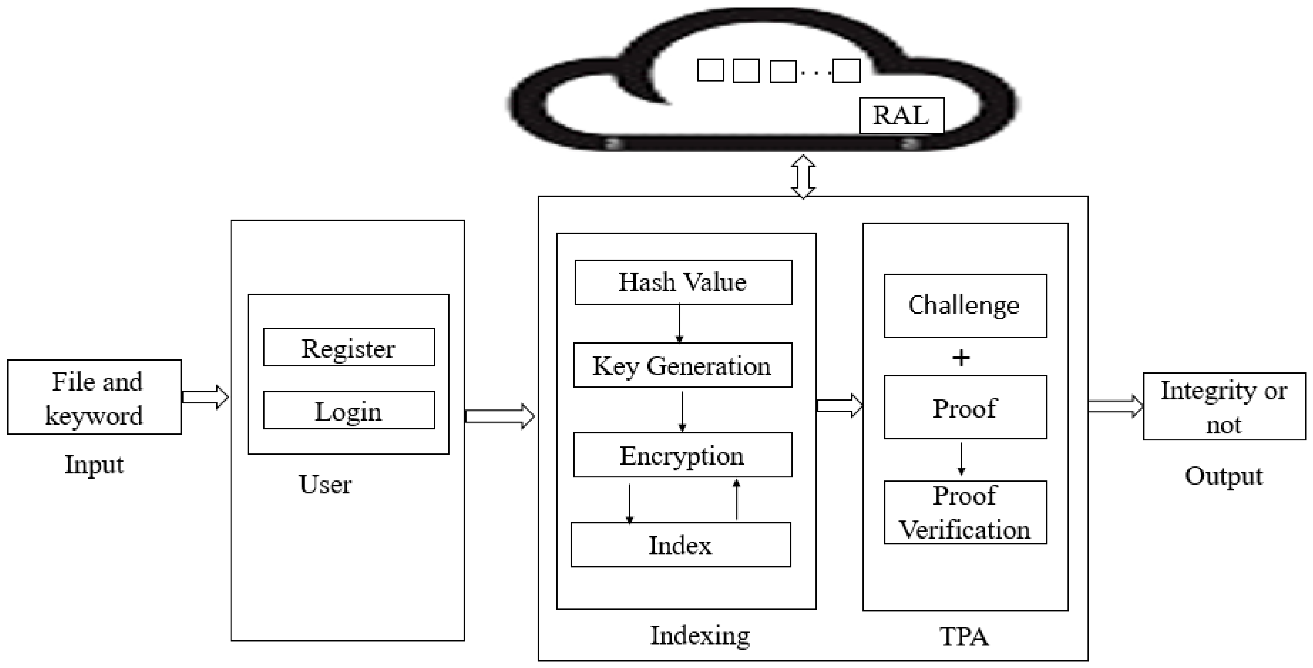


Figure 4.4 System Architecture

The TPA then verifies the authenticity of the RALs and generates an auditing proof without knowing which files contain the keyword or how many files contain it. The auditing proof is sent back to the user, who can use it to verify the integrity of the files containing the keyword without revealing any sensitive information to the TPA.

In terms of system architecture, the proposed scheme requires the implementation of several components, including a client-side application for generating RALs and uploading encrypted files to the cloud, a server-side application for storing RALs and encrypted files in the cloud, and a TPA application for verifying RALs and generating auditing proofs.

4.5 DATA FLOW DIAGRAMS (DFD)

A data flow diagram (DFD) is a graphical representation of the

“flow” of data through an information system, modeling its process aspects. A DFD is often used as a preliminary step to create an over review of the system, which can later be elaborated.

A DFD shows what kinds of data will be input to and output from the system, where the data will come from and go to, and where the data will be stored it does not show information about the timing of the process, or information about whether process will operating sequence or in parallel. In our project, we constructed data flow in three levels (level 0, level 1, level 2). A DFD is intended to serve as a communication tool among:

- System analysis
- End users
- Database designers
- System Programmers
- Other member of the project team.

4.5.1 DFD Level-0

DFD Level 0 is also known as a Context Diagram. It's a high-level overview of the entire system or process that's being studied or modelled. It's meant to be a quick glance into the system, displaying it as a single high-level process with its connections to external entities.

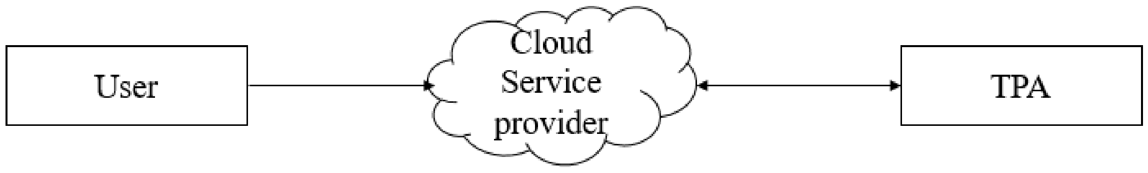


Figure 4.5.1 Data Flow Diagram Level-0

4.5.2 DFD Level-1

The context diagram is split into numerous bubbles/processes in 1-level DFD. The key functions of the system are highlighted at this level, and the high level process of 0-level DFD is broken down into sub processes.

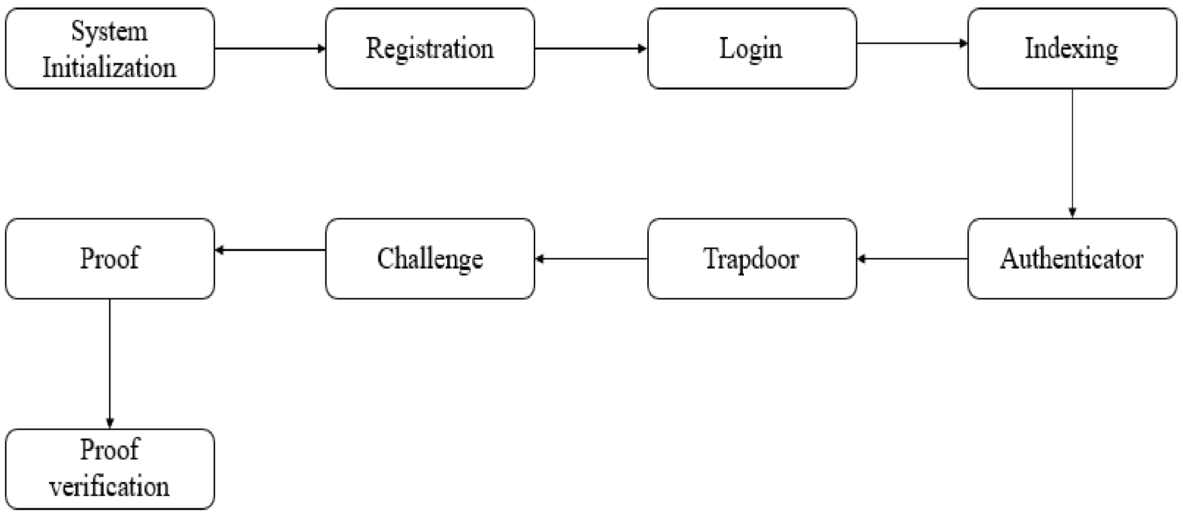


Figure 4.5.2 Data Flow Diagram Level-1

4.6 UML DIAGRAMS

UML is an acronym for Unified Modeling Language. In the field of object-oriented software engineering, UML is a standardised general-purpose modelling language that is designed to give a consistent way to visualise the architecture of a system. The goal is for

UML to become a standard language for developing object-oriented software models. Although UML is not a development approach in and of itself, it was created to work in tandem with the most widely used object-oriented software development method. The Unified Modelling Language is a programming language for specifying, visualising, constructing, and documenting software system artefacts, as well as business modelling and other non-software systems. The UML is a collection of best engineering practises for modelling large and complex systems that have been demonstrated to work. To express the design of software projects, the UML primarily employs pictorial notation.

UML defines nine types of Diagrams: Class, Use Case, Sequence, Collaboration, State Chart, Activity, Component and Deployment.

4.6.1 Use Case Diagram

A use case is a system analysis methodology for identifying, clarifying, and organising system needs. A use case is a collection of conceivable sequences of interactions between the system and users in a specific environment, all of which are tied to a specific purpose. The ellipse is used to depict it. Any external entity that interacts with the represented system is referred to as an actor.

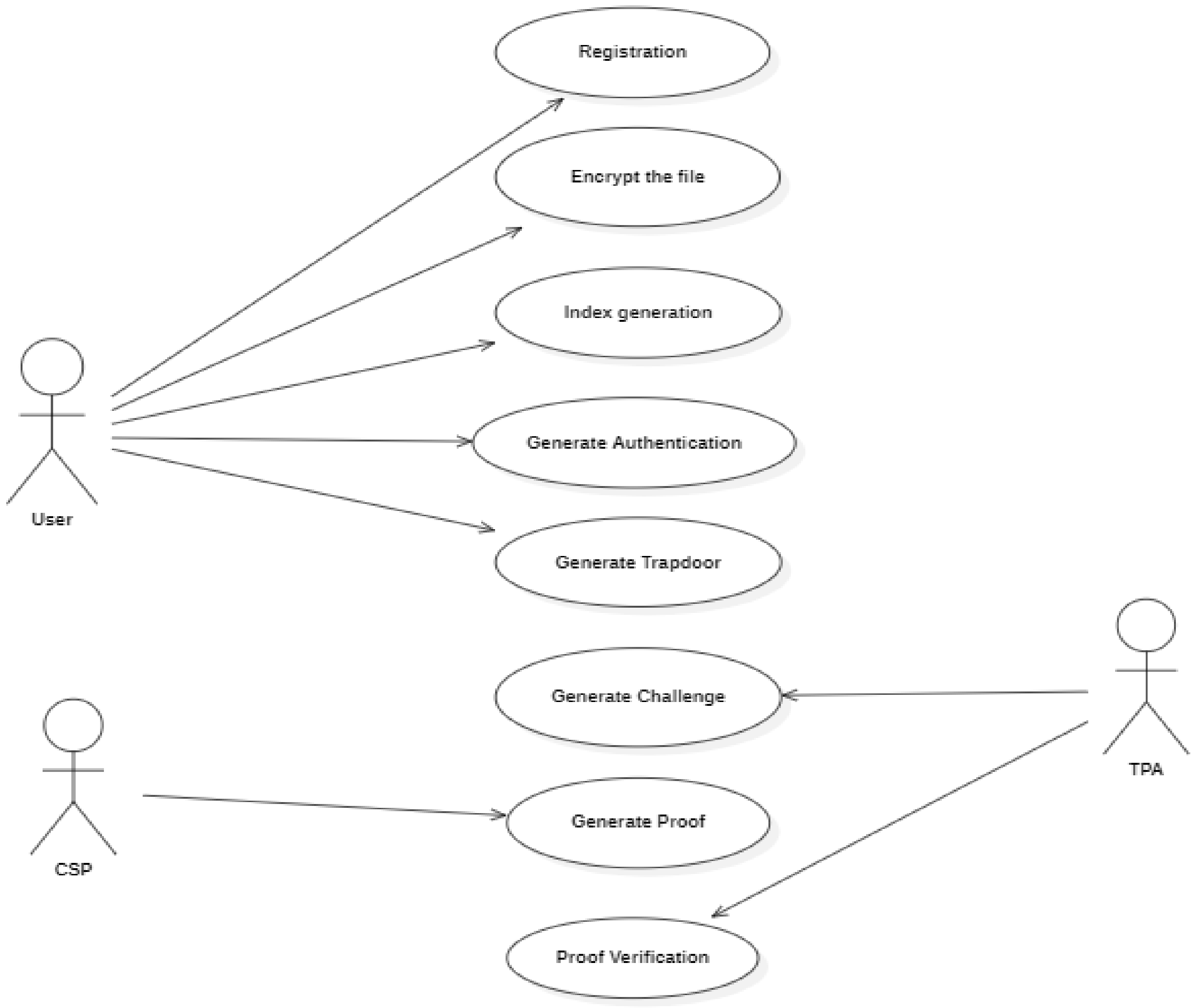


Figure 4.6.1 Use Case Diagram

4.6.2 Class Diagram

In the Unified Modelling Language, a class diagram depicts the relationships and source code dependencies between classes. A class defines the methods and variables in an object, which is a specific entity in a programme or the unit of code that represents that entity in this context. In all types of objectoriented programming, class diagrams are useful (OOP). The concept has been developed as OOP modelling paradigms have progressed over the years.

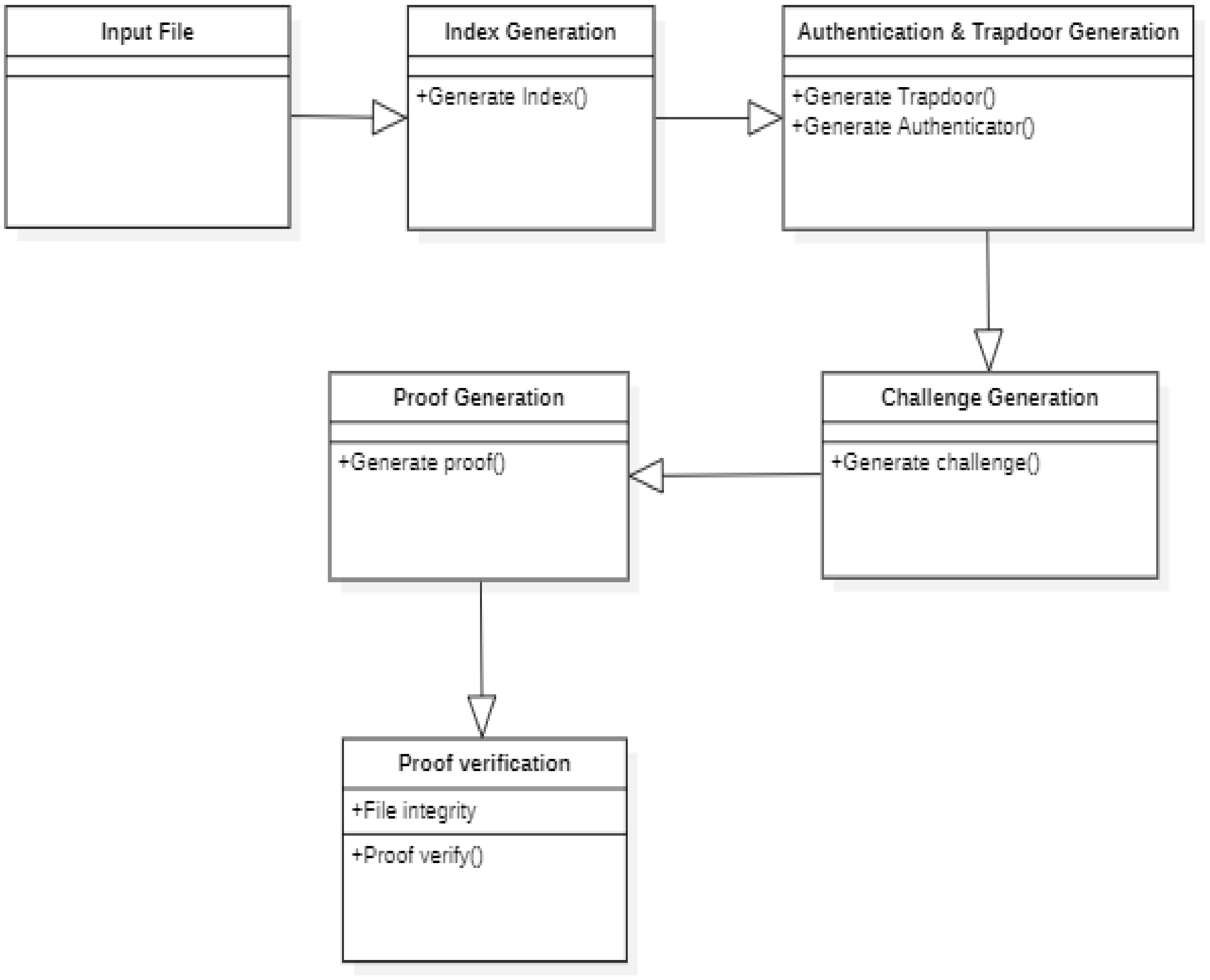


Figure 4.6.2 Class Diagram

4.6.3 Sequence Diagram

A sequence diagram depicts item interactions in chronological order. It illustrates the scenario's objects and classes, as well as the sequence of messages sent between them to carry out the scenario's functionality. In the Logical View of the system under development, sequence diagrams are often related with use case realisations. Event diagrams and event scenarios are other names for sequence diagrams. A sequence diagram depicts different processes or objects that exist at the same time as parallel vertical lines, and the messages that are passed between them as horizontal arrows.

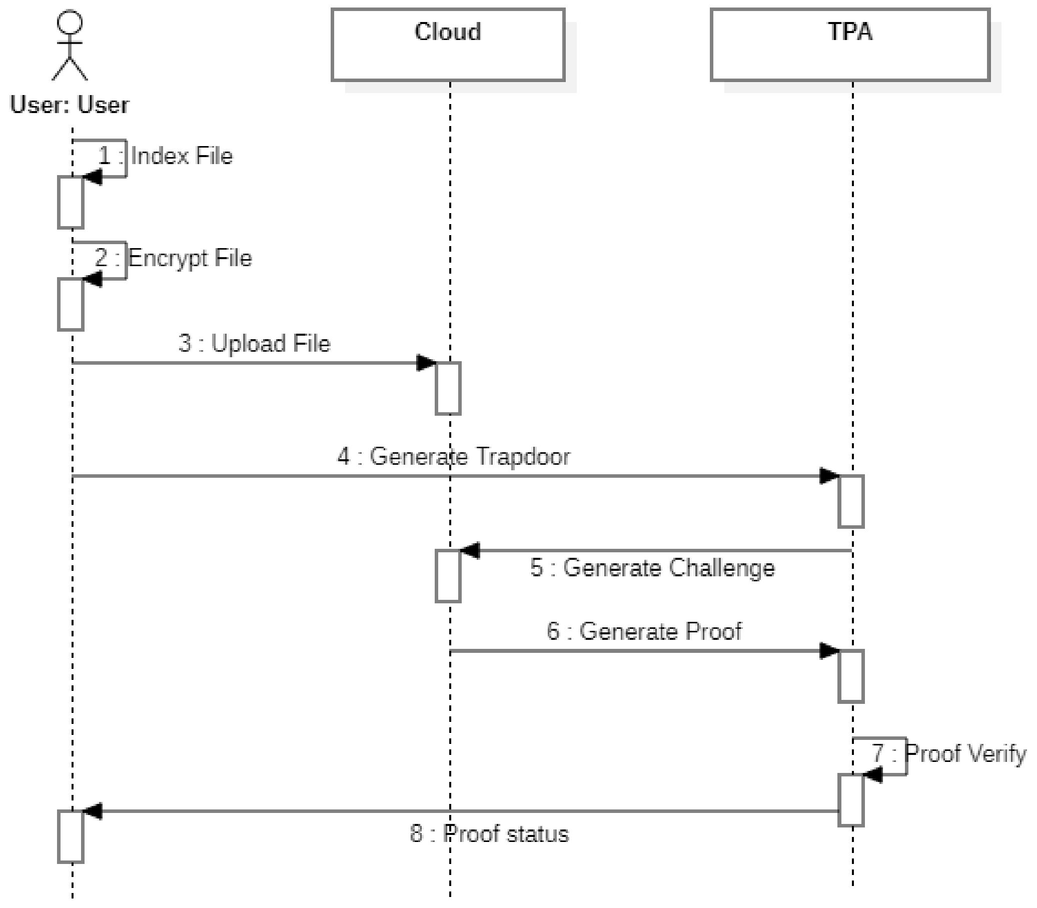


Figure 4.6.3 Sequence Diagram

4.6.4 State Chart Diagram

A state diagram is a sort of diagram used to illustrate the behavior of systems in computer science and related subjects. State diagrams imply that the system being depicted has a finite number of states; in some cases, this is true, while in others, it is a valid abstraction. State diagrams come in a variety of shapes and sizes, each with its own set of semantics.

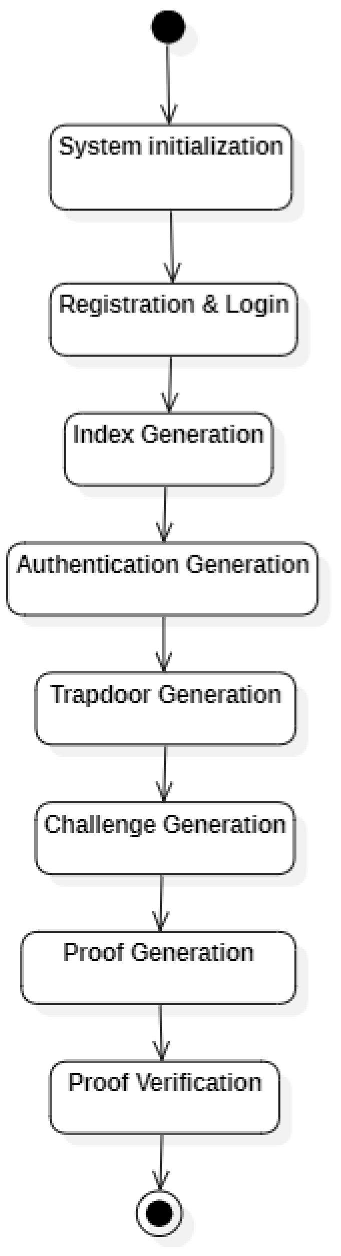


Figure 4.6.4 State Chart Diagram

4.6.5 Component Diagram

UML Component diagrams are used to depict the physical aspects of object-oriented systems. They are used for visualising, describing, and documenting component-based systems, as well as for forward and reverse engineering to create executable systems.

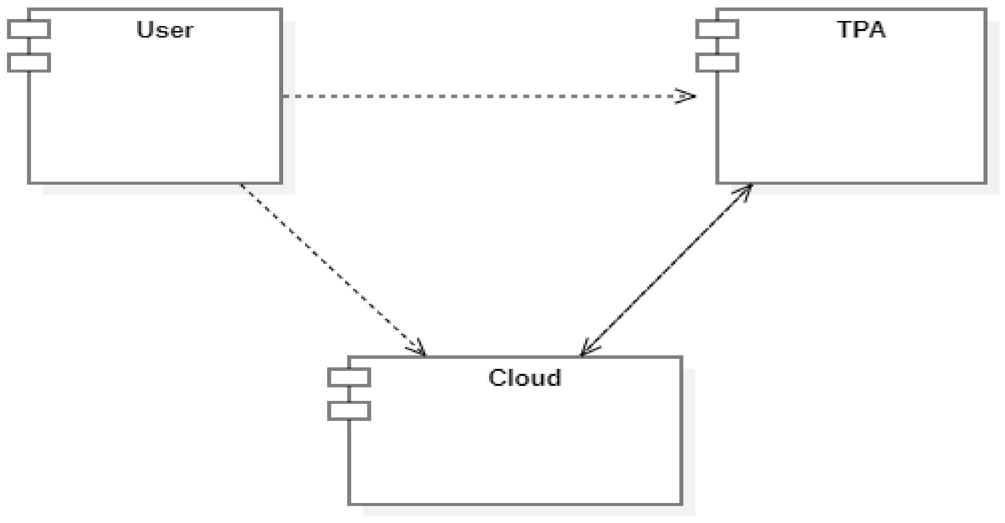


Figure 4.6.5 Component Diagram

4.6.6 Deployment Diagram

A deployment diagram is a sort of UML diagram that depicts a system's execution architecture, containing nodes like hardware or software execution environments, as well as the middleware that connects them. Deployment diagrams are commonly used to depict a system's physical hardware and software.

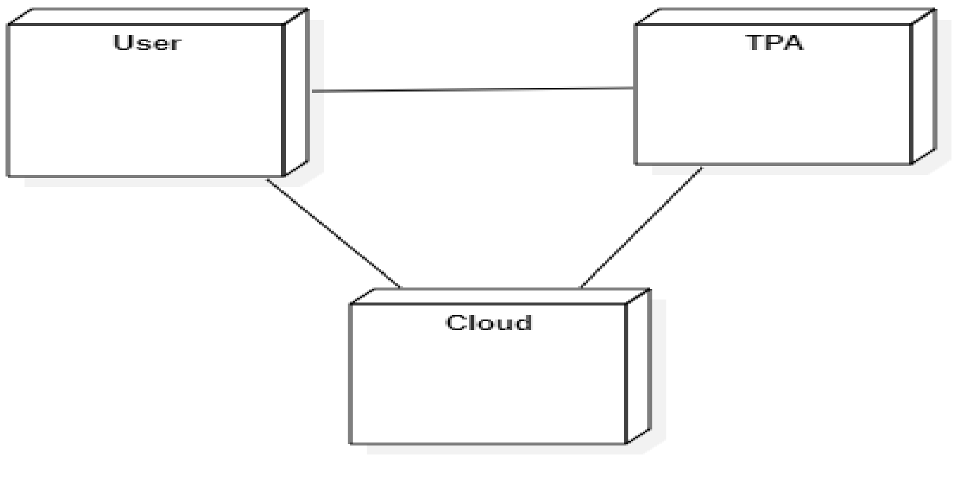


Figure 4.6.6 Deployment Diagram

CHAPTER 5

IMPLEMENTATION

5.1 MODULES

This project contains a total of 6 modules. They are listed as follows

1. Registration
2. Indexing
3. Cloud
4. TAP

5.2 MODULE EXPLANATION

The overall module consists of four key components. Registration involves creating user accounts by providing credentials. Indexing organizes and structures data for efficient searching and retrieval. Cloud computing delivers computing services over the internet, providing scalability and cost-efficiency. Enables the sharing of sensitive data while protecting individual privacy through techniques like encryption. Together, these components facilitate user access, data organization, and privacy protection in various computing environments to check integrity.

5.2.1 Registration

Input: Files and keyword

Output: Encrypted file block and keyword set

Algorithm:

1. The user register and creates account.
2. Then login to the account.
3. Randomly choose the secret key x and public key y.

4. The file is splits into s blocks and then the blocks are encrypted by symmetric encryption.

5. Build the keyword set w_k from the files.

5.2.2 Indexing

Input: Encrypted file block and keyword set

Output: Secure index, Authenticator, Trapdoor

Algorithm:

1. Generate three secure Hash functions $H1:\{0,1\}^* \rightarrow G_1$, $H_2\{0,1\}^* \rightarrow G_1$, $H_3\{0,1\}^* \rightarrow G_1$
2. Computes $\pi(w_k)$ as the address of each row in the secure index.
3. Encrypts the index vector. $ev_\pi(w_k) = vw_k \oplus f(\pi(w_k))$.
4. Computes the RAL $\Omega_{\pi(w_k)=}\{\Omega_{w_{k,1}}, \Omega_{w_{k,2}}, \dots, \Omega_{w_{k,s}}\}$.
5. Computes the Secure index $I = \{\pi(w_k), ev_{\pi(w_k)}, \Omega_{\pi(w_k)}\}k = 1,2, \dots., m$.
6. Computes the authenticator $\sigma_{ij} = [H_1(ID_i||j) . u^{eij}]^x$.
7. Computes the search trapdoor as $T_{w'} = \{\pi(w'), f(\pi(w'))\}$.

5.2.3 Cloud

Input: Challenge, Secure index, Encrypted data blocks and Authenticators

Output: Proof

Algorithm:

1. It stores Auditing Challenge, Secure index I, encrypted data blocks and Authenticators in cloud.

$Chal = \left\{T_{w',} \{j, v_j\}_{j\in Q}\right\} \quad T_{w'=\{\pi(w'), f(\pi(w'))\}} \quad v_{wk} = ev_{\pi\left(w_k\right)} \oplus f\left(\pi(w_k)\right)$

2. Using RAL, The cloud generate Auditing Proof and it sends to TPA.

$$T = \prod_{i\in S_{w_K}} \prod_{i\in Q} \sigma_{ij}^{vj} . \prod_{j\in Q} \Omega_{w_k,} j^{vj},\ \mu = \Sigma_{i\in S_{w_k}} \Sigma_{j\epsilon Q} C_{ij} .\ v_{j.}$$

5.2.4 TPA

Input: Trapdoor and proof

Output: Challenge and proof verification

Algorithm:

1. It takes Trapdoor as input.

2. And it generate the Auditing Challenge then TPA sends challenge to Cloud.

$$Chal = \left\{T_{w\prime,} \{j, v_j\}_{j\in Q}\right\}.$$

3. TPA receives the Auditing proof from the cloud.

4. TPA verifies proof using Auditing Challenge sent by TPA and Proof generated by the cloud.

$$e(T, g) \overset{?}{=} e\left(\left(\prod_{j\in Q} \left(H_3(j).\ \ H_2(\pi(w')||j)\right)^{v_j}\right).\ u^{\mu},\ y\right)$$

CHAPTER 6

SYSTEM TESTING

6.1 TESTING

Errors are discovered during testing. A good test case is one that has a high chance of uncovering an error that has yet to be detected. It ensures that the entire set of programmes runs smoothly. System testing necessitates a test containing numerous key actions and processes for every programme, string, and system, and is critical to the effective implementation of a new system. This is your last chance to catch something and fix it. It's utilised for quality control. Testing is an important aspect of the creation and maintenance of software.

The purpose of testing at this phase is to check that the specification has been accurately and thoroughly included into the design, as well as the design's correctness. For example, no logic flaws in the design must be recognised before coding begins; otherwise, the cost of correcting the flaws will be significantly higher, as reflected. Inspection and walkthrough are both effective methods for detecting design flaws.

The goal of testing is to find mistakes. Testing is the practise of attempting to find all possible flaws or weaknesses in a work product. It provides a means of testing the functionality of components and sub-assemblies in order to ensure that the software system satisfies its requirements and user expectations and does not fail in an unacceptable fashion. There are many different types of tests. Each test type is designed to fulfil a distinct testing need.

6.2 TYPES OF TESTING

6.2.1 Unit Testing

Unit testing entails creating test cases to ensure that the program's core logic is working properly and that programme input produces valid output. Validation should be performed on all decision branches and internal code flow. It is the testing of the application's individual software units. It's done after an individual unit's 38 completion and before it's integrated. This is an intrusive structural test that relies on prior knowledge of the structure. Unit tests are used to test a specific business process, application, or system configuration at the component level. Unit tests guarantee that each distinct path of a business process adheres to the stated standards and has clearly defined input and output.

6.2.2 Integration Testing

Integration testing is a method of building a program's structure while also conducting tests to find interfacing errors. Integration testing, in other words, is the complete testing of the project's set of modules. The goal is to develop a programme structure from of untested modules.

The tester should identify key modules. It's best to test critical components as soon as feasible. One method is to wait until all of the components have passed testing before combining and testing them. This method came up as a result of unstructured testing of small applications. The new module, as well as its interconnections.

6.2.3 Functional Testing

The code was put through its paces with nominal input values for which the expected outcomes were known, as well as boundary values such logically related inputs, files with identical elements, and empty files.

Functional tests demonstrate that the functions being tested are available in accordance with the business and technical requirements, system documentation, and user manuals. The following items are the focus of functional testing.

1. Valid Input: identified classes of valid input must be accepted.
2. Invalid Input: identified classes of invalid input must be rejected.
3. Functions: identified functions must be exercised.
4. Output: identified classes of application outputs must be exercised.
5. Systems/Procedures: interfacing systems or procedures must be invoked.

6.2.4 System Testing

System testing guarantees that the complete integrated software system complies with the specifications. It checks a setup to ensure that the results are known and predictable. The configuration-oriented system integration test is a form of system testing. System testing is based on process descriptions and flows, with an emphasis on process links and integration points that are pre-driven.

6.3 TESTING TECHNIQUES

6.3.1 Testing

Testing is the process of running a programme with the goal of detecting errors. A excellent test case is one that has a high chance of uncovering an error that has yet to be detected. A successful test is one that identifies an error that has yet to be identified.

System testing is a stage of implementation focused at ensuring that the system performs as planned before going live. It ensures that the entire set of programmes runs smoothly. System testing necessitates a test containing numerous key actions and processes for every programme, string, and system, and is critical to the effective implementation of a new system. Before 35 the system is installed for user acceptability testing, this is the last chance to find and rectify faults.

6.3.1.1 White Box Testing

This testing is also called as Glass box testing. In this testing, by knowing the specific functions that a product has been design to perform test can be conducted that demonstrate each function is fully operational at the same time searching for errors in each function. It is a test case design method that uses the control structure of the procedural design to derive test cases

6.3.1.2 Black Box Testing

By understanding a product's internal function, tests can be carried out to guarantee that "all gears mesh," or that the internal operation performs as expected and that all internal components have been thoroughly exercised. It is primarily concerned with the software's functional requirements.

6.4 TEST STRATEGY AND APPROACH

For the software developer, a software testing plan serves as a road map. Testing is a predetermined action that can be planned ahead of time and carried out in a systematic manner. As a result, a template for software testing that includes a series of steps into which we may insert individual test case creation methodologies should include the following features:

1. Testing begins at the module level and works “outward” toward the integration of the entire computer based system.
2. Different testing techniques are appropriate at different points in time.
3. The developer of the software and an independent test group conducts testing.
4. Testing and Debugging are different activities but debugging must be accumulated in any testing strategy.

6.4.1 Integration Testing

Integration testing is a method of creating a program's structure while also doing tests to find problems. Individual modules, which are prone to interface issues, should not be expected to work immediately after being combined. Of course, "putting them together" – interfacing – is the issue.

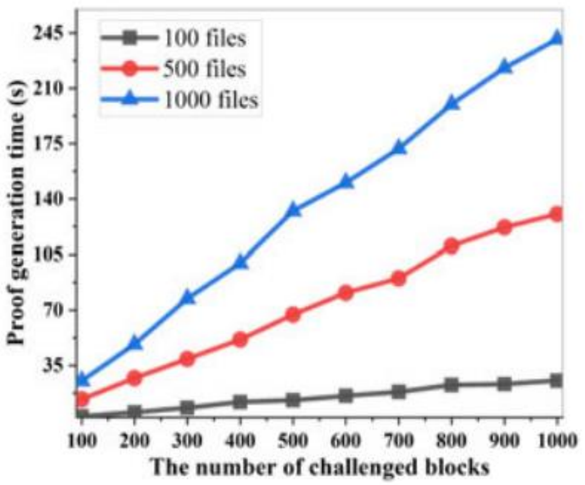
6.4.2 User Acceptance Testing

The system's user acceptance is a critical aspect in its success. At the time of development, the system under consideration is continually tested for user acceptance by remaining in touch with potential system and user.

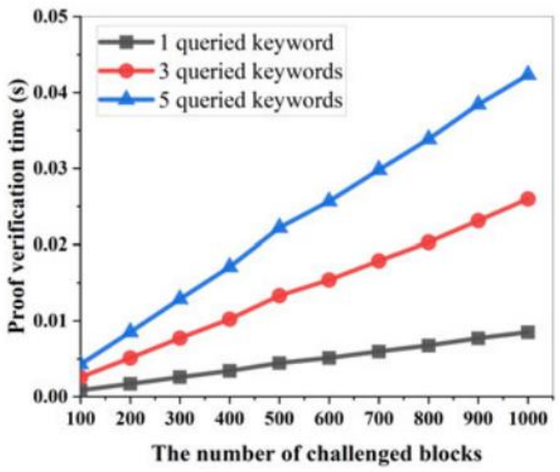
CHAPTER 7

PERFORMANCE ANALYSIS

The TPA needs to generate the auditing challenge and verify the auditing proof. We test the challenge generation time and the proof verification time for different numbers of challenged blocks, ranging from 100 to 1,000, and different numbers of queried keywords. As shown in Figs. 7.1 and 7.2, when challenging 100 blocks and one keyword, the TPA only needs 0.000054s to generate the auditing challenge and 0.0085362s to verify the auditing proof. When challenging 1,000 blocks and 5 keywords, the TPA needs 0.0026s to generate the auditing challenge and 0.04233s to verify the auditing proof. We can observe that the challenge generation time and the proof verification time are related to the number of challenged blocks and the number of queried keywords.



Figs.7.1 The challenge generation time



Figs.7.2 The proof verification time.

CHAPTER 8

CONCLUSION AND FUTURE WORK

8.1 CONCLUSION

Our project proposed a solution to address a new problem of how to achieve cloud data integrity auditing based on the keyword with sensitive information privacy. We design a new label called RAL, which is used to not only authenticate the relation that files contain the queried keyword but also generate the auditing proof without exposing any identity of file containing the queried keyword. We prove the security of the proposed scheme and evaluate the practical effectiveness by comprehensive experiments.

8.2 FUTURE WORK

Improving computation and storage efficiency is crucial. The current scheme requires computing a RAL for each keyword and data block, resulting in O(WN) computation and storage overhead. Finding ways to reduce this complexity independent of the number of keywords (W) and data blocks (N) is a challenge.

APPENDICES

APPENDIX 1: SAMPLE CODING

Main.java:

```
import it.unisa.dia.gas.jpbc.Element;

import it.unisa.dia.gas.jpbc.Pairing;

import it.unisa.dia.gas.jpbc.PairingParametersGenerator;

import it.unisa.dia.gas.plaf.jpbc.pairing.PairingFactory;

import it.unisa.dia.gas.plaf.jpbc.pairing.e.TypeECurveGenerator;

import javax.swing.*;

import java.awt.*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.math.BigInteger;

public class Main extends JFrame {

    JFrame f1, f2, f3;

    JPanel panel1, panel2;

    JLabel l1, l2, l3, l4;

    JButton b1, b2, b3, b4;

    public Main() {

        int rBits = 160;

        int qBits = 512;
```

```
PairingParametersGenerator pg = new TypeECurveGenerator(rBits, qBits);

Pairing pairing = PairingFactory.getPairing("params.properties");

PairingFactory.getInstance().setUsePBCWhenPossible(true);

System.out.println("\n\n 1.System Initialization");

Element u = pairing.getG1().newRandomElement().getImmutable();

Element g = pairing.getG1().newRandomElement().getImmutable();

Element x = pairing.getZr().newRandomElement().getImmutable();

Element y = g.powZn(x);

System.out.println("u is : " + u);

System.out.println("g is : " + g);

System.out.println("x is : " + x);

System.out.println("y is : " + y);

System.out.println("\n\n 2.Setup");

Element b1 = pairing.getZr().newRandomElement().getImmutable();

Element b2 = pairing.getZr().newRandomElement().getImmutable();

Element b3 = pairing.getZr().newRandomElement().getImmutable();

System.out.println("This our actual file blocks");

System.out.println("b1 is:" + b1);

System.out.println("b2 is:" + b2);

System.out.println("b3 is:" + b3);

Element c1 = pairing.getZr().newRandomElement().getImmutable();
Element c2 = pairing.getZr().newRandomElement().getImmutable();

Element c3 = pairing.getZr().newRandomElement().getImmutable();

System.out.println("\nThis our file Encrypted blocks");

System.out.println("c1 is:" + c1);

System.out.println("c2 is:" + c2);

System.out.println("c3 is:" + c3);

System.out.println("\nThis our keywords");

Element k0 = pairing.getZr().newRandomElement().getImmutable();

Element k1 = pairing.getZr().newRandomElement().getImmutable();

Element k2 = pairing.getZr().newRandomElement().getImmutable();

Element keyword[] = new Element[3];

keyword[0] = k0;

keyword[1] = k1;

keyword[2] = k2;

System.out.println("keyword[0] is:" + keyword[0]);

System.out.println("keyword[1] is:" + keyword[1]);

System.out.println("keyword[2] is:" + keyword[2]);

Element indexvector[] = new Element[3];

indexvector[0] = pairing.getZr().newElement(new BigInteger("1"));

indexvector[1] = pairing.getZr().newElement(new BigInteger("1"));

indexvector[2] = pairing.getZr().newElement(new BigInteger("1"));
```

```
        System.out.println("\n\n 3.Index Generation");

        System.out.println("Step 1:");

        Element pik0 = pairing.getZr().newRandomElement().getImmutable();

        Element pik1 = pairing.getZr().newRandomElement().getImmutable();

        Element pik2 = pairing.getZr().newRandomElement().getImmutable();

        System.out.println("pik0 is:" + pik0);

        System.out.println("pik1 is:" + pik1);

        System.out.println("pik2 is:" + pik2);

        System.out.println("Step 2:");

        Element fpik0 = pairing.getZr().newRandomElement().getImmutable();

        Element fpik1 = pairing.getZr().newRandomElement().getImmutable();

        Element fpik2 = pairing.getZr().newRandomElement().getImmutable();

        Element evpiwk0 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[0].toString(), fpik0.toString())));

        System.out.println("evpiwk0 is : " + evpiwk0);

        Element evpiwk1 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[1].toString(), fpik1.toString())));

        System.out.println("evpiwk1 is : " + evpiwk1);

        Element evpiwk2 = pairing.getZr().newElement(new
BigInteger(element_xor(indexvector[2].toString(), fpik2.toString())));

        System.out.println("evpiwk2 is : " + evpiwk2);

        System.out.println("Step 3:");

        Element swk0 = pairing.getZr().newElement(new BigInteger("1"));
        Element swk1 = pairing.getZr().newElement(new BigInteger("1"));

        Element swk2 = pairing.getZr().newElement(new BigInteger("1"));

        System.out.println("swk0 is :" + swk0);

        System.out.println("swk1 is :" + swk1);

        System.out.println("swk2 is :" + swk2);

        System.out.println("Step 4:");

        Element ohm_block1_v1 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block1_v2 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block1_v3 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_wk1 =
((ohm_block1_v1.add(ohm_block1_v2)).add(ohm_block1_v3)).powZn(x);

        Element ohm_block2_v1 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block2_v2 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block2_v3 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_wk2 =
((ohm_block2_v1.add(ohm_block2_v2)).add(ohm_block2_v3)).powZn(x);

        Element ohm_block3_v1 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_block3_v2 =
pairing.getG1().newRandomElement().getImmutable();
```

```
        Element ohm_block3_v3 =
pairing.getG1().newRandomElement().getImmutable();

        Element ohm_wk3 =
((ohm_block3_v1.add(ohm_block3_v2)).add(ohm_block3_v3)).powZn(x);

        System.out.println("ohm_wk1 is " + ohm_wk1);

        System.out.println("ohm_wk2 is " + ohm_wk2);

        System.out.println("ohm_wk3 is " + ohm_wk3);

        System.out.println("\n\n 4.Authenticator Generation");

        Element sigma_file1_block1 =
((ohm_block1_v1.invert()).add(u.powZn(c1))).powZn(x);

        Element sigma_file1_block2 =
((ohm_block2_v1.invert()).add(u.powZn(c2))).powZn(x);

        Element sigma_file1_block3 =
((ohm_block3_v1.invert()).add(u.powZn(c3))).powZn(x);

        System.out.println("sigma_file1_block1 is:" + sigma_file1_block1);

        System.out.println("sigma_file1_block2 is:" + sigma_file1_block2);

        System.out.println("sigma_file1_block3 is:" + sigma_file1_block3);

        System.out.println("\n\n 5.Trapdoor generation");

        System.out.println("Let us assume that, we want search the blocks containing the
keyword k0");

        System.out.println("pik0 is : " + pik0);

        System.out.println("fpik0 is : " + fpik0);

        System.out.println("\n\n 6.Challenge Generation");

        Element v1 = pairing.getZr().newRandomElement().getImmutable();
        Element v2 = pairing.getZr().newRandomElement().getImmutable();

        System.out.println("v1 is : " + v1);

        System.out.println("v2 is : " + v2);

        System.out.println("\n\n 7.Proof Generation");

        System.out.println("Step 1:");

        Element vw0 = pairing.getZr().newElement(new
BigInteger(element_xor(evpiwk0.toString(), fpik0.toString())));

        System.out.println("vw0 is : " + vw0);

        Element swk0_cs = pairing.getZr().newElement(new BigInteger("1"));

        Element T =
((sigma_file1_block1.powZn(v1)).add(sigma_file1_block2.powZn(v2))).add((ohm_wk
1.powZn(v1)).add(ohm_wk2.powZn(v2)));

        Element meu = (c1.mul(v1)).add(c2.mul(v2));

        System.out.println("Step 2 : ");

        System.out.println("T is : " + T);

        System.out.println("meu is : " + meu);

        System.out.println("\n\n 8.Proof Verification");

        Element LHS = pairing.pairing(T, g);

        System.out.println("LHS is :" + LHS);

        Element RHS =
pairing.pairing((((ohm_block1_v2.add(ohm_block1_v3)).powZn(v1)).add((ohm_block
2_v2.add(ohm_block2_v3)).powZn(v2))).add(u.powZn(meu)), y);

        System.out.println("RHS is :" + RHS);

        setSize(1300, 800);
        setLocationRelativeTo(null);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        JLabel w1 = new JLabel(" Enabling Integrity Auditing Based on the Keyword
With Sensitive Information Privacy for Encrypted Cloud Data");

        w1.setFont(new Font("Times New Roman", Font.BOLD, 24));

        w1.setPreferredSize(new Dimension(1200, 80));

        JLabel w2 = new JLabel("System Initialization");

        w2.setFont(new Font("Times New Roman", Font.BOLD, 17));

        w2.setPreferredSize(new Dimension(160, 80));

        JLabel w3 = new JLabel("Setup");

        w3.setFont(new Font("Times New Roman", Font.BOLD, 17));

        w3.setPreferredSize(new Dimension(160, 80));

        JLabel w4 = new JLabel(" ");

        w4.setFont(new Font("Times New Roman", Font.BOLD, 17));

        w4.setPreferredSize(new Dimension(160, 80));

        JTextArea m11 = new JTextArea();

        m11.setPreferredSize(new Dimension(500, 150));

        m11.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        JTextArea m12 = new JTextArea();

        m12.setPreferredSize(new Dimension(500, 150));

        m12.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

        JTextArea m13 = new JTextArea();
```

```
m13.setPreferredSize(new Dimension(500, 100));

m13.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m14 = new JTextArea();

m14.setPreferredSize(new Dimension(500, 100));

m14.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m15 = new JTextArea();

m15.setPreferredSize(new Dimension(1000, 100));

m15.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m16 = new JTextArea();

m16.setPreferredSize(new Dimension(500, 100));

m16.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m17 = new JTextArea();

m17.setPreferredSize(new Dimension(500, 100));

m17.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m18 = new JTextArea();

m18.setPreferredSize(new Dimension(1000, 250));

m18.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m19 = new JTextArea();

m19.setPreferredSize(new Dimension(800, 200));

m19.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m21 = new JTextArea();
```

```
m21.setPreferredSize(new Dimension(800, 150));

m21.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m22 = new JTextArea();

m22.setPreferredSize(new Dimension(800, 100));

m22.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m23 = new JTextArea();

m23.setPreferredSize(new Dimension(800, 150));

m23.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m24 = new JTextArea();

m24.setPreferredSize(new Dimension(500, 150));

m24.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JTextArea m25 = new JTextArea();

m25.setPreferredSize(new Dimension(500, 150));

m25.setBorder(BorderFactory.createLineBorder(Color.BLACK, 1));

JButton bt1 = new JButton("System Initialization");

bt1.setPreferredSize(new Dimension(250, 40));

bt1.setBackground(new Color(222,184,135));

JButton bt2 = new JButton("Setup");

bt2.setPreferredSize(new Dimension(250, 40));

bt2.setBackground(new Color(222,184,135));

JButton bt3 = new JButton("Index generation");
```

```
bt3.setPreferredSize(new Dimension(250, 40));

bt3.setBackground(new Color(222,184,135));

JButton bt4 = new JButton("Authenticator generation");

bt4.setPreferredSize(new Dimension(200, 40));

bt4.setBackground(new Color(222,184,135));

JButton bt5 = new JButton("Trapdoor");

bt5.setPreferredSize(new Dimension(150, 40));

bt5.setBackground(new Color(222,184,135));

JButton bt6 = new JButton("Challenge generation");

bt6.setPreferredSize(new Dimension(200, 40));

bt6.setBackground(new Color(222,184,135));


JButton bt7 = new JButton("Proof generation");

bt7.setPreferredSize(new Dimension(150, 40));

bt7.setBackground(new Color(222,184,135));

JButton bt8 = new JButton("Proof Verification");

bt8.setPreferredSize(new Dimension(150, 40));

bt8.setBackground(new Color(222,184,135));

JPanel panel1 = new JPanel();

panel1.setBackground(new Color(250,235,215));

panel1.add(w1);
```

```
        panel1.add(w2);

        panel1.add(m11);

        panel1.add(m12);

        panel1.add(w3);

        panel1.add(m13);

        panel1.add(m14);

        panel1.add(w4);

        panel1.add(m15);

        panel1.add(bt1);

        panel1.add(bt2);

        panel1.add(bt3);

        add(panel1);

        panel1.setVisible(true);

        bt1.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m11.setText("U is :" + u.toString() + "\n G is:" + g.toString());

                m11.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE,
13));

                m11.setLineWrap(true);

                m12.setText("x is :" + x.toString() + "\n\n Y is:" + y.toString());
                m12.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE,
13));

                m12.setLineWrap(true);



            }

        });



        bt2.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m13.setText("b1 is :" + b1.toString() + "\n\nb2 is:" + b2.toString() + "\n\nb3
is:" + b3);

                m13.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE,
13));

                m13.setLineWrap(true);

                m14.setText("c1 is :" + c1.toString() + "\n\nc2 is:" + c2.toString() + "\n\nc3
is: " + c3.toString());

                m14.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE,
13));

                m14.setLineWrap(true);

                m15.setText("Keyword 1 is :" + k0.toString() + "\n\nKeyword 2 is :" +
k1.toString() + "\n\nkeyword 3 is: " + k2.toString());

                m15.setFont(new Font("Times New Roman", Font.ROMAN_BASELINE,
13));

                m15.setLineWrap(true);
```

```
    }
});
bt3.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame frame2 = new JFrame("Indexing");
        frame2.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame2.setSize(1300, 800);
        JPanel panel2 = new JPanel();
        panel2.setBackground(new Color(245,245,220));
        JLabel w5 = new JLabel("Index Generation");
        w5.setFont(new Font("Times New Roman", Font.BOLD, 17));
        w5.setPreferredSize(new Dimension(250, 80));
        JLabel w6 = new JLabel(" ");
        w6.setFont(new Font("Times New Roman", Font.BOLD, 17));
        w6.setPreferredSize(new Dimension(250, 80));
        JButton b31 = new JButton("Indexing");
        b31.setPreferredSize(new Dimension(250, 40));
        b31.setBackground(new Color(222,184,135));
        b31.addActionListener(new ActionListener() {
            @Override
```

```
                    public void actionPerformed(ActionEvent e) {

                        m16.setText("\npik0 is :" + pik0.toString() + "\npik1 is :" +
pik1.toString() + "\npik2 is :" + pik2.toString());

                        m16.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                        m16.setLineWrap(true);

                        m17.setText("\nevpiwko is :" + evpiwk0.toString() + "\nevpiwk1 is :" +
evpiwk1.toString() + "\nevpiwk2 is :" + evpiwk2.toString());

                        m17.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                        m17.setLineWrap(true);

                        m18.setText("\nohm_wk1 is :" + ohm_wk1.toString() + "\n\nohm_wk2
is :" + ohm_wk2.toString() + "\n\nohm_wk3 is :" + ohm_wk3.toString());

                        m18.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                        m18.setLineWrap(true);

                    }

                });

                panel2.add(w5);

                panel2.add(m16);

                panel2.add(m17);

                panel2.add(w6);

                panel2.add(m18);

                panel2.add(b31);
```

```
        panel2.add(bt4);
        frame2.add(panel2);
        frame2.setVisible(true);
    }
});
bt4.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame frame3 = new JFrame("Authenticator");
        frame3.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame3.setSize(1300, 800);
        JPanel panel3 = new JPanel();
        panel3.setBackground(new Color(255,235,205));
        JLabel w5 = new JLabel("Authenticator");
        w5.setFont(new Font("Times New Roman", Font.BOLD, 17));
        w5.setPreferredSize(new Dimension(250, 80));
        JLabel w6 = new JLabel("Trapdoor");
        w6.setFont(new Font("Times New Roman", Font.BOLD, 17));
        w6.setPreferredSize(new Dimension(250, 80));
        JButton b41 = new JButton("Authenticator");
        b41.setPreferredSize(new Dimension(250, 40));
        b41.setBackground(new Color(222,184,135));

        b41.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m19.setText("sigma_file1_block1 is :" + sigma_file1_block1.toString()
+ "\n\nsigma_file1_block2 is :" + sigma_file1_block2.toString() +
"\n\nsigma_file1_block3 is :" + sigma_file1_block3.toString());

                m19.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                m19.setLineWrap(true);

            }

        });

        bt5.addActionListener(new ActionListener() {

            @Override

            public void actionPerformed(ActionEvent e) {

                m21.setText("Trapdoors are : \n wpik0 is :" + pik0.toString() +
"\n\nfpik0 is :" + fpik0.toString());

                m21.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                m21.setLineWrap(true);

            }

        });

        panel3.add(w5);

        panel3.add(m19);
        panel3.add(w6);
        panel3.add(m21);
        panel3.add(b41);
        panel3.add(bt5);
        panel3.add(bt6);
        frame3.add(panel3);
        frame3.setVisible(true);
    }
});
bt6.addActionListener(new ActionListener() {
    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame frame4 = new JFrame("Challenge ");
        frame4.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame4.setSize(1300, 800);
        JPanel panel4 = new JPanel();
        panel4.setBackground(new Color(245,222,179));
        JLabel w5 = new JLabel("Challenge Generation");
        w5.setFont(new Font("Times New Roman", Font.BOLD, 17));
        w5.setPreferredSize(new Dimension(250, 80));
        JLabel w6 = new JLabel("Proof Generation");
```

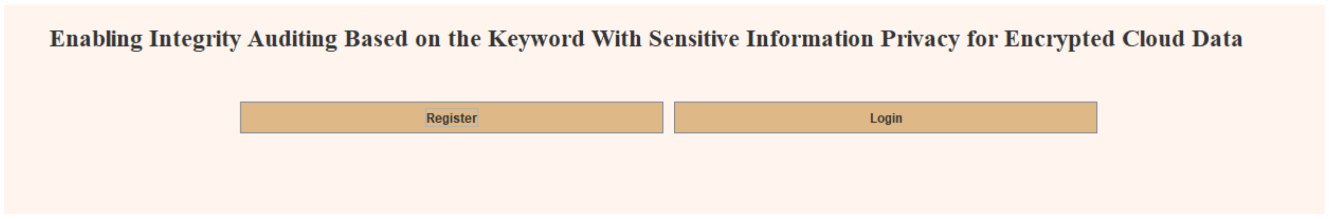
```
            w6.setFont(new Font("Times New Roman", Font.BOLD, 17));

            w6.setPreferredSize(new Dimension(250, 80));

            JButton b41 = new JButton("Challenge Generation");

            b41.setPreferredSize(new Dimension(250, 40));

            b41.setBackground(new Color(222,184,135));

            b41.addActionListener(new ActionListener() {

               @Override

               public void actionPerformed(ActionEvent e) {

                  m22.setText("Challenge generation is :" + v1.toString() + "\n\n" +
v2.toString());

                  m22.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                  m22.setLineWrap(true);

               }

            });

            JButton b42 = new JButton("Proof generation");

            b42.setPreferredSize(new Dimension(250, 40));

            b42.setBackground(new Color(222,184,135));

            b42.addActionListener(new ActionListener() {

               @Override

               public void actionPerformed(ActionEvent e) {

                  m23.setText("vw0 is :" + vw0.toString() + "\n\n T is : " + T.toString() +
"\n\n meu is : " + meu.toString());
```

```
                    m23.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                    m23.setLineWrap(true);

                }

            });

            panel4.add(w5);

            panel4.add(m22);

            panel4.add(w6);

            panel4.add(m23);

            panel4.add(b41);

            panel4.add(b42);

            panel4.add(bt8);

            frame4.add(panel4);

            frame4.setVisible(true);

        }

    });

    bt8.addActionListener(new ActionListener() {

        @Override

        public void actionPerformed(ActionEvent e) {

            JFrame frame5 = new JFrame("Proof Verification ");

            frame5.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

            frame5.setSize(1300, 800);
            JPanel panel5 = new JPanel();

            panel5.setBackground(new Color(255,248,220));

            JLabel w5 = new JLabel("Proof Verification");

            w5.setFont(new Font("Times New Roman", Font.BOLD, 17));

            w5.setPreferredSize(new Dimension(250, 80));

            JButton b41 = new JButton("LHS");

            b41.setPreferredSize(new Dimension(250, 40));

            b41.setBackground(new Color(222,184,135));

            b41.addActionListener(new ActionListener() {

               @Override

               public void actionPerformed(ActionEvent e) {

                  m24.setText("LHS is :" + LHS.toString());

                  m24.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                  m24.setLineWrap(true);

               }

            });

            JButton b42 = new JButton("RHS");

            b42.setPreferredSize(new Dimension(250, 40));

            b42.setBackground(new Color(222,184,135));

            b42.addActionListener(new ActionListener() {

               @Override
```

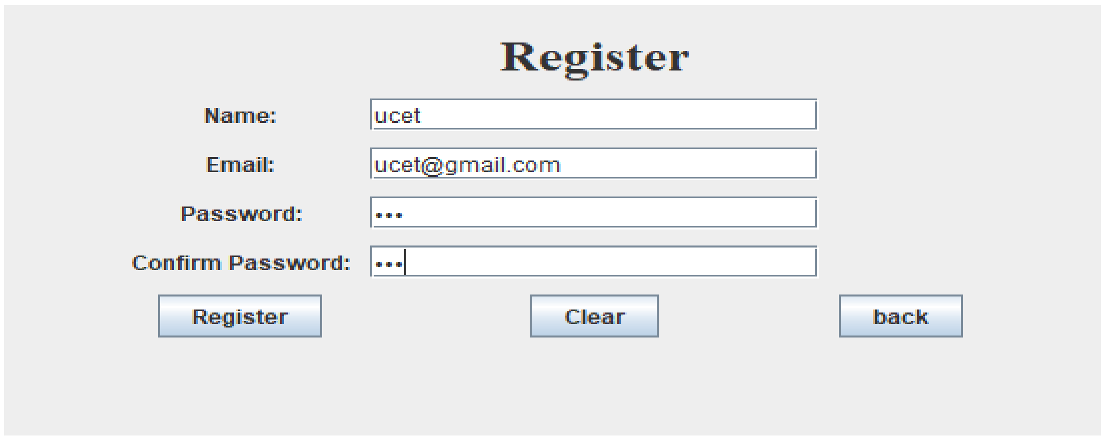
```
                    public void actionPerformed(ActionEvent e) {

                        m25.setText("RHS is :" + RHS.toString());

                        m25.setFont(new Font("Times New Roman",
Font.ROMAN_BASELINE, 15));

                        m25.setLineWrap(true);

                    }

                });

                panel5.add(w5);

                panel5.add(m24);

                panel5.add(m25);

                panel5.add(b41);

                panel5.add(b42);

                frame5.add(panel5);

                frame5.setVisible(true);

            }

        });

    }

    public static String element_xor(String v1, String v2) {

        System.out.println("v1 is : " + v1);

        System.out.println("v2 is : " + v2);

        StringBuffer v3 = new StringBuffer();

        for (int i = 0; i < v1.length(); i++) {
            v3.append(v1.charAt(i) ^ v2.charAt(i));
        }
        System.out.println("v3 is : " + v3);
        return v3.toString();
    }
    public static void main(String[] args) {
       new Main();
    }
}
```

APPENDIX 2: SCREENSHOTS

Home:


Enabling Integrity Auditing Based on the Keyword With Sensitive Information Privacy for Encrypted Cloud Data

Register


Register

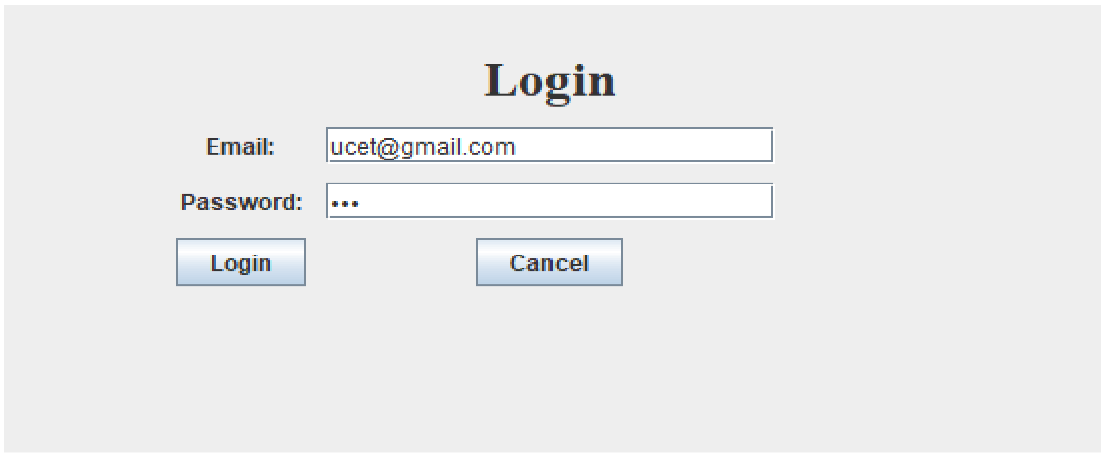
Name: ucet

Email: ucet@gmail.com

Password: •••

Confirm Password: •••

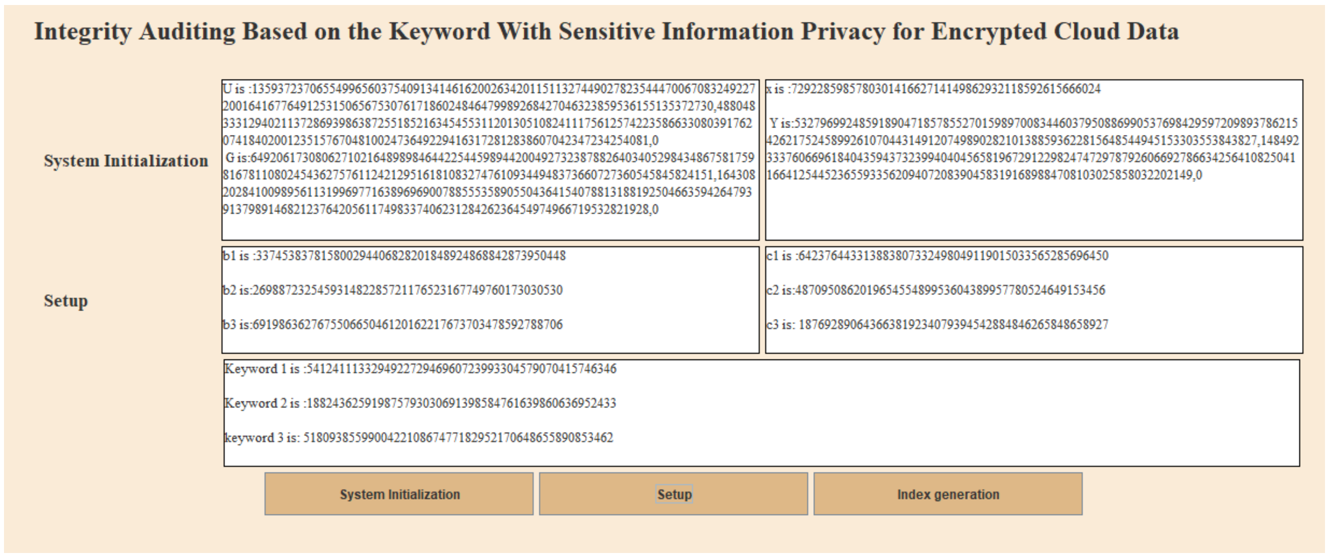
Login


Login

Email: ucet@gmail.com

Password: •••

System Initialization



Integrity Auditing Based on the Keyword With Sensitive Information Privacy for Encrypted Cloud Data

System Initialization

```
U is :135937237065549965603754091341461620026342011511327449027823544470067083249227
200164167764912531506567530761718602484647998926842704632385953615513537273 0,488048
333129402113728693986387255185216345455311201305108241117561257422358663308039 1762
074184020012351576704810024736492294163172812838607042347234254081,0
 G is:649206173080627102164898984644225445989442004927323878826403405298434867581759
816781108024543627576112421295161810832747610934494837366072736054584582415 1,164308
202841009895611319969771638969690078855535890550436415407881318819250466359426479 3
913798914682123764205611749833740623128426236454974966719532821928,0
```

```
x is :72922859857803014166271414986293211859261566 6024

 Y is:532796992485918904718578552701598970083446037950886990537698429597209893786215
426217524589926107044314912074989028210138859362281564854494515330355384382 7,148492
333760669618404359437323994040456581967291229824747297879260669278663425641082504 1
166412544523655933562094072083904583191689884708103025858032202149,0
```

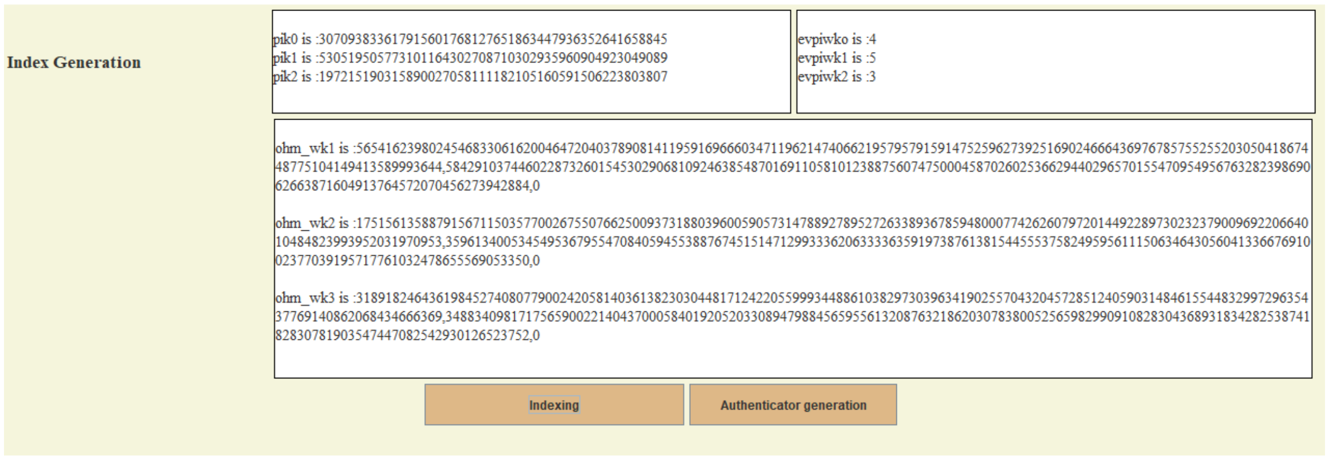
Setup

```
b1 is :33745383781580029440682820184892486884287395044 8

b2 is:2698872325459314822857211765231677497601730305 30

b3 is:69198636276755066504612016221767370347859278870 6
```

```
c1 is :64237644331388380733249804911901503356528569645 0

c2 is:4870950862019654554899536043899577805246491534 56

c3 is: 187692890643663819234079394542884846265848658927
```

```
Keyword 1 is :5412411133294922729469607239933045790704157463 46

Keyword 2 is :18824362591987579303069139858476163986063695243 3

keyword 3 is: 5180938559900422108674771829521706486558908534 62
```

Indexing



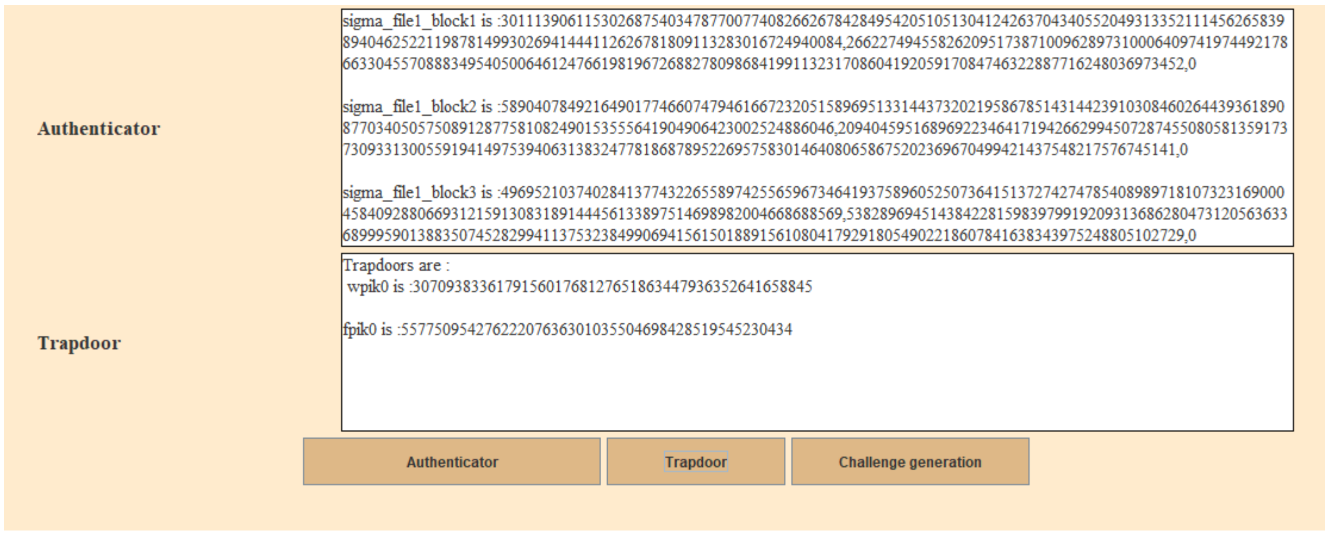
Index Generation

```
pik0 is :30709383361791560176812765186344793635264165884 5
pik1 is :53051950577310116430270871030293596090492304908 9
pik2 is :19721519031589002705811118210516059150622380380 7
```

```
evpiwko is :4
evpiwk1 is :5
evpiwk2 is :3
```

```
ohm_wk1 is :565416239802454683306162004647204037890814119591696660347119621474066219579579159147525962739251690246664369767857555255203050418674
48775104149413589993644,58429103744602287326015453029068109246385487016911058101238875607475000458702602536629440296570155470954956763282398690
6266387160491376457207045627394288 4,0

ohm_wk2 is :17515613588791567115035770026755076625009373188039600590573147889278952726338936785948000774262607972014492289730232379009692206640
10484823993952031970953,359613400534549536795547084059455388767451514712993336206333635919738761381544555375824959561115063464305604133667 6910
02377039195717761032478655569053350,0

ohm_wk3 is :318918246436198452740807790024205814036138230304481712422055999344886103829730396341902557043204572851240590314846155448329972963 54
37769140862068434666369,348834098171756590022140437000584019205203308947988456595561320876321862030783800525659829909108283043689318342825387 41
82830781903547447082542930126523752,0
```

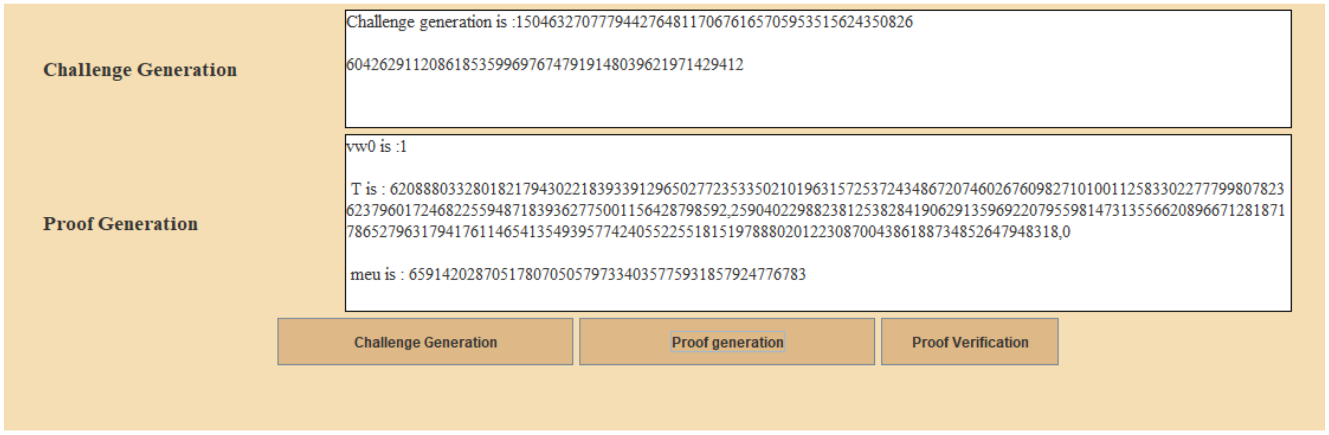
Authenticator and Trapdoor



```
sigma_file1_block1 is :3011139061153026875403478770077408266267842849542051051304124263704340552049313352111456265839
894046252211987814993026941444112626781809113283016724940084,26622749455826209517387100962897310006409741974492178
66330455708883495405006461247661981967268827809868419911323170860419205917084746322887716248036973452,0

sigma_file1_block2 is :5890407849216490177466074794616672320515896951331443732021958678514314423910308460264439361890
87703405057508912877581082490153555641904906423002524886046,209404595168969223464171942662994507287455080581359173
7309331300559194149753940631383247781868789522695758301464080658675202369670499421437548217576745141,0

sigma_file1_block3 is :4969521037402841377432265589742556596734641937589605250736415137274274785408989718107323169000
458409288066931215913083189144456133897514698982004668688569,53828969451438422815983979919209313686280473120563633
68999590138835074528299411375323849906941561501889156108041792918054902218607841638343975248805102729,0
```

```
Trapdoors are :
 wpik0 is :307093833617915601768127651863447936352641658845

fpik0 is :55775095427622207636301035504698428519545230434
```

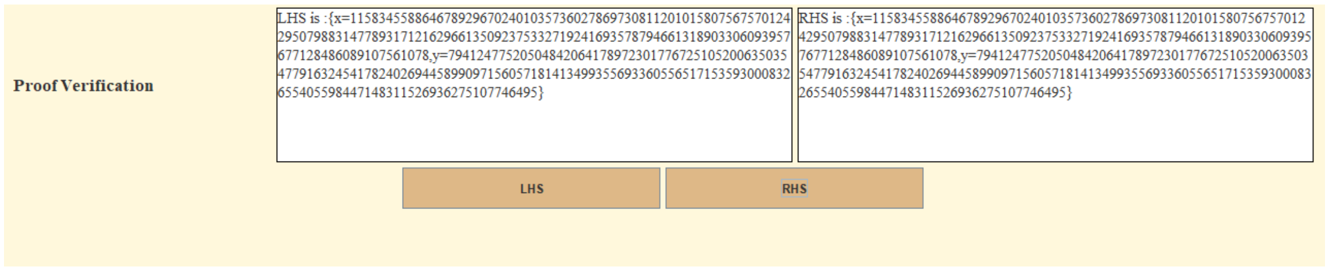
Challenge Generation and proof Generation



```
Challenge generation is :150463270777944276481170676165705953515624350826

604262911208618535996976747919148039621971429412
```

```
vw0 is :1

 T is : 620888033280182179430221839339129650277235335021019631572537243486720746026760982710100112583302277799807823
6237960172468225594871839362775001156428798592,2590402298823812538284190629135969220795598147313556620896671281871
78652796317941761146541354939577424055225518151978880201223087004386188734852647948318,0

 meu is : 659142028705178070505797334035775931857924776783
```

Proof Verification



```
LHS is :{x=1158345588646789296702401035736027869730811201015807567570124
29507988314778931712162966135092375332719241693578794661318903306093957
677128486089107561078,y=79412477520504842064178972301776725105200635035
47791632454178240269445899097156057181413499355693360556517153593000832
655405598447148311526936275107746495}
```

```
RHS is :{x=115834558864678929670240103573602786973081120101580756757012
42950798831477893171216296613509237533271924169357879466131890330609395
7677128486089107561078,y=7941247752050484206417897230177672510520063503
54779163245417824026944589909715605718141349935569336055651715359300083
2655405598447148311526936275107746495}
```

REFERENCES

1. G. Yang, J. Yu, W. Shen, Q. Su, Z. Fu, and R. Hao, ‘Enabling public auditing for shared data in cloud storage supporting identity privacy and traceability’, J. Syst. Softw., vol. 113, pp. 130–139, 2016.

2. R. Bost, P.-A. Fouque, and D. Pointcheval, ‘Verifiable dynamic symmetric searchable encryption: Optimality and forward security’, IACR, Lyon, France, Rep. 2016/062, 2016.

3. R. Curtmola, J. A. Garay, S. Kamara, and R. Ostrovsky, ‘Searchable symmetric encryption: Improved definitions and efficient constructions’, J. Comput. Secur., vol. 19, no. 5, pp. 895–934, 2011.

4. X. Ge, J. Yu, C. Hu, H. Zhang, and R. Hao, ‘Enabling efficient verifiable fuzzy keyword search over encrypted data in cloud computing’, IEEE Access, vol. 6, pp. 45725–45739, 2018.

5. X. Zhu, Q. Liu, and G. Wang, ‘A novel verifiable and dynamic fuzzy keyword search scheme over encrypted data in cloud computing’, in Proc. IEEE Trustcom/BigDataSE/ISPA, 2016, pp. 845–851.

6. Y. Yu, J. Ni, M. H. Au, Y. Mu, B. Wang, and H. Li, ‘Comments on a public auditing mechanism for shared cloud data service’, IEEE Trans. Serv. Comput., vol. 8, no. 6, pp. 998–999, Nov./Dec. 2015.