A

Project Report

on

ENHANCED SECURITY FOR CLOUD-BASED KEYWORD SEARCH ON ENCRYPTED DATA

A report submitted in partial fulfillment of the requirements for the award of the Bachelor Of Technology degree

By

D VIPUL AVINASH

(20EG105311)

R DIVAKAR REDDY

(20EG105344)

M SRIVALLI REDDY

(20EG105702)

Under the guidance of

Dr. J. BALARAJU M.Tech., Ph.D
Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
ANURAG UNIVERSITY
VENKATAPUR– 500088
TELANGANA

YEAR 2023-24

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the report entitled **"ENHANCED SECURITY FOR CLOUD-BASED KEYWORD SEARCH ON ENCRYPTED DATA"** that is being submitted by D Vipul Avinash (20EG105311), R Divakar Reddy (20EG105344) and M Srivalli (20EG105702) in partial fulfillment for the award of B.Tech in Computer Science and Engineering to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision.

The results embodied in this report have not been submitted to any other University or Institute for the award of any degree or diploma

Signature of Supervisor

Dr. J Balaraju
M.Tech., Ph.D
Assistant Professor

Signature of Dean

Dr. G Vishnu Murthy
M.Tech., Ph.D
Professor, CSE

External Examiner

DECLARATION

We hereby declare that the report entitled "**ENHANCED SECURITY FOR CLOUD-BASED KEYWORD SEARCH ON ENCRYPTED DATA**" submitted for the award of Bachelor of technology Degree is our original work and the report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: Anurag University, Hyderabad

D Vipul Avinash (20EG105311)

R Divakar Reddy (20EG105344)

M Srivalli Reddy (20EG105702)

Date:

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. J Balaraju** for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B.Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy** , Dean, Dept. of CSE ,Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram** , Academic coordinator. **Dr. T Shyam Prasad** Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

D Vipul Avinash

(20EG105311)

R Divakar Reddy

(20EG105344)

M Srivalli Reddy

(20EG105702)

ABSTRACT

In pursuit of enhanced data availability and accessibility with a commitment to data secrecy, users increasingly opt to encrypt and outsource their data to cloud servers. However, conducting secure keyword searches on encrypted content poses a significant challenge. This paper introduces an innovative authorized keyword search scheme based on ciphertext-policy attribute-based encryption. The proposed scheme stands out in several aspects. Firstly, it supports a versatile multi-owner and multi-user scenario, accommodating encrypted data from various owners and making it searchable by multiple users. Secondly, through formal security analysis, the scheme is proven semantically secure against chosen keyword and outsiders keyword guessing attacks. Thirdly, an interactive protocol is presented, eliminating the need for secure channels between users and service providers. Fourthly, leveraging bilinear-map accumulators enables efficient user and attribute revocation, ensuring authentication before resource-intensive search operations. Fifthly, the scheme facilitates conjunctive keyword search, allowing the simultaneous search for multiple keywords at minimal cost. Lastly, performance analysis demonstrates the superior efficiency of the proposed scheme compared to closely-related works. This research offers a comprehensive solution to the secure and efficient exploration of encrypted keyword searches in cloud-based environments

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF GRAPHS

1.INTRODUCTION

➢ Cloud computing

Cloud computing is the use of computing resources (hardware and software) that are delivered as a service over a network (typically the Internet). The name comes from the common use of a cloud-shaped symbol as an abstraction for the complex infrastructure it contains in system diagrams. Cloud computing entrusts remote services with a user's data, software and computation. Cloud computing consists of hardware and software resources made available on the Internet as managed third-party services. These services typically provide access to advanced software applications and high-end networks of server computers.

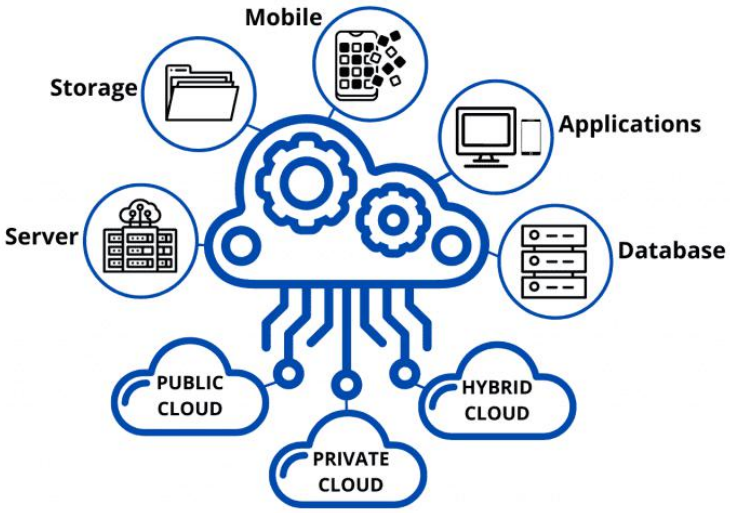


Fig 1.1: Structure of cloud computing

➢ Understanding working principle of Cloud Computing

The goal of cloud computing is to apply traditional supercomputing, or high-performance computing power, normally used by military and research facilities, to perform tens of trillions of computations per second, in consumer-oriented applications such as financial portfolios, to deliver personalized information, to provide data storage or to power large, immersive computer games.

The cloud computing uses networks of large groups of servers typically running low-cost consumer PC technology with specialized connections to spread data-processing chores across them. This shared IT infrastructure contains large pools of systems that are linked together. Often, virtualization techniques are used to maximize the power of cloud computing.

- **Characteristics and Services Models:**

The salient characteristics of cloud computing based on the definitions provided by the National Institute of Standards and Terminology (NIST) are outlined below:

- **On-demand self-service:** A consumer can unilaterally provision computing capabilities, such as server time and network storage, as needed automatically without requiring human interaction with each service's provider.
- **Broad network access:** Capabilities are available over the network and accessed through standard mechanisms that promote use by heterogeneous thin or thick client platforms (e.g., mobile phones, laptops, and PDAs).
- **Resource pooling:** The provider's computing resources are pooled to serve multiple consumers using a multi-tenant model, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. There is a sense of location-independence in that the customer generally has no control or knowledge over the exact location of the provided resources but may be able to specify location at a higher level of abstraction (e.g., country, state, or data center). Examples of resources include storage, processing, memory, network bandwidth, and virtual machines.
- **Rapid elasticity:** Capabilities can be rapidly and elastically provisioned, in some cases automatically, to quickly scale out and rapidly released to quickly scale in. To the consumer, the capabilities available for provisioning often appear to be unlimited and can be purchased in any quantity at any time.
- **Measured service:** Cloud systems automatically control and optimize resource use by leveraging a metering capability at some level of abstraction appropriate to

the type of service (e.g., storage, processing, bandwidth, and active user accounts). Resource usage can be managed, controlled, and reported providing transparency for both the provider and consumer of the utilized service.

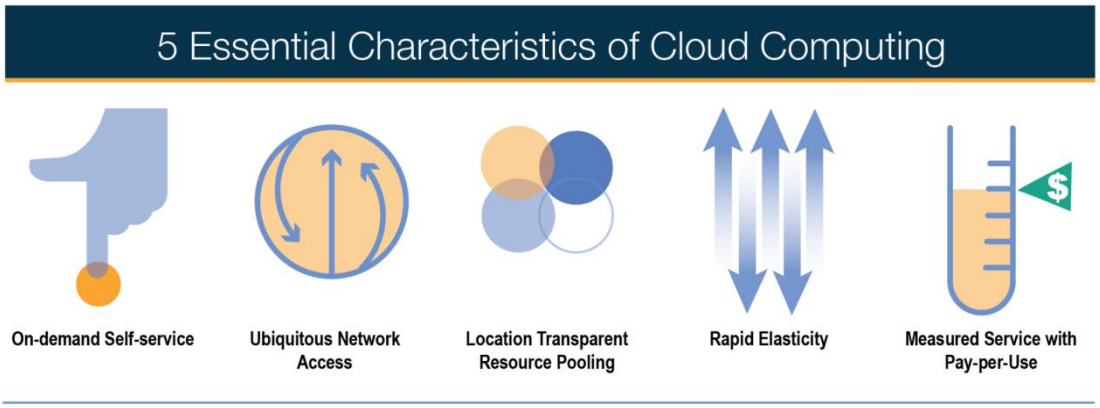


Fig 1.2: Characteristics of cloud computing

- **Services Models:**

Cloud Computing comprises three different service models, namely Infrastructure-as-a-Service (IaaS), Platform-as-a-Service (PaaS), and Software-as-a-Service (SaaS). The three service models or layer are completed by an end user layer that encapsulates the end user perspective on cloud services. The model is shown in figure below. If a cloud user accesses services on the infrastructure layer, for instance, she can run her own applications on the resources of a cloud infrastructure and remain responsible for the support, maintenance, and security of these applications herself. If she accesses a service on the application layer, these tasks are normally taken care of by the cloud service provider.

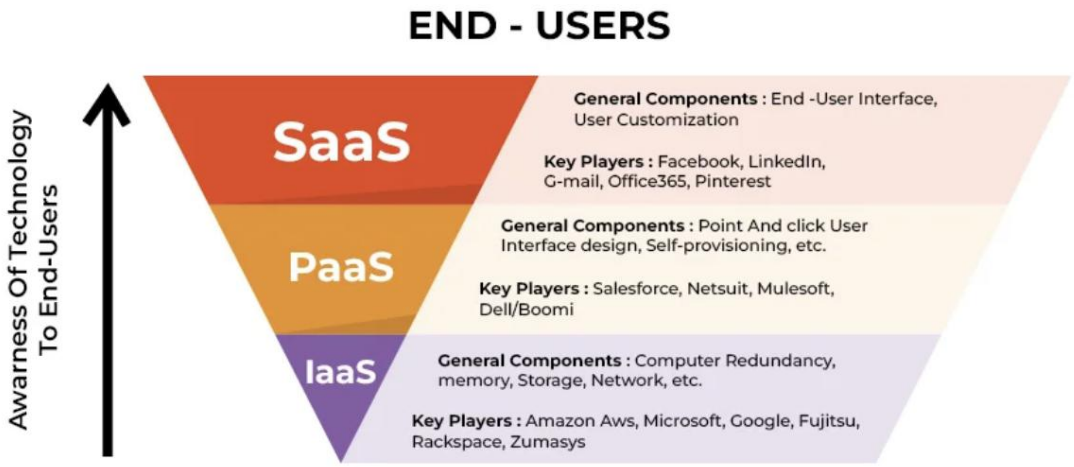


Fig 1.3: Structure of service models

- **Benefits of cloud computing:**

- **Achieve economies of scale** – increase volume output or productivity with fewer people. Your cost per unit, project or product plummets.
- **Reduce spending on technology infrastructure.** Maintain easy access to your information with minimal upfront spending. Pay as you go (weekly, quarterly or yearly), based on demand.
- **Globalize your workforce on the cheap.** People worldwide can access the cloud, provided they have an Internet connection.
- **Streamline processes.** Get more work done in less time with less people.
- Reduce capital costs. There's no need to spend big money on hardware, software or licensing fees.
- **Improve accessibility.** You have access anytime, anywhere, making your life so much easier!
- **Monitor projects more effectively.** Stay within budget and ahead of completion cycle times.
- **Less personnel training is needed.** It takes fewer people to do more work on a cloud, with a minimal learning curve on hardware and software issues.

- **Minimize licensing new software.** Stretch and grow without the need to buy expensive software licenses or programs.
- **Improve flexibility.** You can change direction without serious "people" or "financial" issues at stake.

➢ **Advantages:**

- **Price:** Pay for only the resources used.
- **Security**: Cloud instances are isolated in the network from other instances for improved security.
- **Performance:** Instances can be added instantly for improved performance. Clients have access to the total resources of the Cloud's core hardware.
- **Scalability:** Auto-deploy cloud instances when needed.
- **Uptime:** Uses multiple servers for maximum redundancies. In case of server failure, instances can be automatically created on another server.
- **Control:** Able to login from any location. Server snapshot and a software library lets you deploy custom instances.
- **Traffic:** Deals with spike in traffic with quick deployment of additional instances to handle the load.

2. LITERATURE SURVEY

- **Data security and privacy preservation in cloud storage environments based on cryptographic mechanisms**

Authors: N. Kaaniche and M. Laurent

Recent technological advances have sparked the popularity and success of cloud. This new paradigm is gaining an expanding interest, since it provides cost efficient architectures that support the transmission, storage, and intensive computing of data. However, these promising storage services bring many challenging design issues, considerably due to both loss of data control and abstract nature of clouds. The objective of this survey is to provide a consistent view about both data security concerns and privacy issues that are faced by clients in cloud storage environments. This survey brings a critical comparative analysis of cryptographic defense mechanisms, and beyond this, it explores research directions and technology trends to address the protection of outsourced data in cloud infrastructures.

- **Ensuring security and privacy preservation for cloud data services**

Authors: J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya

With the rapid development of cloud computing, more and more enterprises/individuals are starting to outsource local data to the cloud servers. However, under open networks and not fully trusted cloud environments, they face enormous security and privacy risks (e.g., data leakage or disclosure, data corruption or loss, and user privacy breach) when outsourcing their data to a public cloud or using their outsourced data. Recently, several studies were conducted to address these risks, and a series of solutions were proposed to enable data and privacy protection in untrusted cloud environments. To fully understand the advances and discover the research trends of this area, this survey summarizes and analyzes the state-of-the-art protection technologies. We first present security threats and requirements of an outsourcing data service to a cloud, and follow that with a high-level overview of the

corresponding security technologies. We then dwell on existing protection solutions to achieve secure, dependable, and privacy-assured cloud data services including data search, data computation, data sharing, data storage, and data access. Finally, we propose open challenges and potential research directions in each category of solutions.

- **Searchable symmetric encryption: Improved definitions and efficient constructions**

Authors: R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky

Searchable symmetric encryption (SSE) allows a party to outsource the storage of its data to another party (a server) in a private manner, while maintaining the ability to selectively search over it. This problem has been the focus of active research in recent years. In this paper we show two solutions to SSE that simultaneously enjoy the following properties:

Both solutions are more efficient than all previous constant-round schemes. In particular, the work performed by the server per returned document is constant as opposed to linear in the size of the data.

Both solutions enjoy stronger security guarantees than previous constant-round schemes. In fact, we point out subtle but serious problems with previous notions of security for SSE, and show how to design constructions which avoid these pitfalls. Further, our second solution also achieves what we call adaptive SSE security, where queries to the server can be chosen adaptively (by the adversary) during the execution of the search; this notion is both important in practice and has not been previously considered.

Surprisingly, despite being more secure and more efficient, our SSE schemes are remarkably simple. We consider the simplicity of both solutions as an important step towards the deployment of SSE technologies.As an additional contribution, we also consider multi-user SSE. All prior work on SSE studied the setting where only the owner of the data is capable of submitting search queries. We consider the natural extension where an arbitrary group of parties other than the owner can submit search

queries. We formally define SSE in the multi-user setting, and present an efficient construction that achieves better performance than simply using access control mechanisms.

- **Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking**

Authors: W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li

With the increasing popularity of cloud computing, huge amount of documents are outsourced to the cloud for reduced management cost and ease of access. Although encryption helps protecting user data confidentiality, it leaves the well-functioning yet practically-efficient secure search functions over encrypted data a challenging problem. In this paper, we present a privacy-preserving multi-keyword text search (MTS) scheme with similarity-based ranking to address this problem. To support multi-keyword search and search result ranking, we propose to build the search index based on term frequency and the vector space model with cosine similarity measure to achieve higher search result accuracy. To improve the search efficiency, we propose a tree-based index structure and various adaption methods for multi-dimensional (MD) algorithm so that the practical search efficiency is much better than that of linear search. To further enhance the search privacy, we propose two secure index schemes to meet the stringent privacy requirements under strong threat models, i.e., known ciphertext model and known background model. Finally, we demonstrate the effectiveness and efficiency of the proposed schemes through extensive experimental evaluation.

- **Privacy-preserving multi-keyword ranked search over encrypted cloud data**

Authors: N. Cao, C. Wang, M. Li, K. Ren, and W. Lou

With the advent of cloud computing, data owners are motivated to outsource their complex data management systems from local sites to the commercial public cloud for great flexibility and economic savings. But for protecting data privacy, sensitive data have to be encrypted before outsourcing, which obsoletes traditional data

utilization based on plaintext keyword search. Thus, enabling an encrypted cloud data search service is of paramount importance. Considering the large number of data users and documents in the cloud, it is necessary to allow multiple keywords in the search request and return documents in the order of their relevance to these keywords. Related works on searchable encryption focus on single keyword search or Boolean keyword search, and rarely sort the search results. In this paper, for the first time, we define and solve the challenging problem of privacy-preserving multi-keyword ranked search over encrypted data in cloud computing (MRSE). We establish a set of strict privacy requirements for such a secure cloud data utilization system. Among various multi-keyword semantics, we choose the efficient similarity measure of "coordinate matching," i.e., as many matches as possible, to capture the relevance of data documents to the search query. We further use "inner product similarity" to quantitatively evaluate such similarity measure. We first propose a basic idea for the MRSE based on secure inner product computation, and then give two significantly improved MRSE schemes to achieve various stringent privacy requirements in two different threat models. To improve search experience of the data search service, we further extend these two schemes to support more search semantics. Thorough analysis investigating privacy and efficiency guarantees of proposed schemes is given. Experiments on the real-world data set further show proposed schemes indeed introduce low overhead on computation and communication.

Sl.No	Strategies	Advantages	Disadvantages
1	Hierarchical Predicate Encryption	Selective security. Efficient-computation. Inner-product predicate.	Time-consuming. Expensive. Selective security.
2	Public key encryption with keyword search (PEKS)	Secure and Efficient. Multi-User Data Sharing	Vulnerable to keyword guessing attacks. File injection attacks.
3	SCF-PKEKS	Data privacy. Scalability. Accountability	Keyword limitations. Complexity. Compatibility

Table 2.1 Comparison of Existing Strategies for Problem solve

Sl.No	Author	Strategies	Advantages	Disadvantages
1	P. Xu, S. Tang, P. Xu, Q.Wu,H. Hu, and W. Susilo	Public-key multi-keyword searchable encryption with hidden structures (PMSEHS)	Improved search functionality. Enhanced privacy. Flexible access control.	Potential for false positives. Limited search functionality. Dependence on the cloud server.
2	J. Baek, R. Safavi-Naini, and W. Susilo	Public key encryption with keyword search (PEKS)	Secure and Efficient. Multi-User Data Sharing	Vulnerable to keyword guessing attacks. File injection attacks.
3	W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li	SCF-PKEKS	Data privacy. Scalability. Accountability	Keyword limitations. Complexity. Compatibility

Table 2.2 Comparison of Existing Method from selected Strategies

3. PROPOSED METHOD

3.1 Proposed System

This project proposes a secure and efficient mechanism for authorized keyword search over encrypted data outsourced to a cloud environment. In this paper, we further investigate the issues of keyword search mechanisms and propose a secure and efficient scheme for authorized keyword search over encrypted data for multi-owner and multi-user cloud environment.

The proposed system uses an expressive fine-grained authorized keyword search scheme for designing for multi-owner and multiuser scenarios using the concept of CP-ABE. The system also develops an interactive protocol is constructed between the user and CSP to avoid overhead involved in establishing secure-channel.
In the proposed system the Owner module allows the data owner to upload encrypted files to the cloud server and view their file details and encrypted index. The User module enables authorized users to search for encrypted data based on their granted privileges, with a request sent to the cloud server for verification by the Verification Authority. The Attribute Authority is responsible for approving new user registrations, and the Verification Authority verifies search access rights of users. The CSP stores and retrieves encrypted data and performs keyword search based on valid search access rights. The system is implemented using the DriveHQ cloud service provider for storage. The proposed mechanism ensures strong security guarantees and high search efficiency, making it suitable for data owners looking to securely outsource their data to cloud servers while retaining the ability to search for specific keywords within their encrypted data.

i. **Advantages Of Proposed System:**

- **Fine-Grained Authorized Keyword Search:** The proposed system offers an expressive fine-grained authorized keyword search scheme, which allows multiple owners and users to perform keyword-based searches on encrypted data using CP-ABE. This enhances the flexibility and usability of the system.

- **Efficient Protocol:** The proposed system utilizes an interactive protocol between the user and CSP that helps to avoid overhead involved in establishing a secure channel. This ensures efficient and secure data transfer.

- **Essential Functionalities:** The proposed system includes essential functionalities like allow only the authorized users to perform keyword search, which are added without any significant overhead. This enhances the system's capabilities and improves its usability.
- **Security:** The proposed system is proven to be semantically secure against chosen keyword attack and outsider's keyword guessing attack under standard assumptions. This ensures the confidentiality and integrity of the data.
- **Efficient Computation:** The implementation of the proposed scheme shows that it takes substantially less computation time to perform keyword search, making it suitable for real-life cloud environments compared with closely related works. This improves the efficiency and performance of the system, making it a more viable option for practical use.
- **Keyword secrecy:** without qualified trapdoors any outsider and the CSP should not be able to learn any useful information about the plaintext keywords from the encrypted indexes. Similarly, an outsider should not be able to learn any useful information about the plaintext keywords from the trapdoors. These two security notions can be captured by Keyword Semantic Security, which implies that encrypted indexes do not reveal any useful information about the plaintext keywords. This security notion is referred to as selective in distinguish-ability against chosen keyword attack under selective ciphertext policy model.

3.1.2 System Specifications

- **Hardware Requirements:**

System	: Pentium i3 Processor
Hard Disk	: 500 GB.
Monitor	: 15'' LED
Input Devices	: Keyboard, Mouse
Ram	: 4 GB

- **Software Requirements:**

Operating system	: Windows 10.
Coding Language	: JAVA.
Tool	: Apache Netbeans IDE 16
Database	: MYSQL

3.1.3 System Architecture

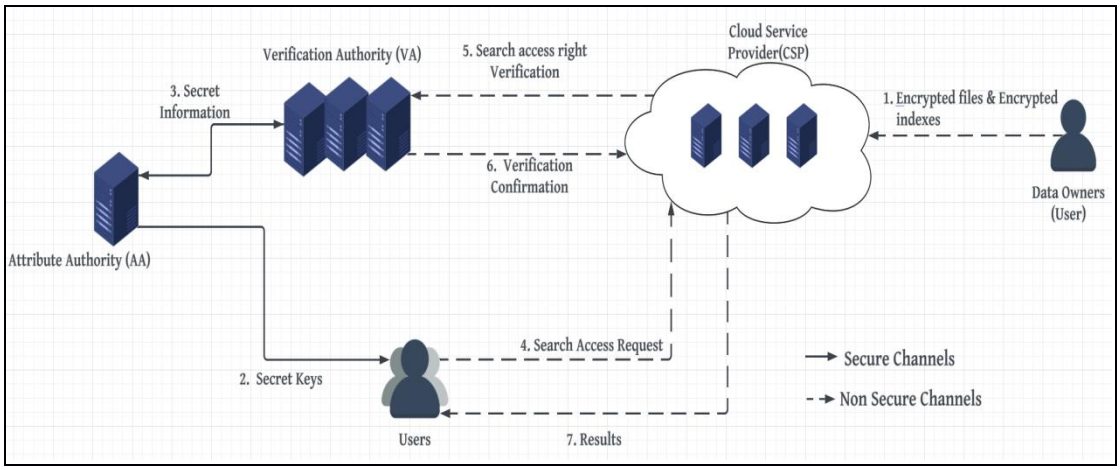


Fig 3.1.3: System Architecture

3.2 System Methodology

3.2.1 System Analysis

i. Existing System:

- Li et al. proposed an PKAKS scheme using the concept of Hierarchical Predicate Encryption for multi-owner and multi-user setting, where only authorized users can perform keyword search over the encrypted indexes. However, it incurs high communication overhead, as a user needs to contact the attribute authority to generate trapdoors for each search request. Also, the scheme is suitable to perform keyword search over structured database which contains limited keywords
- Baek et al. highlight that Boneh et al.'s scheme needs secure-channel between a user and the CSP to transmit trapdoors, which may incur high overhead in terms of communication and processing costs.
- Jiang et al. proposed an SCF-PKEKS scheme where the users can perform keyword search over the authorized encrypted data only. However, none of the SCF-PKEKS schemes provide fine-grained search authorization over encrypted data.

ii. Disadvantages Of Existing System:

The existing system scheme incurs high communication overhead as each search request requires the user to contact the attribute authority to generate trapdoors, which can be time-consuming and expensive.

The existing system scheme requires secure-channel between users and the CSP to transmit trapdoors, which can incur high overhead in terms of communication and processing costs.

The existing system scheme lacks fine-grained search authorization over encrypted data, which can restrict the user's ability to search for specific keywords.

The existing system scheme requires secure-channels between users and the CSP to transmit trapdoors, which can be costly and time-consuming. Additionally, the scheme does not provide conjunctive keyword search and user revocation, which can limit its usefulness in real-world scenarios.

3.2.2 System Study

i. Feasibility Study

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are:

- Economical Feasibility
- Technical Feasibility
- Social Feasibility

- **Economical Feasibility:** This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

- **Technical Feasibility:** This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands

being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

- **Social Feasibility:** The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2.3 System Testing

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 test. Each test type addresses a specific testing requirement.

i. Types Of Tests

- **Unit testing:** Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs 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.

- **Integration testing:** Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed atexposing the problems that arise from the combination of components.
- **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.

  Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

- **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.

- **White Box Testing:** White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.
- **Black Box Testing:** Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot "see" into it. The test provides inputs and responds to outputs without considering how the software works.
- **Unit Testing:** Unit testing is usually conducted as part of a combined code and unit test phase of the software life cycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

ii. Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

All field entries must work properly.

Pages must be activated from the identified link.

The entry screen, messages and responses must not be delayed.

Features to be tested

Verify that the entries are of the correct format

No duplicate entries should be allowed

All links should take the user to the correct page.

- **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.

- **Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

- **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.
- **Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

3.3 Software Environment

i. The Java Programming Language

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes*

The platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

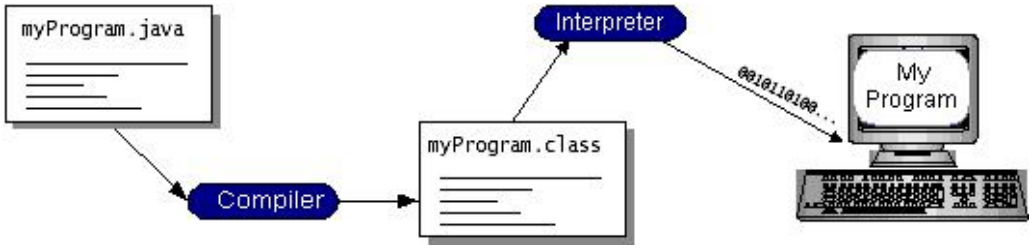


Fig 3.3.1: Java Technology

You can think of Java byte codes as the machine code instructions for the *Java Virtual Machine* (Java VM). Every Java interpreter, whether it's a development tool or a Web browser that can run applets, is an implementation of the Java VM. Java byte codes help make "write once, run anywhere" possible. You can compile your program into byte codes on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java VM. That means that as long as a computer has a Java VM, the same program written in the Java programming language can run on Windows 2000, a Solaris workstation, or on an iMac.

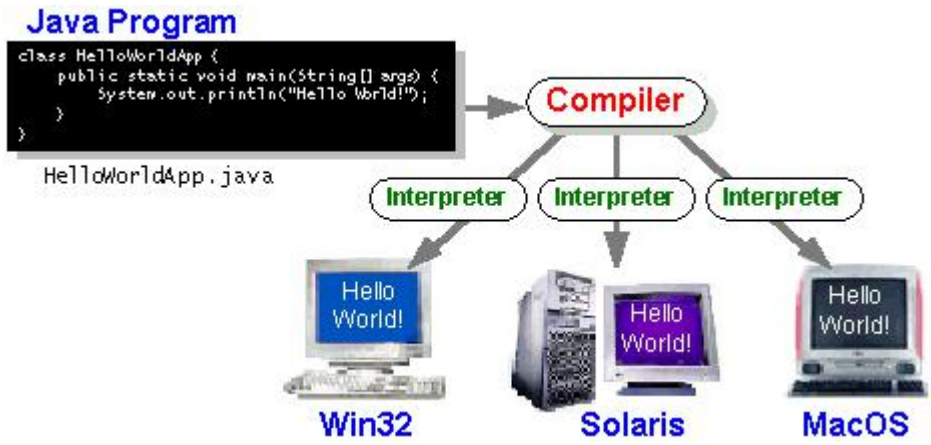


Fig 3.3.2: Java Programming Language

ii. The Java Platform

A *platform* is the hardware or software environment in which a program runs. We've already mentioned some of the most popular platforms like Windows 2000, Linux, Solaris, and MacOS. Most platforms can be described as a combination of the operating system and hardware. The Java platform differs from most other platforms in that it's a software-only platform that runs on top of other hardware-based platforms.

The Java platform has two components:

The *Java Virtual Machine* (Java VM)

The *Java Application Programming Interface* (Java API)

You've already been introduced to the Java VM. It's the base for the Java platform and is ported onto various hardware-based platforms.

The Java API is a large collection of ready-made software components that provide many useful capabilities, such as graphical user interface (GUI) widgets. The Java API is grouped into libraries of related classes and interfaces; these libraries are known as *packages*. The next section, What Can Java Technology Do? Highlights what functionality some of the packages in the Java API provide.

The following figure depicts a program that's running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

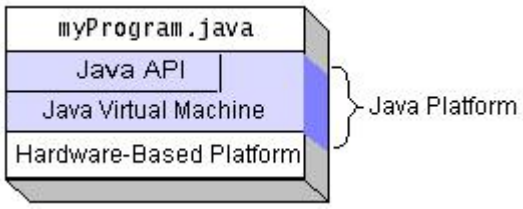


Fig 3.3.3: Program running on Java Platform

Native code is code that after you compile it, the compiled code runs on a specific hardware platform. As a platform-independent environment, the Java platform can be a bit slower than native code. However, smart compilers, well-tuned interpreters, and just-in-time byte code compilers can bring performance close to that of native code without threatening portability.

iii. ODBC

Microsoft Open Database Connectivity (ODBC) is a standard programming interface for application developers and database systems providers. Before ODBC became a *de facto* standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data

source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources.

From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

The advantages of this scheme are so numerous that you are probably thinking there must be some catch. The only disadvantage of ODBC is that it isn't as efficient as talking directly to the native database interface. ODBC has had many detractors make the charge that it is too slow. Microsoft has always claimed that the critical factor in performance is the quality of the driver software that is used. In our humble opinion, this is true. The availability of good ODBC drivers has improved a great deal recently. And anyway, the criticism about performance is somewhat analogous to those who said that compilers would never match the speed of pure assembly language. Maybe not, but the compiler (or ODBC) gives you the opportunity to write

cleaner programs, which means you finish sooner. Meanwhile, computers get faster every year.

iv. JDBC

In an effort to set an independent database standard API for Java; Sun Microsystems developed Java Database Connectivity, or JDBC. JDBC offers a generic SQL database access mechanism that provides a consistent interface to a variety of RDBMSs. This consistent interface is achieved through the use of "plug-in" database connectivity modules, or *drivers*. If a database vendor wishes to have JDBC support, he or she must provide the driver for each platform that the database and Java run on.

To gain a wider acceptance of JDBC, Sun based JDBC's framework on ODBC. As you discovered earlier in this chapter, ODBC has widespread support on a variety of platforms. Basing JDBC on ODBC will allow vendors to bring JDBC drivers to market much faster than developing a completely new connectivity solution.

JDBC was announced in March of 1996. It was released for a 90 day public review that ended June 8, 1996. Because of user input, the final JDBC v1.0 specification was released soon after.

The remainder of this section will cover enough information about JDBC for you to know what it is about and how to use it effectively. This is by no means a complete overview of JDBC. That would fill an entire book.

v. MySQL

- **Database**

A database is a separate application that stores a collection of data. Each database has one or more distinct APIs for creating, accessing, managing, searching and replicating the data it holds.

Other kinds of data stores can also be used, such as files on the file system or large hash tables in memory, but data fetching and writing would not be so fast and easy with those type of systems.

Nowadays, we use relational database management systems (RDBMS) to store and manage huge volume of data. This is called relational database because all the data is stored into different tables and relations are established using primary keys or other keys known as Foreign Keys.

A Relational DataBase Management System (RDBMS) is a software that:

Enables you to implement a database with tables, columns and indexes.
Guarantees the Referential Integrity between rows of various tables.
Updates the indexes automatically.
Interprets an SQL Query and combines information from various tables.

- **MySQL Database**

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons:

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including HP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.
- MySQL is very friendly to PHP, the most appreciated language for web development.

- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customization. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

4. IMPLEMENTATION

4.1 Modules

- Owner
- Users.
- Attribute Authority
- Verification Authority
- Cloud Service Provider

4.2 Modules Description

- **Owner:** In the first module we develop the Owner part. Owner outsources encrypted data to the cloud storage servers. A new owner is first registered and then login into the system. The Owner has the option of File upload, My files and Encrypted Index details. In the file upload part the owner can able to upload the file by providing the Keyword and the file. Once uploaded the file, the owner can view the details in the My files part and in the Encrypted Index part.

- **User:** In the next module we develop the User Module. Users are requesting entities that are authorized to search and access encrypted data based on their granted privileges. The new user is registered first and then user can able to login into the system only if the Attribute Authority approves the new user. If the Attribute Authority doesn't approve the user then the user cannot able to login into the system. Once after the approval of the Attribute Authority, the user can able to login into the system. The user has the option of Get Search Access, Verified Results and All Search Request details. First in the get search access, the user can type the keyword for the file which is required. Once done it is sent to the Cloud Service provider. In the cloud service provider with the help of Verification Authority the search access request is processed. Only after the

approval of both the Cloud service provider and Verification Authority, the search request will be approved to the user.

- **Attribute Authority:** In this module, we develop the Attribute Authority. AA is an independent trusted third-party entity which maintains public parameter and master secret in the system. It also issues access rights in the form of secret keys to the users based on the attributes they possess. Attribute Authority is the entity which approves the new users. Only if the Attribute Authority approves the new user, then only the new user can able to login into the system. If the AA rejects the new user, then the new user cannot able to login into the system. Attribute Authority has the option of User Activations and Active Users details.

- **Verification Authority:** In this module we develop the Verification Authority. VA is also another third-party entity which keeps secret information received from the AA. It is to be noted that VA can also be a set of designated servers under the control of the Cloud Service Provider. VA mainly verifies search access rights of the users upon Cloud Service Provider's request relying on the shared secret information with AA. The Verification authority has the option of viewing the Users, Search Access Request and All access requests. In the Search Access Request, the request sent by the user is shown and the verification of the request is done here in this part. Only if the Verification Authority verifies and give the access, then only the user can able to search and get the required file in the cloud.

- **CSP:** In this module, we develop the Cloud Service Provider. CSP is a third-party entity which maintains the cloud storage servers and performs keyword search over the encrypted data based on a valid search access right of a user. CSP has the options of Access Requests, Verified Request, All Access Request and Downloads details. In Access Requests part, the request sent by the User is shown, which is sent to the Verification Authority from here. Once if the Verification Authority verifies and gives back the verified results, and then in the Verified Request it is being updated. Then only the cloud sends the results to the user. In

the Downloads part the details of file which is downloaded by the user is updated. This module is responsible for the storage and retrieval of encrypted data and traceability mechanisms. It includes functionalities such as the storage and retrieval of encrypted data, the verification of access and traceability records. We have used DriveHQ cloud service provider for storing our files in the cloud. The files uploaded by the owner will be saved in the DriveHQ cloud.

4.3 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

➢ **Objectives**

i. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

ii. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

iii. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

4.4 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

i. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

ii. Select methods for presenting information.

iii. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

4.5 Source Code

- **SQL Connection**

```
package AuthorizedSearch;
import java.sql.Connection;
import java.sql.DriverManager;
public class SQLconnection {
static Connection con;
public static Connection getconnection() {
try {
Class.forName("com.mysql.jdbc.Driver");
con = DriverManager.getConnection("jdbc:mysql://localhost:3306/authkeysearch",
"root", "major");
} catch (Exception e) {
}
return con;
}
}
```

- **Access Verification**

```
package AuthorizedSearch;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
```

```
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class AccessVerification extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
String id = request.getParameter("id");

Connection con = null;
Statement st = null;
Statement st1 = null;
Connection conn = SQLconnection.getconnection();
Statement sto = conn.createStatement();
st = conn.createStatement();

try {
int i = sto.executeUpdate("update searchreq set vstatus='Pending' where id='" + id + "'");
System.out.println("test print==" + id);
if (i != 0)
{
response.sendRedirect("CSearchAccReq.jsp?SentToVA");
}
else
{
System.out.println("failed");
response.sendRedirect("CSearchAccReq.jsp?Failed");
}
} catch (Exception ex) {
ex.printStackTrace();
}
} catch (SQLException ex) {
Logger.getLogger(AccessVerification.class.getName()).log(Level.SEVERE, null, ex);
}
}
```

```
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}

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

public String getServletInfo() {
return "Short description";
}// </editor-fold>

}
```

- **Get Search Access**

```
package AuthorizedSearch;
import AuthorizedSearch.PKE.KEYGEN;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.text.DateFormat;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class GetSearchAccess extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
```

```
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {

String keyw = request.getParameter("trapdoor");
KEYGEN k = new KEYGEN();
String keyword = k.encrypt(keyw.toLowerCase());
HttpSession user = request.getSession(true);
String uid = user.getAttribute("uid").toString();
String uname = user.getAttribute("uname").toString();
String umail = user.getAttribute("umail").toString();
System.out.println(" \n keyword :" + keyword + "\n uid:" + uid);
Connection con = SQLconnection.getconnection();
Statement st = con.createStatement();
Statement st1 = con.createStatement();
try {
DateFormat dateFormat = new SimpleDateFormat("yyyy/MM/dd HH:mm:ss");
Date date = new Date();
String time = dateFormat.format(date);
System.out.println("Date and Time : " + time);
int i = st1.executeUpdate("INSERT into searchreq(uid, uname, keyword, vstatus,
time,email) values('" + uid + "','" + uname + "','" + keyword + "','waiting','" + time +
"','"+umail+"')");
if (i != 0) {
response.sendRedirect("GetSearchAcc.jsp?requestsent");
} else {
response.sendRedirect("GetSearchAcc.jsp?failed");
}
} catch (Exception ex) {
ex.printStackTrace();
}
} catch (SQLException ex) {
Logger.getLogger(GetSearchAccess.class.getName()).log(Level.SEVERE, null, ex);
} catch (Exception ex) {
Logger.getLogger(GetSearchAccess.class.getName()).log(Level.SEVERE, null, ex);
}
}

protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
```

```
protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}

public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

- **User Rejection**

```
package AuthorizedSearch;
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.logging.Level;
import java.util.logging.Logger;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class UserReject extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse
response)
throws ServletException, IOException {
response.setContentType("text/html;charset=UTF-8");
try (PrintWriter out = response.getWriter()) {
String id = request.getParameter("uid");

Connection con = null;
Statement st = null;
Statement st1 = null;
Connection conn = SQLconnection.getconnection();
Statement sto = conn.createStatement();
st = conn.createStatement();

try {
```

```
int i = sto.executeUpdate("update users set ustatus='Rejected' where id='" + id + "'");
System.out.println("test print==" + id);
if (i != 0)
{
response.sendRedirect("UserAct.jsp?Rejected");
}
else
{
System.out.println("failed");
response.sendRedirect("UserAct.jsp?Failed");
}
} catch (Exception ex) {
ex.printStackTrace();
}
} catch (SQLException ex) {
Logger.getLogger(UserReject.class.getName()).log(Level.SEVERE, null, ex);
}
}

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

protected void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
processRequest(request, response);
}
public String getServletInfo() {
return "Short description";
}// </editor-fold>
}
```

- **FTP Connection**

```
package AuthorizedSearch;
import java.io.File;
import java.io.FileInputStream;
import org.apache.commons.net.ftp.FTPClient;

public class FTPcon {
```

```
FTPClient client = new FTPClient();
FileInputStream fis = null;
boolean status;
public boolean upload(File file) {
try {
System.out.println("Check----------------------------------->1");
client.connect("ftp.drivehq.com");
client.login("Vipulavi", "au123456");
client.enterLocalPassiveMode();
fis = new FileInputStream(file);
System.out.println("Check----------------------------------->2");
status = client.storeFile("/" + file.getName(), fis);
//file path of drive storage
client.logout();
fis.close();
} catch (Exception e) {
System.out.println(e);
}
if (status) {
System.out.println("success");
return true;
} else {
System.out.println("failed");
return false;
}
}
}
```

5. OBSERVATIONS

5.1 Parameter Formulas

A CP-ABE-SD scheme possesses the four algorithms: Setup, KeyGen, Encrypt and Decrypt.

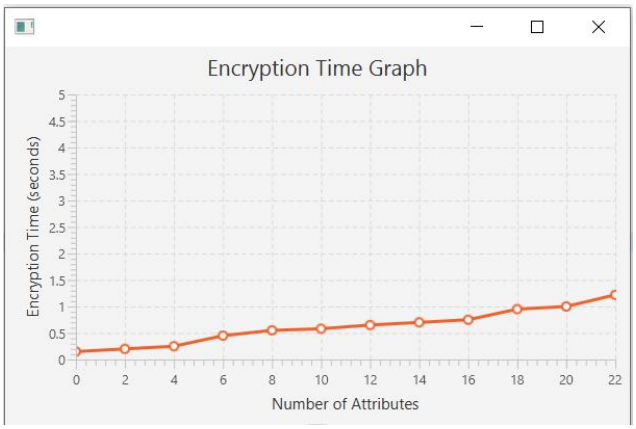
- **Setup** $(1^K) \rightarrow (MK, PK)$. This algorithm is executed by TA. A security parameter 1^K is inputted, it returns the public key PK and the master secret key MK of the system.
- **Encrypt** $(PK, ck, A) \rightarrow$ CT. DO performs the encryption algorithm, which inputs a session key *ck* , *PK* and an access structure and returns the ciphertext CT .
- **KeyGen** $(S, PK, MK) \rightarrow$ SK. The operation is performed by TA. The algorithm inputs attribute set *S, PK* and *MK* and outputs the secret key *SK* related to the attribute set S
- **Decrypt** $(CT, PK, SK) \rightarrow ck$. The algorithm *inputs CT , SK and PK* , it outputs session key *ck* or $\perp$. This operation is run by DU.

Parameter	Previous methods	Proposed method
Fine-Grained Authorized Keyword Search	lacks fine-grained search authorization over encrypted data, which can restrict the user's ability to search for specific keywords.	The proposed system offers an expressive fine-grained authorized keyword search Enhances the flexibility and usability of the system.
Security Configurations	The existing system scheme requires secure-channels between users and the CSP to transmit trapdoors, which can be costly and time-consuming	The proposed system is proven to be semantically secure against chosen keyword attack and outsider's keyword guessing attack under standard assumptions.

Table 5.1.1 Parameter comparison table

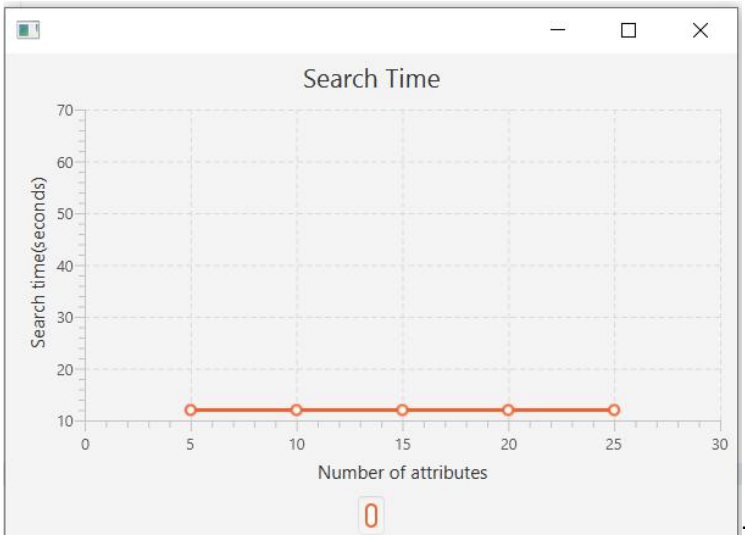
6. DISCUSSION OF RESULTS

Experiment 1: Here we can find that as the number of attributes increases there is a slight increase in encryption time.



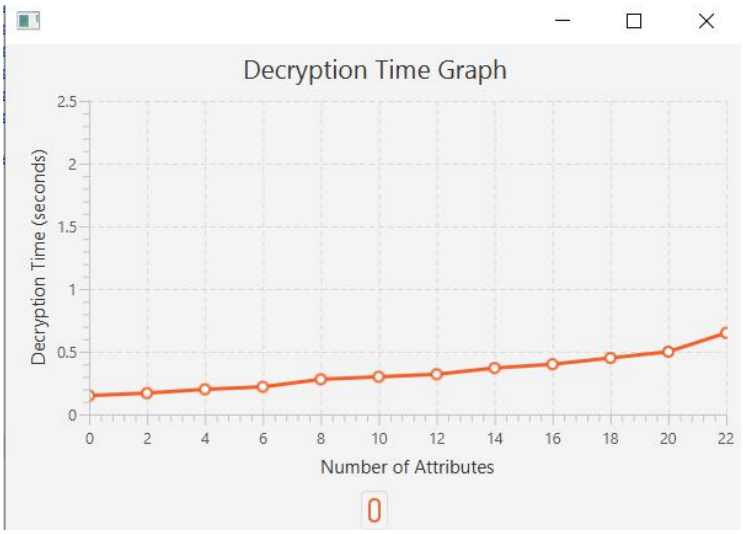
Graph 6.1: Encryption Time vs Number of Attributes

Experiment 2: The graph demonstrates us that the search time remains constant as the number of attributes increases.



Graph 6.2: Search Time Vs Number of Attributes

Experiment 3: The graph demonstrates that as number of attributes increases, decryption time increases.



Graph 6.3: Decryption Time vs Number of Attributes

7. CONCLUSION

In this paper, a fine-grained authorized expressive keyword search scheme has been proposed. The proposed scheme uses the benefits of CP-ABE to perform efficient keyword search over encrypted data and for authorizing users.The security analysis is performed under standard complexity assumptions and it shows that the proposed scheme is semantically secure against the chosen keyword attack. The scheme is also resistant against keyword guessing attacks. Moreover, the performance analysis of the proposed scheme shows that it outperforms the closely-related works in terms of storage, communication overhead, and computational overhead.

Security, efficiency, and query expressiveness are the three main factors that define the practical usability of a keyword search scheme. A user expects that the keyword search scheme provides strong security and high efficiency (in terms of computation and communication costs), as high as for normal plaintext search. Further, the query expressiveness will provide the user to make different types of queries. In this paper, we have made an effort to maintain a balance among these three factors. We believe that there is still much work to do for improving the balance among these three factors, i.e., security, efficiency, and query expressiveness. This will lead to the use of the keyword search schemes in privacy-preserving personalized services over the Internet like online advertisements or news.

8. REFERENCES

[1] J. Tang, Y. Cui, Q. Li, K. Ren, J. Liu, and R. Buyya, “Ensuring security and privacy preservation for cloud data services,” ACM Comput. Survey, vol. 49, no. 1, pp. 13:1–13:39, Jun. 2016.

[2] R. Curtmola, J. Garay, S. Kamara, and R. Ostrovsky, “Searchable symmetric encryption: Improved definitions and efficient constructions,” in Proc. 13th ACM Conf. Comput. Commun. Secur., 2016, pp. 79–88.

[3] W. Sun, B. Wang, N. Cao, M. Li, W. Lou, Y. T. Hou, and H. Li, “Privacy-preserving multi-keyword text search in the cloud supporting similarity-based ranking,” in Proc. 8th ACM SIGSAC Symp. Inf. Compute. Commun. Secur., 2013, pp. 71–82.

[4] 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. 2014.

[5] P. Xu, S. Tang, P. Xu, Q.Wu,H. Hu, and W. Susilo, “Practical multi-keyword and Boolean search over encrypted E-mail in cloud server,” IEEE Trans. Serv. Comput., 2019, doi:10.1109/TSC.2019.2903502.

[6] W. Sun, S. Yu, W. Lou, Y. T. Hou, and H. Li, “Protecting your right: Verifiable attribute-based keyword search with fine-grained owner-enforced search authorization in the cloud,” IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 4, pp. 1187–1198, Apr. 2016.

[7] J. Baek, R. Safavi-Naini, and W. Susilo, “Public key encryption with keyword search revisited,” in Proc. Int. Conf. Comput. Sci. Appl. Part I, 2008, pp. 1249–1259.

[8] T. Okamoto and K. Takashima, “Hierarchical predicate encryption for inner-products,” in Proc. Int. Conf. Theory Appl. Cryptology Inf. Secur., 2009, pp. 214–231.