***A Major Project Report On***

**MULTI TENANCY CLOUD DATA WITH A SHARED PRIVACY PRESERVING TRUSTED KEYWORD SEARCH**

**Submitted to**

**Jawaharlal Nehru Technological University, Hyderabad**

*in partial fulfillment of the requirements for the award of the degree*

**BACHELOR OF TECHNOLOGY**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**by**

**SAURAV KUMAR MANDAL 21831A05M1**

**DEEKSHITA THUMMA 21831A05K9**

**HEERAKAR ABHISHEK 20831A0561**

***Under the Esteemed Guidance of***

**Mr. G. LAKPATHI**

**Assistant Professor, CSE**

**Department Of Computer Science & Engineering**

**Guru Nanak Institute of Technology**

**Ibrahimpatnam, Hyderabad, R.R District -501506**

**May, 2025**



**GURU NANAK INSTITUTE OF TECHNOLOGY**

**(Affiliated to JNTUH- Hyderabad)**

**Ranga Reddy District-501506**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**CERTIFICATE**

This is to certify that the project entitled **"** MULTI TENANCY CLOUD DATA WITH A SHARED PRIVACY PRESEVING TRUSTED KEYWORD SEARCH " is being presented with a report by SAURAV KUMAR MANDAL(21831A05M1), DEEKSHITA THUMMA(21831A05K9), HEERAKAR ABHISHEK (20831A0561) in partial fulfilment for the award of Degree of Bachelor of Technology in Computer Science and Engineering, to Jawaharlal Nehru Technological University, Hyde

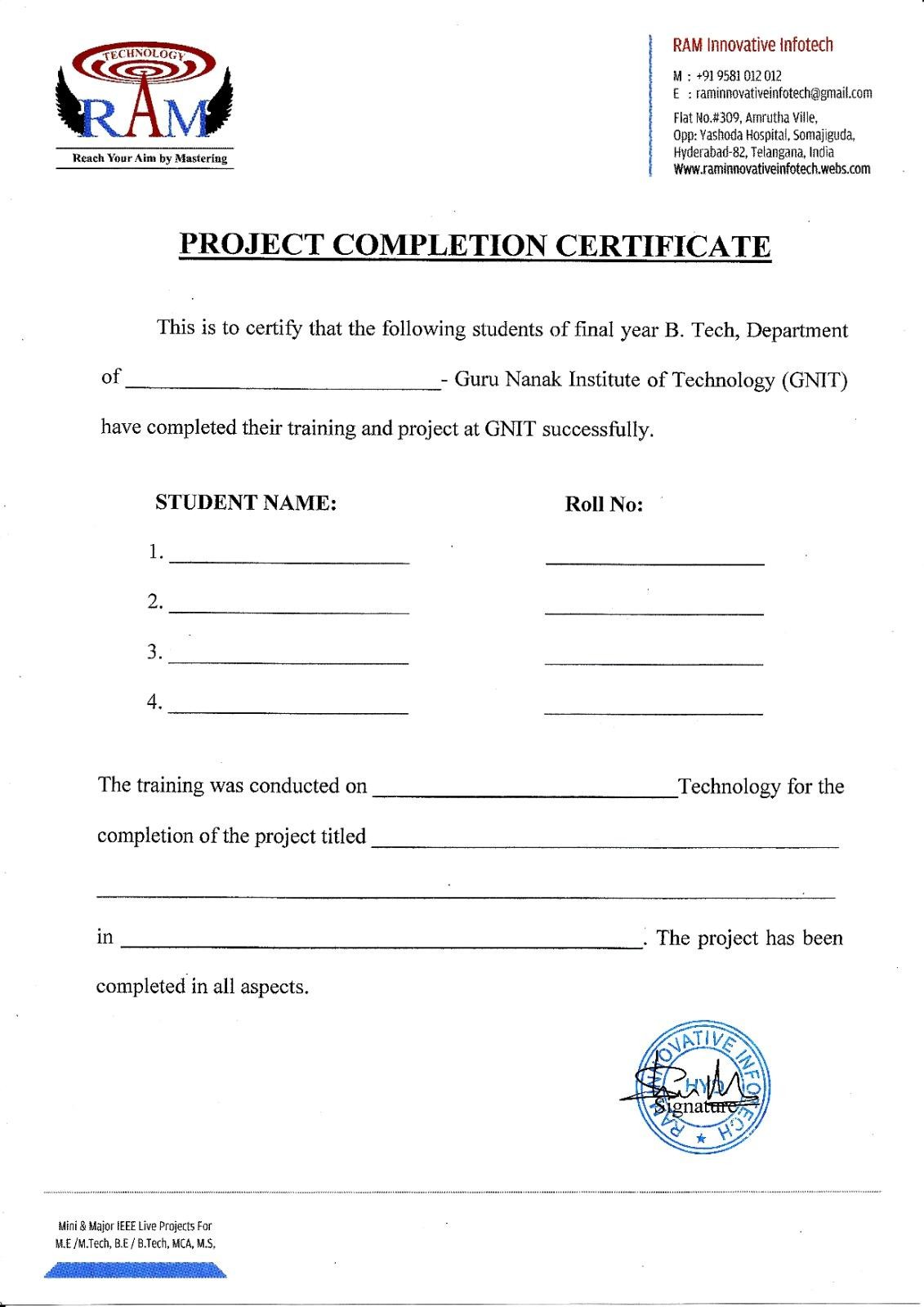
**Internal Guide Project Coordinator**

**Mr. G. Lakpathi Mr.D. Srinivas**

**Assistant professor Assistant Professor**

**Head of the Department External Examiner**

**Dr .B. Santhosh Kumar**



CSE

Saurav Kumar Mandal 21831A05M1

Deekshita Thumma 21831A05K9

Heerakar Abhishek 20831A0561

Java

Multi Tenancy cloud data with a shared Privacy

Preserving Trusted Keyword Search

2025

**GURU NANAK INSTITUTE OF TECHNOLOGY**

**Ibrahimpatnam, R.R. Dist. 501506**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**VISION**

* To be recognized as a leading department of Computer Science and Engineering in the region by students, employers and be known for leadership, Ethics, and commitment to fostering quality teaching-learning, research, and innovation.

**MISSION**

* Nurture young individuals into knowledgeable, skill-full and ethical professionals in their pursuit of Computer Science and Engineering.
* Nurture the faculty to expose them to world-class infrastructure.
* Sustain high performance by excellence in teaching, research and innovations.
* Extensive partnerships and collaborations with foreign universities for technology upgradation.
* Develop Industry-Interaction for innovation and product development.

**Declaration of Student**

We,SAURAV KUMAR MANDAL (21831A05M1), DEEKSHITA THUMMA (21831A05K9), HEERAKAR ABHISHEK (20831A0561),hereby declare that the major project titled **“MULTI** TENANCY CLOUD DATA WITH A SHARED PRIVACY PRESEVING TRUSTED KEYWORD SEARCH**”** has been carried out by us as part of the requirements for the award of the Degree of B. Tech in the Department of Computer Science And Engineering at Guru Nanak Institute of Technology.

We confirm the following:

The project was undertaken by us under the supervision of our guide, Mr. G. LAKPATHI , from the selection of the topic to the completion of the final report.

1. We have ensured that the results presented in the report are accurate and based on our original work.
2. To the best of our knowledge, the content of this report is free from plagiarism and adheres to ethical standards.
3. Each member of the team has contributed significantly and appropriately to the project work.
4. The project report has been prepared with diligence, ensuring clarity, accuracy, and adherence to academic standards.

We further declare that this report has not been submitted, in part or full, to any other institution or university for the award of any degree or diploma.

SAURAV KUMAR MANDAL Signature

21831A05M1

DEEKSHITA THUMMA Signature

21831A05K9

HEERAKAR ABHISHEK Signature

20831A0561

Date:

Place: GNIT, Ibrahimpatnam

**DECLARATION OF GUIDE**

I,Mr. G. LAKPATHI, hereby declare that I have guided the major project titled “MULTI TENANCY CLOUD DATA WITH A SHARED PRIVACY PRESEVING TRUSTED KEYWORD SEARCH” undertaken by SAURAV KUMAR MANDAL (21831A05M1), DEEKSHITA THUMMA (21831A05K9), HEERAKAR ABHISHEK (20831A0561). This project was carried out towards the fulfillment of the requirements for the award of the Degree of B. Tech in Computer Science and Engineering at Guru Nanak Institute of Technology.

As the guide, I confirm the following:

1. I have overseen the entire project process, from the selection of the project title to the submission of the final report.
2. I have reviewed and certified the accuracy and relevance of the results presented in the report.
3. The work carried out is original, free from plagiarism, and adheres to ethical guidelines.
4. The contributions of each student have been appropriately recognized and assessed.
5. The project report has been prepared under my supervision, ensuring adherence to high standards of quality, clarity, and structure.

I further certify that this project report has not been previously submitted in part or full for the award of any degree or diploma by any institution or university.

Name of Guide: Mr. G. LAKPATHI Signature of the Guide

Date:

Place:Gnit,Ibrahimpatnam

Name of HOD: Dr. B. SANTOSH KUMAR Signature of the HOD

Department: Computer Science Engineering

Date:

Place:Gnit,Ibrahimpatnam

**ACKNOWLEDGEMENT**

We would like to express our sincere gratitude to our internal guide, Mr. G. LAKPATHI, Assistant Professor, Department of Computer Science and Engineering, for his valuable guidance, encouragement, and continuous support throughout the duration of this project.

We are also thankful to Dr. B. SANTOSH KUMAR, Professor of Computer Science and Engineering, for his expert supervision and helpful suggestions, which contributed significantly to the successful completion of this project.

We would also like to thank the faculty members of the Department of Computer Science and Engineering and the Lab Technicians for their assistance and cooperation during the practical work of our project.

We are grateful to our friends and well-wishers for their encouragement, collaboration, and useful feedback throughout the project journey.

Lastly, we sincerely thank our parents for their constant support, patience, and motivation, which helped us complete this project successfully.

Saurav Kumar Mandal 21831A05M1

Deekshita Thumma 21831A05K9

Heerakar Abhishek 20831A0561

**TABLE OF CONTENT**

**DESCRIPTION PAGE NUMBER**

CERTIFICATE…………………………………………………………………….

DECLERATION OF STUDENT…………………………………………………... iv

DECLERATION OF GUIDE………………………………………………………. v

ACKNOWLEDGEMENT………………………………………………………….. vi

ABSTRACT………………………………………………………………………… x

LIST OF FIGURES…………………………………………………………………. xi

LIST OF NOTATIONS……………………………………………………………... xii-xiv

LIST OF ABBREVIATIONS……………………………………………………….. xv

**1.CHAPTER: INTRODUCTION………………………………………………………1-4**

1.1 General……………………………………………………………………………. 1

1.2 Scope of the project……………………………………………………………….. 2

1.3 Objective ………………………………………………………………………….. 2-3

1.4 Problem Statement………………………………………………………………… 3

1.5 Existing System……………………………………………………………………. 3

1.5.1 Existing System Disadvantages……………………………………………… 3

1.6 Organization of the Thesis…………………………………………………………. 4

**2.CHAPTER: LITERATURE SURVEY…………………………………………… 5-9**

2.1 Literature Survey……………………………………………………………………5-9

**3.CHAPTER: DESIGN AND DEVELOPMENT…………………………………... 10-61**

3.1 Problem statement / Need Analysis………………………………………………..10

3.1.1 Proposed System……………………………………………………………..10

3.1.2 Proposed System Advantages………………………………………………..10

3.1.3 System Requirements during Preliminary Analysis…………………………11

3.1.3.1 Hardware Requirements……………………………………………...11

3.1.3.2 Software Requirements……………………………………………….11

3.1.3.3 Functional Requirements……………………………………………..12

3.1.3.4 Non-Functional Requirements……………………………………….. 12-13

3.2 System Architecture…………………………………………………………………13-14

3.2.1 Justifying Design………………………………………………………………14-15

3.3 Design Methodology…………………………………………………………………15

3.3.1 General…………………………………………………………………………15

3.3.2 Methodologies………………………………………………………………….15

3.2.2.1 Modules Name………………………………………………………….15

3.2.2.2 Modules Explanation and Diagram…………………………………….16-22

3.3.3 Technique used or Algorithm used……………………………………………..23

3.4 Component Design / Subsystem Design……………………………………………...24

3.4.1 General………………………………………………………………………….24

3.4.2 Use Case Diagram………………………………………………………………24

3.4.3 Class Diagram…………………………………………………………………..25

3.4.4 Object Diagram…………………………………………………………………26

3.4.5 State Diagram…………………………………………………………………...27

3.4.6 Sequence Diagram………………………………………………………………28

3.4.7 Collaboration Diagram………………………………………………………….29

3.4.8 Activity Diagram………………………………………………………………..30

3.4.9 Component Diagram…………………………………………………………….31

3.4.10 Data Flow Diagram …………………………………………………………....32-33

3.4.11 E-R Diagram……………………………………………………………………33-34

3.4.12 Deployment Diagram…………………………………………………………...34-35

3.5 Tools and Technologies………………………………………………………………..35

3.5.1 Features of Java…………………………………………………………………. 35

3.5.1.2 The Java Framework……………………………………………………..35

3.5.1.3 Objectives of Java………………………………………………………..36-37

3.5.1.4 Java Swing Overview……………………………………………………37-38

3.5.1.5 Evolution of Collection………………………………………………….38-41

3.5.2 MYSQL Conformance…………………………………………………………..41-42

3.5.3 Apache Tomcat…………………………………………………………………..42

3.6 Implementation and Development……………………………………………………43-53

3.7 Testing and Validation………………………………………………………………..54

3.7.1 Development Methodologies…………………………………………………..54

3.7.2 Types of Testing……………………………………………………………….54

3.7.2.1 Unit Testing…………………………………………………………….54

3.7.2.2 Functional Testing……………………………………………………..54-55

3.7.2.3 System Testing…………………………………………………………55

3.7.2.4 Performance Testing……………………………………………………55

3.7.2.5 Integration Testing……………………………………………………...55

3.7.2.6 Acceptance Testing…………………………………………………….55-56

3.7.2.7 Build the Test Plan……………………………………………………..56

3.8 Photographs / Snapshots……………………………………………………………….56

3.8.1 Home Page of an Online Banking System……………………………………….56

3.8.2 Admin Dashboard Login Interface……………………………………………….57

3.8.3 Admin Home Page………………………………………………………………..57

3.8.4 Bank Registration Page…………………………………………………………...57

3.8.5 Bank Details Page………………………………………………………………...58

3.8.6 Bank Login Page………………………………………………………………….58

3.8.7 Bank Home Page………………………………………………………………….58

3.8.8 Customer Management Page……………………………………………………...59-61

**4.CHAPTER: RESULT AND DISCUSSION…………………………………………….62-64**

4.1 Results and Discussions…………………………………………………………………62-64

4.2 Scope for future work…………………………………………………………………....64

**5.CHAPTER: CONCLUSION…………………………………………………………….65**

5.1 General…………………………………………………………………………………..65

**ABSTRACT**

cloud service models intrinsically cater to multiple tenants. In current multi-tenancy model, cloud service providers isolate data within a single tenant boundary with no or minimum cross-tenant interaction. With the booming of cloud applications, allowing a user to search across tenants is crucial to utilize stored data more effectively. However, conducting such a search operation is inherently risky, primarily due to privacy concerns. Moreover, existing schemes typically focus on a single tenant and are not well suited to extend support to a multi-tenancy cloud, where each tenant operates independently. In this article, to address the above issue, we provide a privacy preserving, verifiable, accountable, and parallelizable solution for “privacy-preserving keyword search problem” among multiple independent data owners. We consider a scenario in which each tenant is a data owner and a user’s goal is to efficiently search for granted documents that contain the target keyword among all the data owners. We first propose a verifiable yet accountable keyword searchable encryption (VAKSE) scheme through symmetric bilinear mapping. For verifiability, a message authentication code (MAC) is computed for each associated piece of data. To maintain a consistent size of MAC, the computed MACs undergo an exclusive OR operation. For accountability, we propose a keyword-based accountable token mechanism where the client’s identity is seamlessly embedded without compromising privacy. Furthermore, we introduce the parallel VAKSE scheme, in which the inverted index is partitioned into small segments and all of them can be processed synchronously. We also conduct formal security analysis and comprehensive experiments to demonstrate the data privacy preservation and efficiency of the proposed schemes, respectively.

**LIST OF FIGURES**

|  |  |  |
| --- | --- | --- |
| **FIGURE NO** | **NAME OF THE FIGURE** | **PAGE NO.** |
| 2.3.2 | Module Diagram |  |
| 4.2.1 | Use case Diagram |  |
| 4.2.2 | Class Diagram |  |
| 4.2.3 | Object Diagram |  |
| 4.2.4 | Component Diagram |  |
| 4.2.5 | Deployment Diagram |  |
| 4.2.6 | Sequence Diagram |  |
| 4.2.7 | Collaboration Diagram |  |
| 4.2.8 | State Diagram |  |
| 4.2.9 | Activity Diagram |  |
| 4.3 | Data Flow Diagram |  |
| 4.4 | E-R Diagram |  |
| 4.5 | System Architecture |  |
| 7.1 | Home Page |  |

**LIST OF SYSMBOLS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **S.NO** | **NAME** | **NOTATION** | | **DESCRIPTION** | |
| 1. | Class | *Class Name*  *-attribute*  *-attribute*  *+operation*  *+operation*  *+operation*  *+ public*  *-private*  *# protected* | | Represents a collection of similar entities grouped together. | |
| 2. | Association | name  Class B  Class A  Class A  Class B | | Associations represent static relationships between classes. Roles represents the way the two classes see each other. | |
| 3. | Actor | Class A  Class A  Class B  Class B | | It aggregates several classes into a single class. | |
| 5. | Aggregation | Interaction between the system and external environment | |
| 5. | Relation  (uses) | | Uses | | Used for additional process communication. | |
| 6. | Relation  (extends) | | extends | | Extends relationship is used when one use case is similar to another use case but does a bit more. | |
| 7. | Communication | |  | | Communication between various use cases. | |
| 8. | State | | State | | State of the process. | |
| 9. | Initial State | |  | | Initial state of the object | |
| 10. | Final state | |  | | Final state of the object | |
| 11. | Control flow | |  | | Represents various control flow between the states. | |
| 12. | Decision box | |  | | Represents decision making process from a constraint | |
| 13. | Usecase | |  | | Interact ion between the system and external environment. | |

|  |  |  |  |
| --- | --- | --- | --- |
| 14. | Component |  | Represents physical modules which is a collection of components. |
| 15. | Node |  | Represents physical modules which are a collection of components. |
| 16. | Data Process/State |  | A circle in DFD represents a state or process which has been triggered due to some event or action. |
| 17. | External entity |  | Represents external entities such as keyboard,sensors,etc. |
| 18. | Transition |  | Represents communication that occurs between processes. |
| 19. | Object Lifeline |  | Represents the vertical dimensions that the object communications. |
| 20. | Message | Message | Represents the message exchanged. |

**LIST OF ABBREVATION**

|  |  |  |
| --- | --- | --- |
| **S.NO** | **ABBREVATION** | **EXPANSION** |
| 1**.** | DB | Database |
| 2. | JVM | Java Virtual Machine |
| 3. | JSP | Java Server Page |
| 4. | PWS | Personalized Web Search |
| 5. | UPS | User Personalized Search |
| 6. | JRE | Java Runtime Environment |

**CHAPTER 1**

**INTRODUCTION**

* 1. **GENERAL**

CLOUD computing has had a profound impact on data management. It offers massive storage and computing resources, payment-on-demand, and flexible scalability. Motivated by these advantages, thousands of clients are opting for cloud services. One typical application area is healthcare, and some applications are Healthvana [1] and CDPHP [2]; both the platforms are the tenants of Amazon [3]. Healthvana stores patient reports and CDPHP stores doctor information. It is desirable for a patient to search both the datasets to find the most suitable doctor by matching the patient data with the doctor information. For example, HIV patients store their reports in Healthvana and seek for suitable doctors from CDPHP. However, such a search across tenancies is challenging. Each tenant is an independent data owner and must abide the privacy laws, such as HIPAA [4], which are enforced to protect individuals’ medical data privacy. In addition, for their own interests, companies treat patient data as an asset and tend to maintain complete control over it. Data encryption is the best practice for maintaining data privacy. Each data owner encrypts their data before outsourcing it to the cloud. This guarantees the confidentiality of the data but greatly reduces their utility. A user must download an entire dataset in order to retrieve one piece of data. Considering data utility and privacy, Song et al. [5] introduced the primitives of symmetric searchable encryption (SSE). SSE is a keyword search technique that allows search over the cipher text without decryption. Goh et al. [6] proposed a secure index to improve search efficiency. Subsequently, Curtmola et al. [7] formalized the security definition of SSE and proposed two constructions that corresponded to non adaptive semantic security and adaptive semantic security.

In early research, most works on SSE focused on the honest but-curious cloud service provider (CSP). In such a model, the search result is fully trusted and the CSP is assumed to honestly follow the protocol specification. Search results in practice may contain corrupted data due to underlying hardware/software failures. In addition, for self-interest, the CSP may deviate from the protocol specification. For example, to reduce computational costs, CSP may randomly choose data as a search result. To mitigate this problem, Chai and Gong [8] proposed verifiable SSE, where the search result includes not only retrieved documents but also proof of the correctness and completeness of the search. The correctness of the search means that the returned search result matches the query. completeness of the search means that the retrieved data has not been tampered with. In addition, Chen et al. [9] proposed an authenticated Merkle hash tree to verify the search result. Although significant progress has been made by the existing constructions [8], [9], the verifiable property comes at the high cost of extra storage and computation. There is still room for solutions that are more practical. Recently, with increasing demand for users (e.g., physicians), previous SSE constructions, providing a client either full access to the data or no access, expose their short term. It is desirable to design a fine-grained access control mechanism to enable data owners to selectively grant grant clients access to their data. To achieve this goal, Han et al. [10] proposed to apply attribute-based encryption [11] to solve this problem and provided a general solution in the context of public-key keyword search scenarios. With this design, only a keyword search request that matches the predefined access structure can retrieve the target document. The above searchable encryption schemes rely on public key encryption, which is inefficient compared with symmetric encryption. Moreover, none of them are suitable for use with dynamic dynamic access structures since the access structure is associated with either a key or cipher text. Any change in the access structure may result in all of the ciphertext or keys being renewed. Furthermore, all the mentioned works failed to allow a client to search the data from multiple data owners, where each data owner encrypts their data with a unique key. The existing SSE schemes only support a client to search over a single data owner [5], [12], [13]. However, in our scenario, a client needs to search for data outsourced by multiple independent data owners. For example, to identify a medical treatment for a cancer patient, a physician may need to analyze medical data from thousands of contributors (e.g., patients). An intuitive solution for this scenario is to deploy existing schemes [5], [12], [13] for each data owner, where each data owner manages their outsourced data independently. For each service request, the physician generates a specialized (authorized) request for each owner’s data and sends it to the CSP. This results in a high volume of requests for a single query. Another approach is to adopt the recent Multi-Writer Encrypted Database [14], which allows multiple data owners to store data and allows clients to search across data owners.

**1.2 SCOPE OF THE PROJECT**

In this paper, Data encryption is the best practice for maintaining data privacy. Each data owner encrypts their data before outsourcing it to the cloud. This guarantees the confidentiality of the data but greatly reduces their utility. A user must download an entire dataset in order to retrieve one piece of data. the primitives of symmetric searchable encryption (SSE). SSE is a keyword search technique that allows search over the cipher text without decryption

**1.3 OBJECTIVE**

The objective of Herein, we propose a framework that is, to the best of our knowledge, the first to to tackle all the above challenges. In the proposed scheme, each data owner encrypts its own dataset (with its unique key) and outsources the storage and processing tasks for search operations to the CSP. For privacy, all the data are encrypted using a standard symmetric encryption algorithm and an index is established for privacy-preserving search. The proposed indexing mechanism provides not only privacy but also the ability to search over the datasets of several data owners. Each data owner encrypts its own dataset independently, and the search ability of cipher text is enabled across all the data owners through this indexing mechanism

**1.4 PROBLEM STATEMENT**

In today’s digital era, cloud computing has become a widely adopted solution for data storage and management due to its scalability, cost-efficiency, and flexibility. Among its architectures, multi-tenancy is particularly popular, as it allows multiple users or organizations (tenants) to share the same physical infrastructure while maintaining logical data separation. However, this shared nature of resources introduces significant security and privacy concerns. In a multi-tenant environment, sensitive data from different tenants is often stored on the same cloud platform, raising the risk of data leakage, unauthorized access, and cross-tenant privacy breaches.

To ensure data confidentiality, most cloud users encrypt their data before uploading it to the cloud. However, this practice creates another challenge: the inability to perform efficient keyword searches over encrypted data. Traditional keyword search techniques are either not secure or inefficient in encrypted environments, especially when multiple users are involved. Moreover, cloud service providers are often considered "honest but curious," meaning they may follow the protocols but still try to infer sensitive information from users’ search patterns or data. Therefore, there is a critical need for a secure keyword search mechanism that enables users to search encrypted data without revealing the content of either the data or the search queries.

**1.5 EXISTING SYSTEM:**

* Moreover, existing schemes typically focus on a single tenant and are not well suited to extend support to a multi-tenancy cloud, where each tenant operates independently.
* In this to address the above issue, we provide a privacy preserving, verifiable, accountable, and parallelizable solution for “privacy-preserving keyword search problem” among multiple independent data owners.
* We consider a scenario in which each tenant is a data owner and a user’s goal is to efficiently search for granted documents that contain the target keyword among all the data owners.

**1.5.1 Existing System Disadvantages:**

* No Security authentication.
* It cannot be security to a data.

**1.6 ORGANIZATION THESIS**

This thesis is organized into five main chapters to provide a structured understanding of the project and its development. Chapter 1 introduces the concept of cloud computing and highlights the growing importance of multi-tenancy in modern cloud environments. It presents the core problem addressed in this work—ensuring privacy-preserving and trusted keyword search over shared cloud data—and outlines the objectives, scope, and relevance of the project. Chapter 2 provides a comprehensive review of existing literature related to encrypted keyword search, data privacy in cloud computing, and multi-tenancy challenges. It discusses various approaches used in past research and identifies the limitations that this project aims to overcome.

Chapter 3 focuses on the design and architectural details of the proposed system. It explains how the system is structured to support multiple tenants securely, how data and search queries are handled confidentially, and how Java technology is utilized to build the solution. Diagrams such as data flow diagrams (DFD) and use case models are included to represent system behavior and interactions.

**CHAPTER 2**

**LITERATURE SURVEY**

**2.1 LITERATURE SURVEY**

**TITLE**: Attribute-based expressive and ranked keyword search over encrypted documents in cloud computing

**AUTHOR**: Q. Huang, G. Yan, and Q. Wei,

**YEAR**: 2023

**DESCRIPTION**:

In recent years, several new notions of security have begun receiving consideration for public-key cryptosystems, beyond the standard of security against adaptive chosen ciphertext attack (CCA2). Among these are security against randomness reset attacks, in which the randomness used in encryption is forcibly set to some previous value, and against constant secret-key leakage attacks, wherein the constant factor of a secret key’s bits is leaked. In terms of formal security definitions, cast as attack games between a challenger and an adversary, a joint combination of these attacks means that the adversary has access to additional encryption queries under a randomness of his own choosing along with secret-key leakage queries. This implies that both the encryption and decryption processes of a cryptosystem are being tampered under this security notion. In this paper, we attempt to address this problem of a joint combination of randomness and secret-key leakage attacks through two cryptosystems that incorporate hash proof system and randomness extractor primitives. The first cryptosystem relies on the random oracle model and is secure against a class of adversaries, called non-reversing adversaries. We remove the random oracle oracle assumption and the non-reversing adversary requirement in our second cryptosystem, which is a standard model that relies on a proposed primitive called  lossy functions. These functions allow up to *M* lossy branches in the collection to substantially lose information, allowing the cryptosystem to use this loss of information for several encryption and challenge queries. For each cryptosystem, we present detailed security proofs using the game-hopping procedure. In addition, we present a concrete instantiation of lossy functions in the end of the paper—which relies on the DDH assumption.

**TITLE:** Secure keyword search and data sharing mechanism for cloud computing,

**AUTHOR:** C. Ge, W. Susilo, Z. Liu, J. Xia, P. Szalachowski, and L. Fang,

**YEAR:** 2023

**DESCRIPTION:**

The emergence of cloud infrastructure has significantly reduced the costs of hardware and software resources in computing infrastructure. To ensure security, the data is usually encrypted before it's outsourced to the cloud. Unlike searching and sharing the plain data, it is challenging to search and share the data after encryption. Nevertheless, it is a critical task for the cloud service provider as the users expect the cloud to conduct a quick search and return the result without losing data confidentiality. To overcome these problems, we propose a ciphertext-policy attribute-based mechanism with keyword search and data sharing (CPAB-KSDS) for encrypted cloud data. The proposed solution not only supports attribute-based keyword search but also enables attribute-based data sharing at the same time, which is in contrast to the existing solutions that only support either one of two features. Additionally, the keyword in our scheme can be updated during the sharing phase without interacting with the PKG. In this article, we describe the notion of CPAB-KSDS as well as its security model. Besides, we propose a concrete scheme and prove that it is against chosen ciphertext attack and chosen keyword attack secure in the random oracle model. Finally, the proposed construction is demonstrated practical and efficient in the performance and property comparison.

**TITLE:** Omnes pro uno: Practical multi-writer encrypted database

**AUTHOR:** J. Wang and S. S. Chow,

**YEAR:** 2022

**DESCRIPTION:**

Multi-writer encrypted databases allow a reader to search over data contributed by multiple writers securely. Public-key searchable encryption (PKSE) appears to be the right primitive. However, its search latency is not welcomed in practice for having public-key operations linear in the entire database. In contrast, symmetric searchable encryption (SSE) realizes sublinear search, but it is inherently not multi-writer. This paper aims for the best of both SSE and PKSE, i.e., sublinear search and multiple writers, by formalizing hybrid searchable encryption (HSE), with some seemingly conflicting yet desirable features, requiring new insights to achieve.

Our first contribution is a history-based security definition with new flavors of leakage concerning updates and writer corruptions, which are absent in the only known multi-writer notion of PKSE since it is vacuously secure against writers. HSE, built on top of dynamic SSE (DSSE), should satisfy the de facto standard of forward privacy. Its multi-writer support, again, makes the known approach (of secret state maintenance) fails. HSE should also feature efficient controllable search – each search can be confined to a different writer subset, while the search token size remains constant. For these, we devise a new partial rebuild technique and two new building blocks (of independent interests) – ID-coupling key-aggregate encryption and (optimal) epoch-based forward-private DSSE.Our evaluation over real-world datasets shows that HSE, surpassing prior arts by orders of magnitude, is concretely efficient for popular multi-writer database applications.

.

**TITLE:** Multi-authority fine-grained access control with accountability and its application in cloud.

**AUTHOR:** M. W. Zhang, W. X. Song, and J. X. Zhang

**YEAR:** 2022

**DESCRIPTION:**

Attribute-based encryption (ABE) is one of critical primitives for the application of fine-grained access control. To reduce the trust assumption on the attribute authority and in the meanwhile enhancing the privacy of users and the security of the [encryption scheme](https://www.sciencedirect.com/topics/computer-science/encryption-scheme), the notion of multi-authority ABE with an anonymous key issuing protocol has been proposed. In an ABE scheme, it allows to encrypt data for a set of users satisfying some specified attribute policy and any leakage of a decryption key cannot be associated to a user. As a result, a misbehaving user could abuse the property of access [anonymity](https://www.sciencedirect.com/topics/computer-science/anonymity) by sharing its key other unauthorized users. On the other hand, the previous work mainly focus on the key-policy ABE, which cannot support cipher text-policy access control. In this paper, we propose a privacy-aware multi-authority cipher text-policy ABE scheme with accountability, which hides the attribute information in the [ciphertext](https://www.sciencedirect.com/topics/computer-science/ciphertext) and allows to trace the dishonest user identity who shares the decryption key. The efficiency analysis demonstrates that the new scheme is efficient, and the computational overhead in the tracing algorithm is only proportional to the length of the identity. Finally, we also show how to apply it in [cloud computing](https://www.sciencedirect.com/topics/computer-science/cloud-computing) to achieve accountable fine-grained [access control system](https://www.sciencedirect.com/topics/computer-science/access-control-system).

**TITLE:** “A general transformation from KP ABE to searchable encryption

**AUTHOR:** K. Lee

**YEAR:** 2021

**DESCRIPTION:**

Users are inclined to share sensitive data in a remote server if no strong security mechanism is in place. Searchable encryption satisfies the need of users to execute a search [encrypted data](https://www.sciencedirect.com/topics/computer-science/encrypted-data). Previous searchable [encryption methods](https://www.sciencedirect.com/topics/computer-science/encryption-method) such as “public key encryption with [keyword search](https://www.sciencedirect.com/topics/computer-science/keyword-search) (PEKS)” restricted the data access to certain users, because only the assigned users were able to search the [encrypted data](https://www.sciencedirect.com/topics/computer-science/encrypted-data). In this paper we will discuss the relation between [Attribute Based Encryption](https://www.sciencedirect.com/topics/computer-science/attribute-based-encryption) (ABE) and searchable encryption and define a weak [anonymity](https://www.sciencedirect.com/topics/computer-science/anonymity) of the [ABE scheme](https://www.sciencedirect.com/topics/computer-science/encryption-scheme), named “attribute privacy”. With this weak [anonymity](https://www.sciencedirect.com/topics/computer-science/anonymity), we propose a general transformation from ABE to [Attribute Based Encryption](https://www.sciencedirect.com/topics/computer-science/attribute-based-encryption) with Keyword Search (ABEKS) and a concrete [attribute private](https://www.sciencedirect.com/topics/computer-science/private-attribute) key-policy ABE (KP-ABE) scheme. We present an ABEKS scheme based on this KP-ABE scheme and permit multi-users to execute a flexible search on the remote encrypted data.

**TITLE:** “Verifiable dynamic sym metric searchable encryption: Optimality and forward security.

**AUTHOR:** R. Bost, P.-A. Fouque, and D. Pointcheval

**YEAR:** 2023

**DESCRIPTION:**

Symmetric Searchable Encryption (SSE) is a very efficient and practical way for data owners to out- source storage of a database to a server while providing privacy guarantees. Such SSE schemes enable clients to encrypt their database while still performing queries for retrieving documents matching some keyword. This functionality is interesting to secure cloud storage, and efficient schemes have been de- signed in the past. However, security against malicious servers has been overlooked in most previous constructions and these only addressed security against honest-but-curious servers. In this paper, we study and design the first efficient SSE schemes provably secure against malicious servers. First, we give lower bounds on the complexity of such verifiable SSE schemes. Then, we construct generic solutions matching these bounds using efficient verifiable data structures. Finally, we modify an existing SSE scheme that also provides forward secrecy of search queries, and make it prov- ably secure against active adversaries, without increasing the computational complexity of the original scheme.

**CHAPTER 3**

**DESIGN AND DEVELOPMENT**

**3.1.1 PROBLEM STATEMENT/ NEED ANALYSIS:**

Multi-tenant scenarios, there is often a need for shared access, where certain authorized users can perform searches on behalf of others or collaborate using a shared keyword space. Managing this shared access while preserving privacy and ensuring trust in the search results is a complex issue. Current solutions often fall short in terms of trustworthiness, efficiency, and scalability. Hence, this project aims to design and implement a Java-based system that provides a privacy-preserving and trusted keyword search mechanism suitable for multi-tenancy cloud environments. The system will focus on ensuring data security, protecting search queries, enabling authorized shared access, and delivering trustworthy search results—addressing the pressing need for secure and reliable data retrieval in modern cloud systems.

**3.1.1 PROPOSED SYSTEM**

* We first propose a verifiable yet accountable keyword searchable encryption (VAKSE) scheme through symmetric bilinear mapping.
* For verifiability, a message authentication code (MAC) is computed for each associated piece of data. To maintain a consistent size of MAC, the computed Macs undergo an exclusive OR operation.
* For accountability, we propose a keyword-based accountable token mechanism where the client’s identity is seamlessly embedded without compromising privacy. Furthermore, we introduce the parallel VAKSE scheme, in which the inverted index is partitioned into small segments and all of them can be processed synchronously.
* We also conduct formal security analysis and comprehensive experiments to demonstrate the data privacy preservation and efficiency of the proposed schemes, respectively.

**3.1.2 PROPOSED SYSTEM ADVANTAGE**

* Stronger authentication
* It can have access data security.

**3.1.3 SYSTEM REQUIREMENTS DURING PRELIMINARY ANALYSIS:**

**3.1.3.1 HARWARE REQUIREMENTS:**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. It should what the system do and not how it should be implemented.

* PROCESSOR : DUAL CORE 2 DUOS.
* RAM : 2GB DD RAM
* HARD DISK : 250 GB

**3.1.3.2 SOFTWARE REQUIREMENTS:**

* FRONT END : J2EE (JSP, SERVLET)
* BACK END : MY SQL 5.5
* OPERATING SYSTEM : WINDOWS 10
* IDE : ECLIPSE

**3.1.3.3 FUNCTIONAL REQUIREMENTS**

A functional requirement defines a function of a software-system or its component. A function is described as a set of inputs, the behavior, Firstly, the system is the first that achieves the standard notion of semantic security for data confidentiality in attribute-based deduplication systems by resorting to the hybrid cloud architecture.

**3.1.3.4 NON-FUNCTIONAL REQUIREMENTS**

Our multi-modal event tracking and evolution framework is suitable for multimedia documents from various social media platforms, which can not only effectively capture their also obtain the evolutionary trends of social events and generate effective event summary details over time. Our proposed mmETM model can exploit the multi-modal property

**SYSTEM** **ACHITECTURE:**

CSP

Verifier

Client’s

Data Owner

Login

Data Owner Details

Client’s Data

All Files Info

Data Request

Login

Login

Key Test

Verify Key

Login

Register

File Upload

Register

Verification

Decrypt Data Download

Query Data

Token Generated

Database

Key Generator

Key Generated

Fig: System Architecture

**EXPLAINATION:**

This diagram represents the architecture of a privacy-preserving and trusted keyword search system in a multi-tenancy cloud environment. It involves three main entities: Data Owners, Clients, and the Cloud Service Provider (CSP). The process begins with data owners, who detect sensitive information and then encrypt both the data and corresponding keyword indexes before outsourcing them to the CSP. This ensures that the CSP, though responsible for storing and managing the data, cannot access the actual content due to encryption. Clients interested in searching data must first go through a registration phase, during which they are authenticated and issued necessary credentials or keys. Once registered, a client initiates a search by sending a request. The system checks whether the client is authorized to access the particular data set. If authorized, the client sends an encrypted keyword query to the CSP. The CSP processes the query over

**3.2.1 JUSIFYING DESIGN:**

Justifying the design involves explaining the reasoning behind the choices made during the design process to ensure they align with the project's goals, user needs, and technical constraints. The design was created with a user-centered approach, focusing on simplicity and ease of use to enhance the overall user experience. Visual elements such as color, layout, and typography were carefully selected to create a clean and professional look while maintaining readability and accessibility. Functional aspects, including navigation and placement of interactive elements, were optimized for efficiency and usability, particularly on mobile devices. Additionally, the design takes into account technical feasibility, ensuring that it can be implemented effectively within the given constraints. Overall, each design decision supports the intended purpose and contributes to a coherent, functional, and user-friendly outcome.

**3.3. DESIGN METHODOLOGY:**

3.3.1 **GENERAL:**

In the above schemes, the client is trusted. In reality, dishonest users may attempt to access data without authorization. Even worse, some users may give away some of their original or transformed keys such that nobody can tell who has distributed these keys. The first problem is called unauthorized access. The second problem is called key abuse. The first problem can be prevented by fine-grained access control, and the second problem can be discouraged by user accountability.

**3.3.2 METHODOLOGIES**

**3.3.2.1 MODULES NAME:**

1. User interface design

2. Csp

3. Verifier

4. Data owner

5.Client

**1. User Interface Design**

In this module we design the windows for the project. These windows are used for secure login for all users. To connect with server user must give their username and password then only they can able to connect the server. If the user already exits directly can login into the server else user must register their details such as username, password and Email id, into the server. Server will create the account for the entire user to maintain upload and download rate. Name will be set as user id. Logging in is usually used to enter a specific page.

User Login

Server

Database

Home Page

Register &Login Page

**2. CSP**

This is the first module the cloud services has a login with a mail id and password. The CSP has a data owner details. The CSP has a generated a key. The CSP has a client’s details. The

CSP have a requested a data .The CSP has a stored a file details. The CSP has a clients add all the members.

Generate Key

CSP

Data Base

Data Owner Details

Client’s Details

Requested Data

Stored File Details

Client Add

**3. Verifier**

This is the Second module of this project the Verifier has a login with a mail id and password. A verifier has a key has a key test it is generated original key or duplicate key. It also have a verify a key then it will send to the data receiver. The verifier has a verification key matching to the d

Verification Key Matching

Key Test

Verifier

Data Base

**4. Data Owner**

This is the third module of this project. In this project data holder has a register with all details and login with a user id and password. The data owner has a file upload to store a data. The data owner has a key generated to share a data.

File Upload

Data Owner

Data Base

Key Generated

**5. Clients’**

This is the fourth module of this project. Clients has a register with all details and then login with a email id and password. The clients has a query data. The client has a token generated to a private key. The client has a verification a data. The Client has a decrypt a data

Decrypt Data Download

Verification

Token Generated

Query Data

Clients

Data Base

**GIVEN INPUT EXPECTED OUTPUT:**

* **User Interface Design**

Input : Enter Login name and Password

Output : If valid user name and password then directly open the home page otherwise show error message and redirect to the registration page.

* **Client’s**

Input : User has a Email and Password

Output: If valid user name and password then directly open the client home page otherwise show error message and redirect to the client login page.

* **Data Owner**

Input : Enter the Email and password

Output : If valid email and password then directly open the data owner home page otherwise show error message and redirect to the data owner login page.

* **Verifier**

Input : Enter the mail and password

* Output: If valid mail and password then directly open the verifier home page otherwise show error message and redirect to the verifier login page.
* **CSP**

Input : Enter the mail id and password

Output: If valid CSP id and password then directly open the CSP home page otherwise show error message and redirect to the CSP login page.

**3.3.3 TECHNIQUE USED OR ALGORITHM USED**

**PROPOSED ALOGORITHM:**

**1.VERIFIABLE YET ACCOUNTABLE KEYWORD SEARCHABLE ENCRYPTION (VAKSE) SCHEME**

A VAKSE consists of four PPT algorithms: Setup, KeyGen, Encap, ⁎ Decap. These algorithms have the following syntax.

**Setup(1).** The setup algorithm takes a security parameter *κ* as input, and produces the master [public key](https://www.sciencedirect.com/topics/computer-science/public-key) *mpk* and the master secret key ∈MSK, where MSK denotes the master secret key space. The master public key *mpk* defines an identity space ID, an encapsulated-key space  and an identity private key space SK. All other algorithms KeyGen, Encap, ⁎, Decap  implicitly include *mpk* as partial input.

**KeyGen(k).** For any identity ∈ID, the key generation algorithm KeyGen uses the master secret key *msk* to sample a private key ∈SK associated with the identity *id*. We stress that, denotes the auxiliary output.

**Encrypt(e).** The valid algorithm Encap creates ciphertext and encapsulated-key pairs  associated with the identity ∈ID, where *C* is a valid encapsulation ciphertext and e∈k is an encapsulated-key.

**Encrypt(E1).** The invalid algorithm ⁎Enca⁎ samples an invalid ciphertext *C*, and takes an identity ∈ID as input. Note that, the algorithm ⁎Enca⁎ only outputs an invalid encapsulation ciphertext.

**Decrpt(D).** The decrypt algorithm is deterministic. It takes a ciphertext *C* and a private key of identity *id* as input, and outputs the encapsulated-key *k*.

**2. SHA-256 (SECURE HASH ALGORITHM 256)**

SHA-256 (Secure Hash Algorithm 256) is a widely used cryptographic algorithm that produces a fixed-length, 256-bit (32-byte) hash value. The purpose of the SHA-256 algorithm is to create a unique digital fingerprint of a piece of data, such as a message or a file.

Generating an SHA-256 hash involves running the input data through a complex mathematical function that produces a unique output value. This output value is the hash, which serves as a digital fingerprint of the input data.

The SHA-256 algorithm is used in many applications, such as digital signatures, password authentication, and blockchain technology. Because the hash value produced by SHA-256 is unique, it is virtually impossible to reverse-engineer the input data**.**

A hash is not ‘encryption’ – it cannot be decrypted back to the original text (it is a ‘one-way’ cryptographic function, and is a fixed size for any size of source text).

**3.4 COMPONENT DESIGN/ SUBSYSTEM**

**3.4.1 GENERAL**

Design Engineering deals with the various UML [Unified Modelling language] diagrams for the implementation of project. Design is a meaningful engineering representation of a thing that is to be built. Software design is a process through which the requirements are translated into representation of the software. Design is the place where quality is rendered in software engineering. Design is the means to accurately translate customer requirements into finished product.

**3.4.2 USE CASE DIAGRAM**

****

FIG:3.4.2 Use Case diagram

**EXPLANATION:**

The primary objective of a use case diagram is to demonstrate the existence of an actor who functions as a data user, a data owner, and a cloud server. Various types of actions are executed. The actor acting as the data user performs actions like searching files, viewing responses, and viewing search keywords. The data owner, who is also an actor, carries out actions such as uploading files and viewing uploaded files. The cloud server, which can also be an actor, performs actions like viewing responses, viewing files, matching keywords, viewing data users, and viewing data owners. All of these actions are executed by the actor.

**3.4.3 CLASS DIAGRAM**

****

**FIG 3.4.3: Class Diagram**

**EXPLANATION**

This class diagram illustrates the interconnection of classes with their respective attributes and methods to carry out security verification. In our project, multiple classes are implicated based on the aforementioned diagram. The data user possesses a class that encompasses attributes and operations. Within this class, there is a login attribute. The operations include searching files, viewing responses, and examining search keywords. On the other hand, the data owner represents a distinct class. Its attributes consist of registration and login, while the operations encompass file uploading and viewing uploaded files. When the cloud server and database are interconnected, all information is stored in a database.

**3.4.4. OBJECT DIAGRAM**



FIG 3.4.4 Object Diagram

**EXPLANATION:**

The above diagram depicts the flow of objects between the classes. It is a diagram that provides a comprehensive or partial view of the structure of a modeled system. In this object diagram, it illustrates how the classes, with their attributes and methods, are interconnected to carry out secure verification. The data user is linked to the data owner's information, and both are connected to a cloud server. All the information is gathered in a database.

**3.4.5 COMPONENT DIAGRAM**

****

FIG 3.4.5 : Component Diagram

**EXPLANATION**

In the Unified Modeling Language, a component diagram depicts how components are connected to form larger components and software systems. The user provides the main query, which is then converted into sub queries and sent through data dissemination to data aggregators. The data user can connect with a data owner to obtain information. The data owner can also depend on a cloud server or store data in a database. The results are displayed to the user by database aggregators. In the diagram, each box represents a component and the arrows indicate dependencies.

**3.4.6 DEPLOYMENT DIAGRAM**



**Fig 3.4.6 : Deployment Diagram**

**EXPLANATION:**

The Deployment Diagram is a type of diagram that specifies the physical hardware on which the software system will execute. It also determines how the software is deployed on the underlying hardware. The user can first log in. Afterwards, they will connect with a database.

**3.4.7 SEQUENCE DIAGRAM**



Fig 3.4.7 : Sequence Diagram

**EXPLANATION:**

Sequence diagrams are graphical representations of stepwise activities and actions in workflows, supporting choice, iteration, and concurrency. In the Unified Modeling Language, sequence diagrams can be sequential. In a sequence diagram, the data user, data owner, and cloud server are connected with a database. They are used to describe the business and operational workflows of components in a system. The UML clearly defines the sequence order.

**3.4.8 COLLABORATION DIAGRAM**



Fig 3.4.8: Collaboration Diagram

**EXPLANATION:**

A collaboration diagram, also known as a communication diagram or interaction diagram, is a way to illustrate the relationships and interactions among software objects in the Unified Modeling Language (UML). It facilitates communication between a data user, data owner, and a database. The cloud server communicates with the database.

**3.4.9 STATE DIAGRAM**

****

Fig 4.2.8: State Diagram

**EXPLANATION:**

State diagrams are loosely defined diagrams used to illustrate workflows of stepwise activities and actions, including support for choice, iteration, and concurrency. State diagrams require that the system described, a data user, has a login. Once logged in, the data user can perform actions. The data owner also needs to have a login. The data owner has the ability to upload

**3.4.10 ACTIVITY DIAGRAM**

****

Fig 3.4.10: Activity Diagram

**EXPLANATION:**

Activity diagrams are graphical representations of workflows of stepwise activities of a data user, data owner, and cloud server actions with support for choice, iteration, and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. The data user can also search for a file, view a response file, and perform a search with a keyword. The data owner can upload a file and then view the uploaded file. The information is gathered in a database. The cloud server can view requests, view files, and perform a matched keyword search in a stepwise manner. An activity diagram shows the overall flow of control.

**3.4.11 DATA FLOW DIAGRAM**

**Level 0**

Register

Home Page

Login

Verify Details

Data base

Error page

**EXPLANATION:**

A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modeling its process aspects. Often, DFDs are a preliminary step used to create an overview of the system that can later be elaborated upon. DFDs can also be used to visualize data processing (structured design).

Data Ownerdetails

Client’s Data

All Files info

Verify Key Client’s

Request Data

Data base

Key Test

Verification Key Matching

File Upload

Testing Key

Query Data

Token Generated

Decrypt Download

**3.4.12 E-R DIAGRAM**

Decrypt File Download

Data Owner Details

Data base

Verify Key

Stored File Info

Key Test

Token Generated

Verification

Query Data

CSP

Verifier

Client’s

Verify

Details

Clients Data

Request data

All Files info

File Upload

Owner Data

**EXPLANATION:**

The entity-relationship model (ERM) is a data model where a user has a username and password. If the username and password are correct, the user will be able to log in. After logging in, the user can search for files and request a specific file. The data owner also has a username and password.

**3.5 TOOLS AND TECHNOLOGIES:**

This chapter is about the software language and the tools used in the development of the project. The platform used here is JAVA. The Primary languages are JAVA,J2EE and J2ME. In this project J2EE is chosen for implementation.

**3.5.1 FEATURES OF JAVA**

**3.5.1.1THE JAVA FRAMEWORK**

Java is a programming language originally developed by James Gosling at Microsystems and released in 1995 as a core component of Sun Microsystems' Java platform. The language derives much of its syntax from C and C++ but has a simpler object model and fewer low-level facilities. Java applications are typically compiled to byte code that can run on any Java Virtual Machine (JVM) regardless of computer architecture. Java is general-purpose, concurrent, class-based, and object-oriented, and is specifically designed to have as few implementation dependencies as possible. It is intended to let application developers "write once, run anywhere".

Java is considered by many as one of the most influential programming languages of the 20th century, and is widely used from application software to web applications the java framework is a new platform independent that simplifies application development internet. Java technology's versatility, efficiency, platform portability, and security make it the ideal technology for network computing. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

**3.5.1.2 OBJECTIVES OF JAVA:**

To see places of Java in Action in our daily life, explore java.com.

## WHY SOFTWARE DEVELOPERS CHOOSE JAVA

Java has been tested, refined, extended, and proven by a dedicated community. And numbering more than 6.5 million developers, it's the largest and most active on the planet. With its versatility, efficiency, and portability, Java has become invaluable to developers by enabling them to:

* Write software on one platform and run it on virtually any other platform
* Create programs to run within a Web browser and Web services
* Develop server-side applications for online forums, stores, polls, HTML forms processing, and more
* Combine applications or services using the Java language to create highly customized applications or services
* Write powerful and efficient applications for mobile phones, remote processors, low-cost consumer products, and practically any other device with a digital heartbeat

## SOME WAYS SOFTWARE DEVELOPERS LEARN JAVA

Today, many colleges and universities offer courses in programming for the Java platform. In addition, developers can also enhance their Java programming skills by reading Sun's java.sun.com Web site, subscribing to Java technology-focused newsletters, using the Java Tutorial and the New to Java Programming Center, and signing up for Web, virtual, or instructor-led courses.

**OBJECT ORIENTED**

To be an Object-Oriented language, any language must follow at least the four characteristics.

1. Inheritance : It is the process of creating the new classes and using the behavior of the existing classes by extending them just to reuse the existing code and adding addition a feature as needed.

2. Encapsulation: It is the mechanism of combining the information and providing the abstraction.

3. Polymorphism: As the name suggest one name multiple form, Polymorphism is the way of providing the different functionality by the functions having the same name based on the signatures of the  methods.

4. Dynamic binding: Sometimes we don't have the knowledge of objects about their specific types while writing our code. It is the way of providing the maximum functionality to a program about the specific type at runtime.

**3.5.1.3 JAVA SWING OVERVIEW**

**ABSTRACT WINDOW TOOLKIT (AWT) IS CROSS-PLATFORM**

Swing provides many controls and widgets to build user interfaces with. Swing class names typically begin with a J such as JButton, JList, JFrame. This is mainly to differentiate them from their AWT counterparts and in general is one-to-one replacements. Swing is built on the concept of Lightweight components vs AWT and SWT's concept of Heavyweight components. The difference between the two is that the Lightweight components are rendered (drawn) using purely Java code, such as drawLine and draw Image, whereas Heavyweight components use the native operating system to render the components. Some components in Swing are actually heavyweight components. The top-level classes and any derived from them are heavyweight as they extend the AWT versions. This is needed because at the root of the UI, the parent windows need to be provided by the OS. These top-level classes include JWindow, JFrame, JDialog and  JApplet. All Swing components to be rendered to the screen must be able to trace their way to a root window of one of those classes.

**NOTE**: It generally it is not a good idea to mix heavyweight components with lightweight components (other than as previously mentioned) as you will encounter layering issues, e.g., a lightweight component that should appear "on top" ends up being obscured by a heavyweight component. The few exceptions to this include using heavyweight components as the root pane and for popup windows. Generally speaking, heavyweight components will render on top of lightweight components and will not be consistent with the look and feel being used in Swing. There are exceptions, but that is an advanced topic. The truly adventurous may want to consider reading this [article](http://java.sun.com/products/jfc/tsc/articles/mixing/) from Sun on mixing heavyweight and lightweight components.

**3.5.1.4 EVOLUTION OF COLLECTION FRAMEWORK:**

Almost all collections in Java are derived from the [java.util.Collection](http://download.oracle.com/javase/7/docs/api/java/util/Collection.html) interface. Collection defines the basic parts of all collections. The interface states the add() and remove() methods for adding to and removing from a collection respectively. Also required is the toArray() method, which converts the collection into a simple array of all the elements in the collection. Finally, the contains() method checks if a specified element is in the collection. The Collection interface is a subinterface of [java.util.Iterable](http://download.oracle.com/javase/7/docs/api/java/util/Iterable.html), so the iterator() method is also provided. All collections have an iterator that goes through all the elements in the collection. Additionally, Collection is a generic. Any collection can be written to store any class. For example, Collection<String> can hold strings, and the elements from the collection can be used as strings without any casting required.

There are three main types of collections:

* Lists: always ordered, may contain duplicates and can be handled the same way as usual arrays
* Sets: cannot contain duplicates and provide random access to their elements
* Maps: connect unique keys with values, provide random access to its keys and may host duplicate values

**LIST:**

Lists are implemented in the JCF via the java.util.List interface. It defines a list as essentially a more flexible version of an array. Elements have a specific order, and duplicate elements are allowed. Elements can be placed in a specific position. They can also be searched for within the list. Two concrete classes implement List. The first is java.util.ArrayList, which implements the list as an array. Whenever functions specific to a list are required, the class moves the elements around within the array in order to do it. The other implementation is java.util.LinkedList. This class stores the elements in nodes that each have a pointer to the previous and next nodes in the list. The list can be traversed by following the pointers, and elements can be added or removed simply by changing the pointers around to place the node in its proper place.

**SET:**

Java's [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface defines the set. A set can't have any duplicate elements in it. Additionally, the set has no set order. As such, elements can't be found by index. Set is implemented by java.util.HashSet,java.util.LinkedHashSet, and java.util.TreeSet. HashSet uses a hash table. More specifically, it uses a [java.util.HashMap](http://download.oracle.com/javase/7/docs/api/java/util/HashMap.html) to store the hashes and elements and to prevent duplicates. Java.util.LinkedHashSet extends this by creating a doubly linked list that links all of the elements by their insertion order. This ensures that the iteration order over the set is predictable. [java.util.TreeSet](http://download.oracle.com/javase/7/docs/api/java/util/TreeSet.html) uses a red-black tree implemented by a [java.util.TreeMap](http://download.oracle.com/javase/7/docs/api/java/util/TreeMap.html). The red-black tree makes sure that there are no duplicates. Additionally, it allows Tree Set to implement java.util.SortedSet.

The [java.util.Set](http://download.oracle.com/javase/7/docs/api/java/util/Set.html) interface is extended by the java.util.SortedSet interface. Unlike a regular set, the elements in a sorted set are sorted, either by the element's compareTo() method, or a method provided to the constructor of the sorted set. The first and last elements of the sorted set can be retrieved, and subsets can be created via minimum and maximum values, as well as beginning or ending at the beginning or ending of the sorted set. The SortedSet interface is implemented by java.util.TreeSet

[java.util.SortedSet](http://download.oracle.com/javase/7/docs/api/java/util/SortedSet.html) is extended further via the java.util.NavigableSet interface. It's similar to SortedSet, but there are a few additional methods. The floor(), ceiling(), lower(), and higher() methods find an element in the set that's close to the parameter. Additionally, a descending iterator over the items in the set is provided. As with SortedSet, java.util.TreeSet implements NavigableSet.

**MAP:**

Maps are defined by the java.util.Map interface in Java. Maps are simple data structures that associate a key with a value. The element is the value. This lets the map be very flexible. If the key is the hash code of the element, the map is essentially a set. If it is just an increasing number, it becomes a list. Maps are implemented by java.util.HashMap, java.util.LinkedHashMap, and java.util.TreeMap. HashMap uses a hash table. The hashes of the keys are used to find the values in various buckets. LinkedHashMap extends this by creating a doubly linked list between the elements. This allows the elements to be accessed in the order in which they were inserted into the map. TreeMap, in contrast to HashMap and LinkedHashMap, uses a red-black tree. The keys are used as the values for the nodes in the tree, and the nodes point to the values in the map

**THREAD:**

Simply put, a threadis a program's path of execution. Most programs written today run as a single thread, causing problems when multiple events or actions need to occur at the same time. Let's say, for example, a program is not capable of drawing pictures while reading keystrokes. The program must give its full attention to the keyboard input lacking the ability to handle more than one event at a time. The ideal solution to this problem is the seamless execution of two or more sections of a program at the same time.

## CREATING THREADS

Java's creators have graciously designed two ways of creating threads: implementing an interface and extending a class. Extending a class is the way Java inherits methods and variables from a parent class. In this case, one can only extend or inherit from a single parent class. This limitation within Java can be overcome by implementing interfaces, which is the most common way to create threads. (Note that the act of inheriting merely allows the class to be run as a thread. It is up to the class to start() execution, etc.)

Interfaces provide a way for programmers to lay the groundwork of a class. They are used to design the requirements for a set of classes to implement. The interface sets everything up, and the class or classes that implement the interface do all the work. The different set of classes that implement the interface have to follow the same rules.

**3.1.4.5 CONCLUSION**

Swing's high level of flexibility is reflected in its inherent ability to override the native host [operating system](http://en.wikipedia.org/wiki/Operating_system) (OS)'s GUI controls for displaying itself. Swing "paints" its controls using the Java 2D APIs, rather than calling a native user interface toolkit. The Java thread scheduler is very simple. All threads have a priority value which can be changed dynamically by calls to the threads setPriority () method. Implementing the above concepts in our project to do the efficient work among the Server.

**3.6 IMPLEMENATATION/ DEVELOPMENT:**

**Coding:**

**1.Ownerbean.Java**

public class Owner Bean {

private String name, email, pwd, mob,age;

public String getAge() {

        return age;

    }

public void setAge(String age) {

this.age = age;

    }

public String getName() {

        return name;

    }

public void setName(String name) {

        this.name = name;

    }

public String getEmail() {

        return email;

}

public void setEmail(String email) {

        this.email = email;

    }

public String getPwd() {

return pwd;

    }

public void setPwd(String pwd) {

        this.pwd = pwd;

    }

public String getMob () {

        return mob;

    }

  public void setMob(String mob) {

        this.mob = mob;

    }

}

2.Uploadbean.Java

import java.io.InputStream;

public class UploadBean {

private String filename,filekeyword, content, enc, tkey,fid,email,label,branch,dept,op,hashCode,MACenc;

public String getFilekeyword() {

    return filekeyword;

}

public void setFilekeyword(String filekeyword) {

    this.filekeyword = filekeyword;

}

public String getBranch() {

    return branch;

}

public void setBranch(String branch) {

    this.branch = branch;

}

public String getDept() {

    return dept;

}

public String getMACenc() {

    return MACenc;

}

public void setMACenc(String mACenc) {

    MACenc = mACenc;

}

public InputStream getPhoto1() {

    return photo1;

}

public void setDept(String dept) {

    this.dept = dept;

}

public String getOp() {

    return op;

}

public void setOp(String op) {

    this.op = op;

}

public String getLabel() {

    return label;

}

public void setLabel(String label) {

    this.label = label;

}

public String getFilename() {

    return filename;

}

public void setFilename(String filename) {

    this.filename = filename;

}

public String getContent() {

    return content;

}

public void setContent(String content) {

    this.content = content;

}

public String getEnc() {

    return enc;

}

public void setEnc(String enc) {

    this.enc = enc;

}

public String getTkey() {

    return tkey;

}

public void setTkey(String tkey) {

    this.tkey = tkey;

}

public String getFid() {

    return fid;

}

public void setFid(String fid) {

    this.fid = fid;

}

public String getEmail() {

    return email;

}

public void setEmail(String email) {

    this.email = email;

}

private InputStream photo1;

public void setPhoto1(InputStream photo1) {

*// TODO Auto-generated method stub*

    this.photo1 = photo1;

}

public InputStream getPhoto() {

*// TODO Auto-generated method stub*

    return photo1;

}

public String getHashCode() {

    return hashCode;

}

public void setHashCode(String hashCode) {

    this.hashCode = hashCode;

}

}

3.USERBEAN.JAVA

package com.beans;

public class UserBean {

    private String name, email, pwd, mob,age;

    public String getAge() {

        return age;

    }

    public void setAge(String age) {

        this.age = age;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

    public String getPwd() {

        return pwd;

    }

    public void setPwd(String pwd) {

        this.pwd = pwd;

    }

    public String getMob() {

        return mob;

    }

    public void setMob(String mob) {

        this.mob = mob;

    }

}

3.ADMIN

package com.beans;

public class UserBean {

    private String name, email, pwd, mob,age;

    public String getAge() {

        return age;

    }

    public void setAge(String age) {

        this.age = age;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

public String getEmail() {

        return email;

    }

    public void setEmail(String email) {

        this.email = email;

    }

public String getPwd() {

        return pwd;

    }

public void setPwd(String pwd) {

        this.pwd = pwd;

    }

public String getMob() {

        return mob;

    }

public void setMob(String mob) {

        this.mob = mob;

    }

}

4.Adminapprove Request

package com.servlets;

import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import com.Dao.DBConnection;

*/\*\**

*\* Servlet implementation class AdminApproveRequest*

*\*/*

@WebServlet("/AdminApproveRequest")

public class AdminApproveRequest extends HttpServlet {

    private static final long serialVersionUID = 1L;

*/\*\**

*\* @see HttpServlet#HttpServlet()*

*\*/*

    public AdminApproveRequest() {

        super();

*// TODO Auto-generated constructor stub*

    }

*/\*\**

*\* @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)*

*\*/*

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

*// TODO Auto-generated method stub*

    }

*/\*\**

*\* @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)*

*\*/*

    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {

*// TODO Auto-generated method stub*

         PrintWriter out = response.getWriter();

            String email = request.getParameter("email"); *// Get the email from the request*

*// SQL query to update the student's status*

*// Logic to approve the student*

            String sql = "UPDATE student SET Status = 'Approved' WHERE email = ?";

            ResultSet rs = DBConnection.getData2(sql);

            try (Connection conn = DBConnection.getStudent1(sql); *// Your method to get a connection*

                 PreparedStatement pstmt = conn.prepareStatement(sql)) {

                pstmt.setString(1, email); *// Set the email parameter*

                int rowsAffected = pstmt.executeUpdate(); *// Execute the update*

                if (rowsAffected > 0) {

*// If the update was successful*

                    response.sendRedirect("Added.jsp?message=Approved successfully");

                } else {

*// If no rows were updated (email not found)*

                    response.sendRedirect("Added.jsp?message=No student found with the provided email");

                }

            } catch (SQLException e) {

                e.printStackTrace(); *// Print the stack trace for debugging*

                response.sendRedirect("Added.jsp?message=Error occurred while approving");

            }

        }

    }

**3.7 TESTING AND VALIDATION:**

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

**3.7.1 DEVELOPING METHODOLOGIES:**

The test process is initiated by developing a comprehensive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

**3.7.2 TYPES OF TESTS**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Sl. No | Test scenario | User action | Expected result | Actual Result | Remarks |
| 1. | Registration | Users registering into the system. | Register into the system. | Successfully alert registered message. | Pass |
| 2. | Login | 1. Entered correct password. | 1. Log into the system.  2. Alert generated. | 1. Successfully logged in.  2. Successfully generated the alert. | Pass |
| 3. | Cloud Services | It have a all details stores in cloud | Massages sending data user alert is generated. | Successfully generated the alert and massages sending | Successful |
| 4. | Verifier | It will have keys test and verify | key has to actions | Successfully generated the alert to Verifier message | Successful |
| 5. | Data Owner | Owner Data has a store a file | Massages Alert is generated | Successfully generated the alert for data controller massage has displays | Successful |
| 6. | Client’s | clients has a search file and it will have a key response | Massages Alert is generated | Successfully generated the alert for client massage has displays | Successful |

**3.7.2.1 UNIT TESTING**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program input produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**3.7.2.2 FUNCTIONAL TEST**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

**3.7.2.3 SYSTEM TEST**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**3.7.2.4 PERFORMANCE TEST**

The Performance test ensures that the output be produced within the time limits,and the time taken by the system for compiling, giving response to the users and request being send to the system for to retrieve the results.

**3.7.2.5 INTEGRATION TESTING**

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**3.7.2.6 ACCEPTANCE TESTING**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Acceptance Testing For Data Synchronization:**

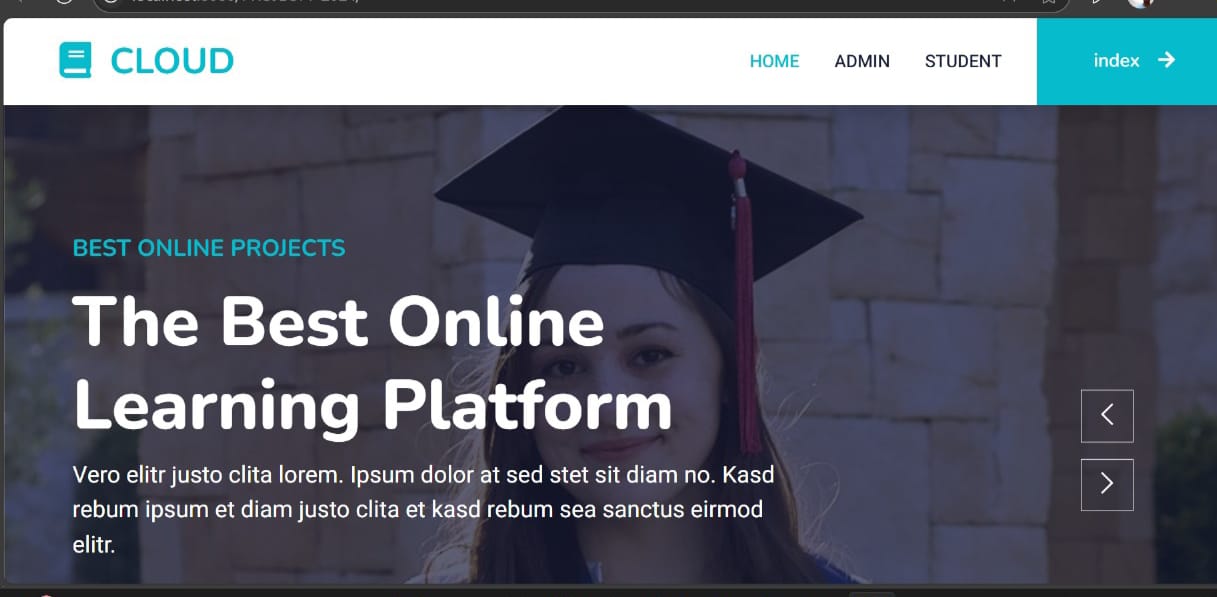
* The Acknowledgements will be received by the Sender Node after the Packets are received by the Destination Node
* The Route add operation is done only when there is a Route request in need
* The Status of Nodes information is done automatically in the Cache Updation process

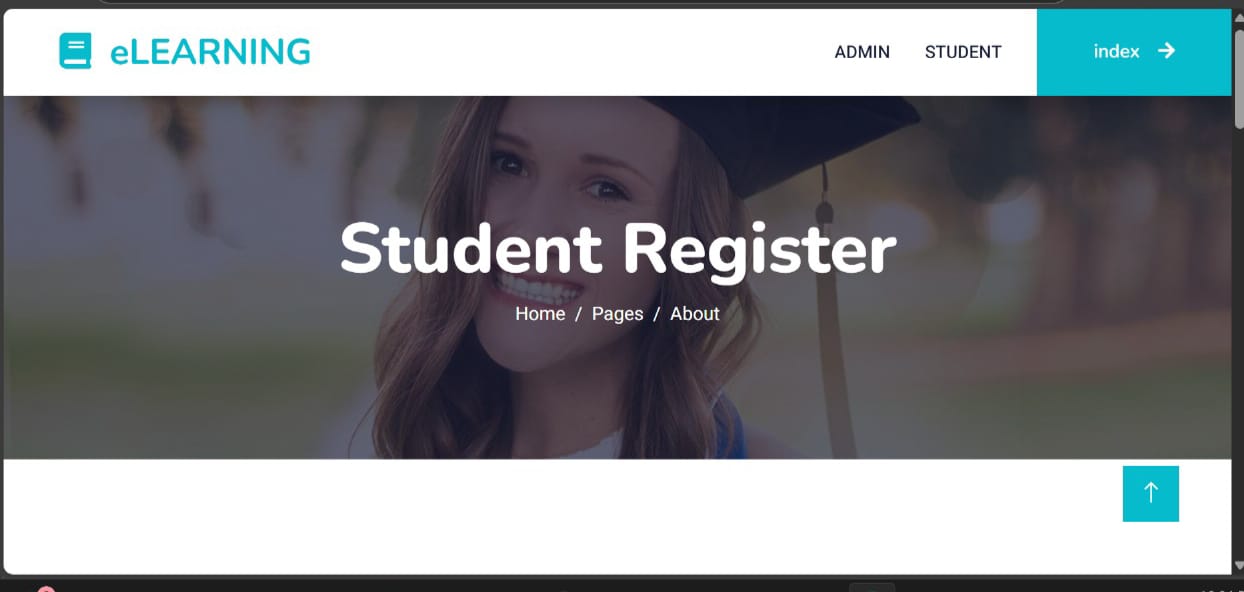
**3.7.2.7 BUILD THE TEST PLAN:**

Any project can be divided into units that can be further performed for detailed processing. Then a testing strategy for each of this unit is carried out. Unit testing helps to identity the possible bugs in the individual component, so the component that has bugs can be identified and can be rectified from errors.

**3.8 PHOTOGRAPHS/SNAPSHOTS:**

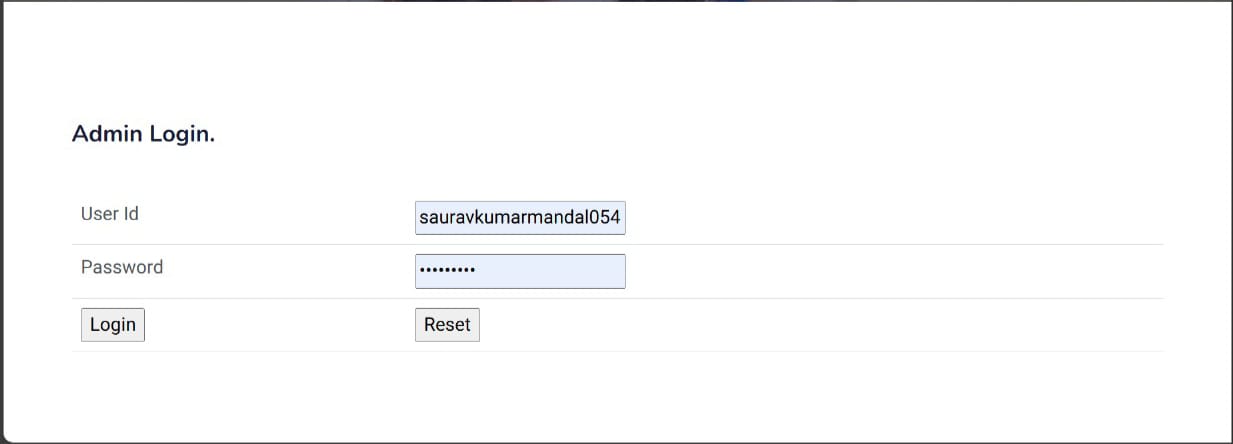
This project is implements like web application using COREJAVA and the Server process is maintained using the SOCKET & SERVERSOCKET and the Design part is played by Cascading Style Sheet.



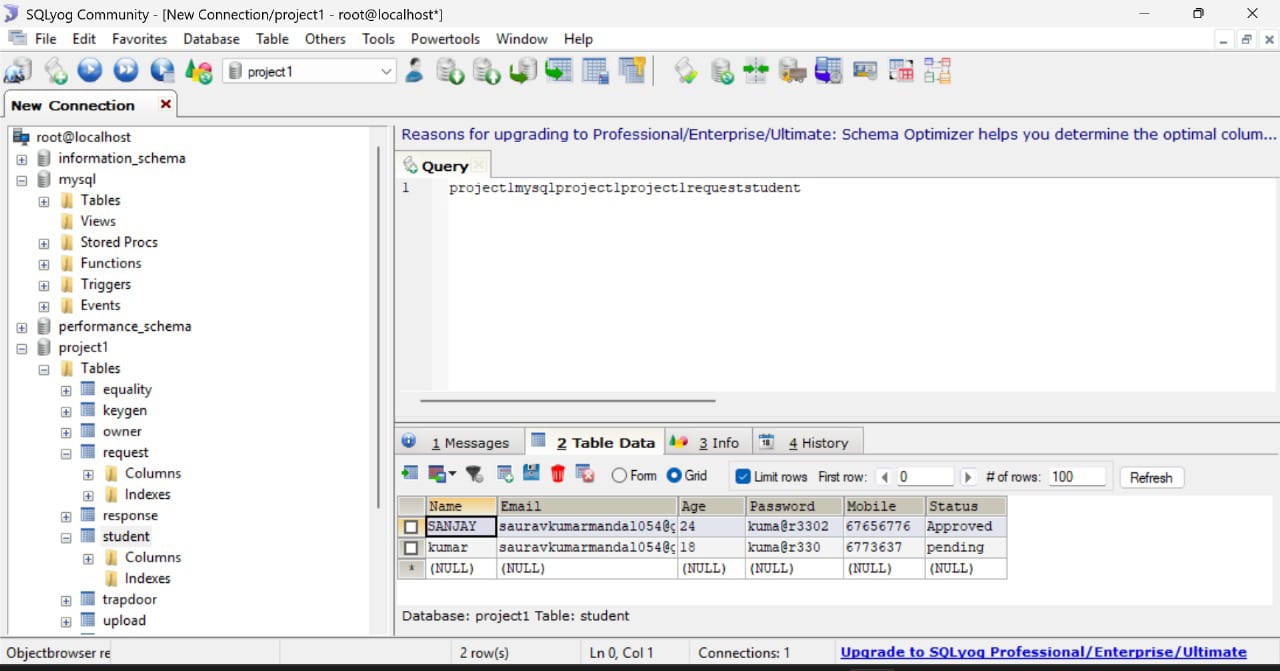


Header – “eLEARNING”

Represents a cloud-based platform (multi-tenancy) where multiple students and admins access resources.Connects to your project idea where multiple tenants (students/admins) store/search data securely.

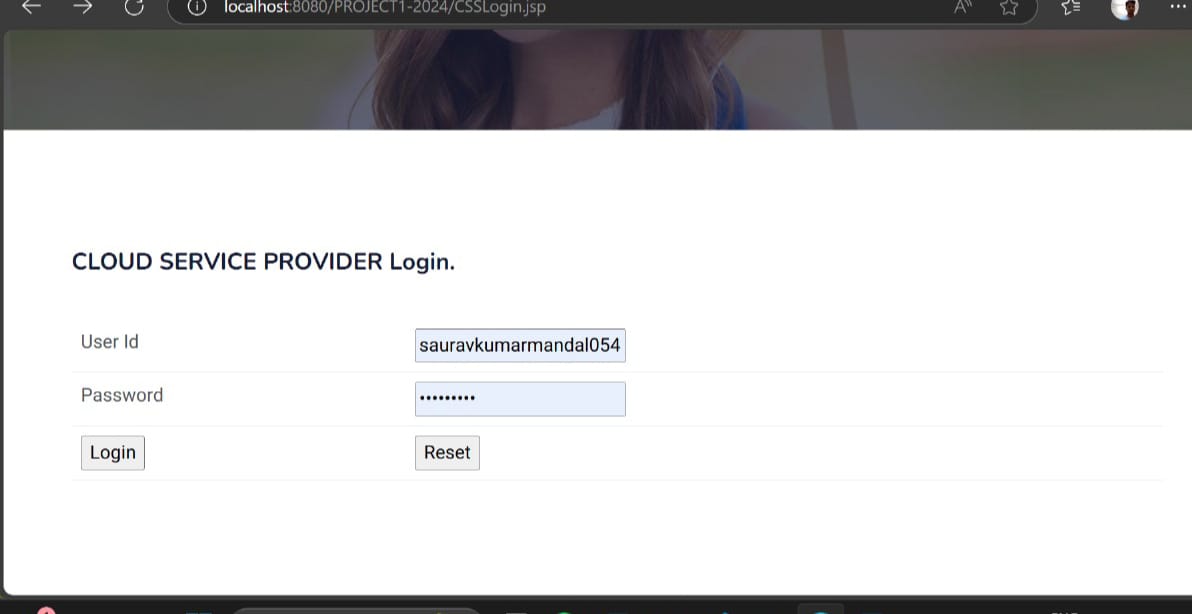


This Admin Login page is a key part of access control in a multi-tenant cloud system. It ensures that only authorized administrators can manage encrypted student data and oversee secure keyword searches. In your project, the credentials would be verified without revealing sensitive details, supporting privacy and trust in a cloud environment

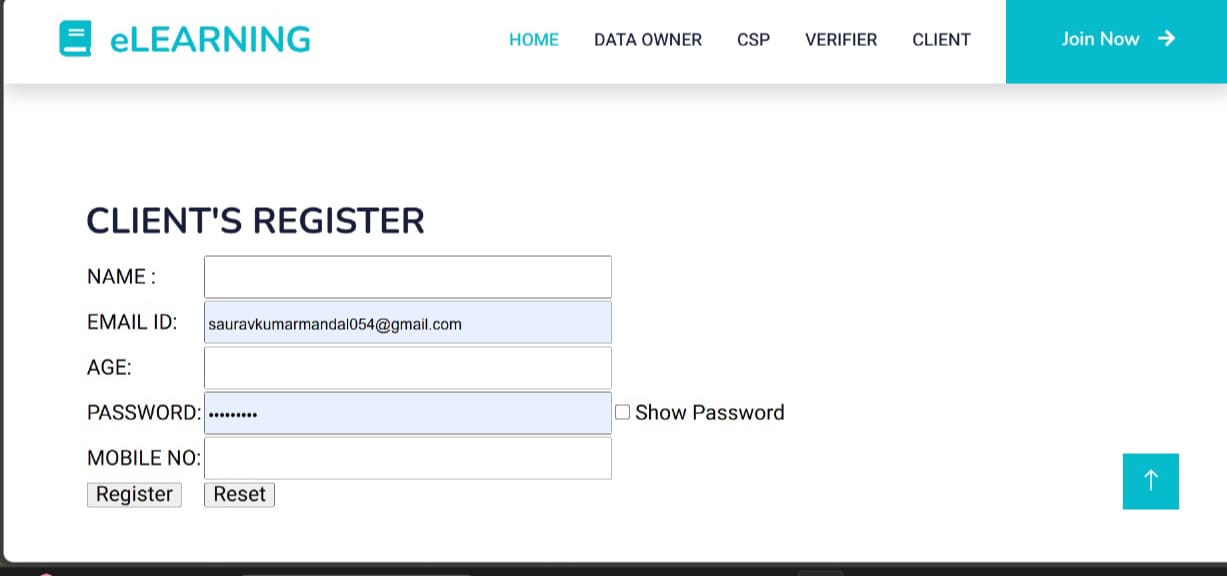


This screenshot displays the student table under the project1 database in SQLyog. It stores user data such as name, email, age, password, mobile number, and status (approved/pending).

This is the homepage of the project, showcasing a cloud-based learning platform with secure access for Admin and Student users. It serves as the main entry point for accessing project



The displayed page is a login interface for a Cloud Service Provider (CSP) web application. It is hosted locally, as indicated by the URL starting with "localhost:8080", and is part of a project titled "PROJECT1-2024" with the JSP file named "CSSLogin.jsp". The interface includes a title that reads "CLOUD SERVICE PROVIDER Login." followed by two input fields labeled "User Id" and "Password".



The displayed image shows a client registration page for an eLearning platform. At the top, the navigation bar includes options such as Home, Data Owner, CSP, Verifier, and Client, along with a "Join Now" button on the right. The main section is titled "CLIENT'S REGISTER" and features a form that requires the user to input their details. The form includes fields for Name, Email ID, Age, Password, and Mobile Number. In the screenshot, the Email ID is pre-filled

**CHAPTER**

**RESULTS AND DISCUSSION**

**1**. **system overview**

The system enables multiple tenants (users or organizations) to securely store encrypted data in a shared cloud infrastructure. A privacy-preserving and trusted keyword search mechanism is used, where:

Tenants upload encrypted documents.Keywords are extracted, encrypted, and indexed.

Search is performed without revealing the keyword, document content, or user identity to the cloud service provider.

**2. Evaluation Metrics**

To following metrics were uses

assess the efficiency, security, and usability of the system, the

Search Time (Latency)

Precision and Recall

Index Size

Encryption/Decryption Overhead

**3. Experimental Setup**

Environment: Java-based implementation using a simulated cloud environment.

Dataset: Used a synthetic dataset of documents with predefined keywords.

Tenants: Simulated multi-user environment with 10–100 users.

Hardware: 8 GB RAM, Intel i5, local test server.

**4. Performance Results**

A. Search Time Analysis |

Observation:

Search time grows linearly with the number of encrypted documents due to the use of optimized keyword index trees (like Inverted Bloom Filters or T-Tree based indexes)

D. Encryption/Decryption Overhead

Encryption time (per document): ~0.8 sec

Decryption time (per document): ~0.

Acceptable performance for small to medium workloads. For large files, hybrid encryption (symmetric for data, asymmetric for keys) is used to reduce load**.**

5. Security Evaluation

A. Resistance to Keyword Guessing Attacks: Keywords are encrypted using deterministic + random padding, preventing frequency analysis.

B. Access Pattern Hiding: Randomized search tokens and dummy queries help hide true access patterns.

C. Role of Trusted Authority (TA): Verifies the integrity of keyword tokens and users before allowing searches. Maintains key management and access control securely

6. Discussion

Scalability:

The architecture scales well up to thousands of documents and tens of users. For real-world deployment, distributed indexing and parallel search could be incorporated.

Efficiency:

Search is fast and accurate. The added privacy mechanisms do not significantly degrade performance.

Security:

The system maintains confidentiality, integrity, and query privacy effectively. It’s suitable for applications in finance, healthcare, legal, and multi-client SaaS platforms.

Limitations: Not suitable for extremely large-scale, real-time search.

Initial key management setup requires a trusted authority, which may be a bottleneck.

Revocation of user access is not instant; future versions should include dynamic re-keying.

**4.2 SCOPE FOR FUTURE WORK:**

**1.Support for Dynamic Data Operations**

Current implementation focuses on static datasets. In future:

Efficient insert, delete, and update operations on encrypted data and index should be enabled without rebuilding the index.Support for dynamic multi-user environments where tenants frequently join or leave.

**2. Search by Phrase and Fuzzy Matching**

Present system supports exact keyword matching.Future work can enable phrase-based searches and approximate keyword matching (e.g., typo tolerance), enhancing user-friendliness.

**3. Decentralized Key Management**

The current Trusted Authority (TA) model is centralized. Future work could explore blockchain-based decentralized key distribution and trust models, reducing single-point-of-failure risks**.**

**4.Integration with Real-World Cloud Services**

Deployment on platforms like AWS, Azure, or Google Cloud can test scalability in real-world environments.Use of cloud-native encryption APIs and serverless functions to manage secure indexing dynamically.

**CHAPTER 5**

**CONCLUSION**

5.1 **GENERAL**

In this project, a secure and efficient framework was proposed and implemented to enable privacy-preserving keyword search over multi-tenancy cloud environments. The system allows multiple users or organizations (tenants) to store and search encrypted data on a shared cloud platform while ensuring data confidentiality, user privacy, and trust through the involvement of a Trusted Authority (TA) and searchable encryption techniques.The approach effectively addresses key challenges such as data isolation among tenants, secure keyword indexing, and preventing unauthorized access

**5.2 FUTURE ENHANCEMENT**

Finally, we formally analyzed the security of our proposed schemes and conducted extensive experiments to show their effectiveness. For future work, we intend to enhance the security and performance of PVAKSE further.

**5.3 CONCLUSION:**

Herein, we propose a privacy-preserving, efficient, verifiable, accountable, and parallel solution for the keyword search problem in a multitenant cloud environment. To achieve this, we devised a privacy-preserving inverted index to enable a verifiable ciphertext search. Each entry contains encrypted keyword and document identity pairs and the compressed MAC for all corresponding documents. Then, we designed a fine-grained access control mechanism through keyword-based token generation. Moreover, we embedded the user identity into the token to achieve user accountability. All those components were built into the VAKSE scheme. To further improve search efficiency, we introduced the PVAKSE, in which the inverted index was partitioned into small segments that could be searched synchronously

**9.3 REFERENCES**

[1] (2022). Healthvana. [Online]. Available: https://healthvana.com

[2] (2022). CDPHP. [Online]. Available: https://www.cdphp.com

[3] (2022). Customer Success Stories. [Online]. Available: https://aws. amazon.com/solutions/case-studies/

[4] (2022). HiPAA. [Online]. Available: http://www.cms.hhs.gov/ HIPAAGenInfo

[5] D. Xiaoding Song, D. Wagner, and A. Perrig, “Practical techniques for searches on encrypted data,” in Proc. IEEE Symp. Secur. Privacy. (S&P), May 2000, pp. 44–55.

[6] E.-J. Goh, “Secure indexes,” Cryptol. ePrint Arch., Oct. 2003.

[7] R. Curtmola, J. 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.

[8] Q. Chai and G. Gong, “Verifiable symmetric searchable encryption for semi-honest-but-curious cloud servers,” in Proc. IEEE Int. Conf. Commun. (ICC), Jun. 2012, pp. 917–922.

[9] C. Chen et al., “An efficient privacy-preserving ranked keyword search method,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 4, pp. 951–963, Apr. 2016.

[10] F. Han, J. Qin, H. Zhao, and J. Hu, “A general transformation from KP ABE to searchable encryption,” Future Gener. Comput. Syst., vol. 30, pp. 107–115, Jan. 2014.

[11] V. Goyal, O. Pandey, A. Sahai, and B. Waters, “Attribute-based encryp tion for fine-grained access control of encrypted data,” in Proc. 13th ACM Conf. Comput. Commun. Secur., Oct. 2006, pp. 89–98.

[12] R. Brinkman, L. Feng, J. Doumen, P. H. Hartel, and W. Jonker, “Efficient tree search in encrypted data,” Inf. Syst. Secur., vol. 13, no. 3, pp. 14–21, May 2004.

[13] R. A. Popa, C. M. S. Redfield, N. Zeldovich, and H. Balakrishnan, “CryptDB: Protecting confidentiality with encrypted query process ing,” in Proc. 23rd ACM Symp. Oper. Syst. Principles (SOSP), 2011, pp. 85–100.

[14] J. Wang and S. S. Chow, “Omnes pro uno: Practical multi-writer encrypted database,” in Proc. 31st USENIX Secur. Symp. (USENIX Security), 2022, pp. 2371–2388.

[15] J. Li, K. Ren, B. Zhu, and Z. Wan, “Privacy-aware attribute-based encryption with user accountability,” in Proc. Int. Conf. Inf. Secur. Berlin, Germany: Springer, 2009, pp. 347–362. [16] J. Li, X. Chen, S. S. M. Chow, Q. Huang, D. S. Wong, and Z. Liu, “Multi-authority fine-grained access control with accountability and its application in cloud,” J. Netw. Comput. Appl., vol. 112, pp. 89–96, Jun. 2018.

[17] A. Soleimanian and S. Khazaei, “Publicly verifiable searchable sym metric encryption based on efficient cryptographic components,” Des., Codes Cryptogr., vol. 87, no. 1, pp. 123–147, 2019.

[18] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, “Privacy preserving multi-keyword ranked search over encrypted cloud data,” IEEE Trans. Parallel Distrib. Syst., vol. 25, no. 1, pp. 222–233, Jan. 2013.

[19] J. Wang et al., “Efficient verifiable fuzzy keyword search over encrypted data in cloud computing,” Comput. Sci. Inf. Syst., vol. 10, no. 2, pp. 667–684, 2013.

[20] W. Sun et al., “Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking,” in Proc. 8th ACM SIGSAC Symp. Inf., Comput. Commun. Secur., 2013, pp. 71–82.

[21] Z. Fu, L. Xia, X. Sun, A. X. Liu, and G. Xie, “Semantic-aware searching over encrypted data for cloud computing,” IEEE Trans. Inf. Forensics Security, vol. 13, no. 9, pp. 2359–2371, Sep. 2018.

[22] S. Kamara and C. Papamanthou, “Parallel and dynamic searchable symmetric encryption,” in Proc. Int. Conf. Financial Cryptography Data Secur. Berlin, Germany: Springer, 2013, pp. 258–274.

[23] W. Sun, X. Liu, W. Lou, Y. T. Hou, and H. Li, “Catch you if you lie to me: Efficient verifiable conjunctive keyword search over large dynamic encrypted cloud data,” in Proc. IEEE Conf. Comput. Commun. (INFOCOM), Apr. 2015, pp. 2110–2118..

[24] R. Bost, P.-A. Fouque, and D. Pointcheval, “Verifiable dynamic sym metric searchable encryption: Optimality and forward security,” Cryptol. ePrint Arch., Jan. 2016.

[25] K. Kurosawa and Y. Ohtaki, “UC-secure searchable symmetric encryp tion,” in Proc. Int. Conf. Financial Cryptogr. Data Secur. Berlin, Germany: Springer, 2012, pp. 285–298.

[26] J. Zhu, Q. Li, C. Wang, X. Yuan, Q. Wang, and K. Ren, “Enabling generic, verifiable, and secure data search in cloud services,” IEEE Trans. Parallel Distrib. Syst., vol. 29, no. 8, pp. 1721–1735, Aug. 2018.

[27] M. Hinek, S. Jiang, R. Safavi-Naini, and S. Shahandashti, “Attribute based encryption with key cloning protection,” Tech. Rep. 2008/478, 2008.

[28] S. Yu, K. Ren, W. Lou, and J. Li, “Defending against key abuse attacks in KP-ABE enabled broadcast systems,” in Proc. Int. Conf. Secur. Privacy Commun. Syst. Berlin, Germany: Springer, 2009, pp. 311–329.

[29] Q. Zheng, S. Xu, and G. Ateniese, “VABKS: Verifiable attribute-based keyword search over outsourced encrypted data,” in Proc. IEEE Conf. Comput. Commun., Apr. 2014, pp. 522–530.

[30] D. Boneh, G. D. Crescenzo, R. Ostrovsky, and G. Persiano, “Public key encryption with keyword search,” in Proc. Int. Conf. Theory Appl. Cryptograph. Techn. Berlin, Germany: Springer, 2004, pp. 506–522.

**PUBLICATION**

