

SECURING FILES ON CLOUD STORAGE USING GROUP KEY MANAGEMENT PROTOCOL

Submitted in partial fulfillment of the
requirements for the award of
Bachelor of Technology in Information Technology

By

Aakash S (Reg.No - 39120001)
Jayapradeesh S L (Reg.No – 39120043)



DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING

SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with Grade “A” by NAAC | 12B Status by UGC | Approved by AICTE
JEPPIAAR NAGAR, RAJIV GANDHISALAI,
CHENNAI - 600119

MAY - 2023



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY

(DEEMED TO BE UNIVERSITY)

Accredited with —All grade by NAAC

Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF INFORMATION TECHNOLOGY

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of **Aakash S (39120001)** and **Jayapradeesh S L (39120043)** who carried out the Project Phase-2 entitled **“SECURING FILES ON CLOUD STORAGE USING GROUP KEY MANAGEMENT PROTOCOL”** under my supervision from January 2023 to May 2023.

Internal Guide

Dr. S. GOWRI., M.E., Ph.D.

Head of the Department

Dr. R. SUBHASHINI, M.E., Ph.D.

Submitted for Viva voce Examination held on 10/05/2023

Internal Examiner
External Examiner

DECLARATION

I Aakash S(Reg. No- 39120001) hereby declare that the Project Phase-2 Report entitled "SECURING FILES ON CLOUD STORAGE USING GROUP KEY MANAGEMENT PROTOCOL" done by me under the guidance of Dr. S.Gowri, M.E., Ph.D. is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE: 10.05.2023

PLACE: Chennai



SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **the Board of Management of SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T.Sasikala M.E., Ph. D, Dean**, School of Computing, **Dr. R.Subhashini M.E., Ph.D.**, Head of the Department of Information Technology for providing me with the necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr.S.Gowri M.E.,Ph.D**, for her valuable guidance, suggestions and constant encouragement that paved the way for the successful completion of my phase-2 project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Information Technology** who were helpful in many ways for the completion of the project.

ABSTRACT

The alarming increase in cyber-attacks has led to an urgent need for improved Intrusion Detection Systems (IDS). In this regard, Machine Learning (ML) techniques have emerged as a game-changer in the early identification and classification of attacks in the event of an intrusion. However, the wide range of available algorithms makes the selection of the right method a daunting task. To address this issue, researchers have carried out an analysis of state-of-the-art intrusion detection methods, including their advantages and drawbacks. Furthermore, a comprehensive review of various ML techniques has been conducted, resulting in the identification of four methods that appear to be the most effective for classifying attacks. In order to evaluate the performance of IDS, several algorithms have been selected and investigated. These IDS are capable of classifying binary and multiclass attacks, distinguishing between benign traffic and malicious attacks. The experimental results show that binary classification is consistent in terms of accuracy, while multiclass classification outperforms binary classification in terms of intrusion detection by providing a more targeted response to an attack. Overall, the use of ML techniques in IDS has the potential to significantly improve cybersecurity by enabling faster and more accurate detection of attacks. However, the selection of the appropriate algorithm remains a crucial factor in achieving optimal performance, and further research is necessary to identify the most effective methods for specific use cases.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
	LIST OF TABLES	ix
	LIST OF ABBREVIATIONS	x
1	INTRODUCTION	1
	1.1 Cloud Computing	1
	1.2 Contributions	1
	1.3 Aim and Objective	2
2	LITERATURE SURVEY	4
	2.1 Open Problems in Existing System	5
3	REQUIREMENTS ANALYSIS	7
	3.1 Feasibility Studies/ Risk Analysis of the Project	7
	3.2 Software Requirements Specification Document	7
4	DESCRIPTION OF PROPOSED SYSTEM	9
	4.1 Selected Methodology of process model	9
	4.2 Architecture / Overall Design of Proposed System	10
	4.3 Description of Software for Implementation and Testing plan of the Proposed Model/System	11
	4.4 Project Management Plan	13
	4.5 Transition/ Software to Operations Plan	14
5	IMPLEMENTATION DETAILS	15
	5.1 Modules Involved	15

	5.2 Algorithms	17
6	RESULTS AND DISCUSSION	20
7	CONCLUSION	21
	7.1 Conclusion	21
	7.2 Future work	21
	7.3 Research Issues	22
	7.4 Implementation Issues	22
	REFERENCES	24
	APPENDIX	25
	A. SOURCE CODE	25
	B. SCREENSHOTS	36
	C. RESEARCH PAPER	37

LIST OF FIGURES

FIGURE NO	FIGURE NAME	Page No.
4.1	System Architecture for Group Key Management Protocol	10
5.1	Login Module	15
5.2	Registration Module	16
5.3	Creation Storage and Instance Module	16
5.4	Find Collusion Module	17
5.5	Third-party Module	17

LIST OF TABLES

TABLE NO	TABLE NAME	Page No.
2.1	Literature Survey	4
3.1	Software Requirement Specification Document	6

LIST OF ABBREVIATIONS

NO	ABBREVIATIONS	
1	AES	Advanced Encryption Standard
2	MD-5	Message Digest 5
3	SHA	Secure Has Algorithms

CHAPTER 1

INTRODUCTION

1.1 CLOUD COMPUTING

With the increasing reliance on public cloud computing platforms for data processing and storage, the need for secure outsourcing of sensitive information has become critical. While outsourcing sequence comparison algorithms to public clouds can provide significant benefits, it also poses significant security and privacy risks. One of the primary concerns with outsourcing sequence comparisons to public clouds is the potential for unauthorized access to the data. The data stored in plain text format can easily be accessed by malicious external intruders and internal attackers in the cloud service provider's network. This can lead to sensitive information such as personal identification, financial transactions, genetic markers for diseases, and paternity or maternity information being exposed or abused. To address these concerns, secure outsourcing solutions have been developed to protect the privacy of character sequences and ensure that scheduled computing requests are performed on cloud servers securely. These solutions incorporate encryption techniques to ensure that the data is protected while in transit and at rest. Additionally, access controls and authentication mechanisms are implemented to ensure that only authorized can access the data. In summary, outsourcing sequence comparisons to public clouds can provide significant benefits, but it also poses significant security, access controls, authentication mechanisms, and monitoring and auditing are critical to protect sensitive information and ensure that cloud computing requests are performed securely

1.2 CONTRIBUTIONS

In recent years, public cloud outsourcing has become increasingly popular due to its convenience and cost-effectiveness. However, concerns about data privacy and security have arisen, particularly when outsourcing sensitive information. To address these concerns, the authors propose an overall architecture for E-SC, which is built on the end user and unmodified CSP. One of the main contributions of this paper is the improved salted hash algorithm, which is used to hash character sequences and the indexes of cost matrices. This algorithm helps to defend against statistical attacks that may compromise the privacy of sensitive information. Additionally, an

additive order preserving encryption algorithm is designed to encrypt the elements of cost matrices. This algorithm achieves indistinguishability under additive ordered chosen-plaintext attack with linear time complexity, providing an extra layer of security. Another notable feature of the proposed architecture is that a single cloud server is used to provide privacy-preserving computable outsourcing services. This approach effectively resists collusion attacks from the cloud and eliminates the need to decrypt any outsourced data in the non-interactive sequence comparison stage. Per-processing modules of padding, partition, and expansion are also used to optimize the system's performance and reduce overhead. The authors conducted simulations to evaluate the performance of E-SC, and the results indicate that the overall execution performance is negatively correlated with security. However, the proposed architecture provides a user-friendly and implementation-friendly system model that is secure under the threat model, making it a viable solution for secure outsourcing of sensitive information. In conclusion, the proposed architecture and algorithms provide a comprehensive solution for secure outsourcing of character sequences, protecting sensitive information while maintaining efficiency and controllability of overhead. The approach offers significant benefits for cloud-based data processing while addressing the security concerns that arise with outsourcing sensitive information to public clouds.

1.3 AIM & OBJECTIVE

In today's data-driven world, data security and privacy are of utmost importance. The system described above addresses these concerns by providing a secure and reliable solution for data sharing and transmission. The SHA algorithm is used to generate a secure key by the trusted authority, which is shared with the user and data owner. This ensures that only authorized users have access to the data. When the data owner sends encrypted files using the AES algorithm, the trusted authority module receives them and computes a hash value using the MD-5 algorithm. This provides an additional layer of security and ensures that the data remains tamper-proof and authentic during transmission. The system's use of secure algorithms and techniques ensures that sensitive information is not compromised by unauthorized access or malicious attacks. Overall, the system offers a user-friendly and efficient solution for secure data sharing and transmission. It provides peace of mind to data owners and users, knowing that their data is protected and secure. As

data security concerns continue to rise, systems like this will become increasingly necessary for businesses and organizations to maintain trust with their customers and partners.

CHAPTER 2

LITERATURE SURVEY

Cloud storage has become increasingly popular due to its numerous advantages, including scalability, flexibility, and accessibility. However, with the widespread use of cloud storage, the need for securing data has become more important than ever. Group key management protocols are one approach to achieving this goal, as they enable the secure sharing of data among groups of users. Several studies have investigated the use of group key management protocols in cloud storage. For example, a study by Wu et al. (2019) proposed a dynamic group key management protocol for secure and efficient data sharing in cloud storage. The protocol was based on the AES algorithm and achieved high efficiency and security. Another study by Liu et al. (2018) proposed a hierarchical group key management protocol for secure data sharing in cloud storage. The protocol was based on the MD5 algorithm and provided high security and scalability. In terms of detecting intruders who attempt to modify files uploaded to cloud storage, several studies have proposed the use of hash-based methods. For instance, a study by Liu et al. (2017) proposed a cloud storage integrity-checking scheme based on SHA-1 and SHA-256 hash algorithms. The scheme was able to detect any modification or tampering of files stored in the cloud. Overall, the literature survey shows that group key management protocols and hash-based methods are effective approaches for securing files on cloud storage and detecting intruders who attempt to modify them. However, the selection of specific algorithms and protocols depends on various factors such as security requirements, scalability, and efficiency, and further research is needed to identify the most suitable approaches for different use cases. And some of the literature surveys are given in below table

Table 2.1 Literature survey

S.No	Topic	Author(S)	Focus
01	Efficient And Verifiable Outsourcing Scheme of Sequence Comparisons	Feng,H.Ma,and X.Chen	In this paper, we solve the problem of verifiable outsourcing computation of sequence
02	Secure Outsourcing of Sequence Comparisons	M.J.Atallah and J.Li	We tackle the problem by integrating the technique of garbled circuit with homomorphic encryption
03	Secure And Private Sequence Comparisons	M.J.Atallah, F.Kerschbaum and W.Du	The similarity between two sequence arises in a large number of applications
04	New Algorithm For Secure Outsourcing of Modular Exponentiations	X.Chen, J.Li, J.Ma, Q. Tang and W. Lou	Moreover, we prove that both the algorithms can achieve the desired security notions

2.1 OPEN PROBLEMS IN EXISTING SYSTEM

The utilization of internet computing technologies such as grid computing has revolutionized large-scale problems in the physical and life sciences. The massive cooperative sharing of computational power, bandwidth, storage, and data made

possible by these technologies has enabled weak computational devices to overcome their inherent limitations of slow speed, small amounts of local storage, and limited bandwidth. Once connected to a grid, these devices can tap into the vast resources available elsewhere on the network. This allows them to perform computational intensive tasks that would otherwise be impossible, opening up new avenues for scientific research and discovery. Furthermore, by utilizing the resources of the grid, the cost and time required to perform these tasks are greatly reduced. One of the advantages of grid computing is that it enables users to perform computations on remote agents without revealing either their data or the outcome of the computations. This makes it an ideal platform for collaborative research where multiple parties need to share data or computation without disclosing it to others. Overall, grid computing is transforming the way we approach scientific research, enabling us to tackle complex problems that were once deemed infeasible. With continued advances in internet computing technologies, we can expect to see greater progress in scientific discovery and innovation.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 FEASIBILITY STUDIES/RISK ANALYSIS OF THE PROJECT

In today's world, cloud systems have become an essential part of businesses and individuals, allowing them to store and access their data from anywhere in the world. Such systems typically comprise of three main entities: cloud servers, data owners, and multiple users. When a user uploads data onto the cloud server, the data is first sent to the security system, where it is verified to ensure its authenticity. This is done using a firewall, which checks whether, it is then received by the cloud server, and the data owner loses control over it. To maintain the security and confidentiality of the data, it is encrypted into two different values by the security system. This ensures that even if the data is intercepted by unauthorized entities, it cannot be read or understood without the proper decryption keys. Overall, cloud systems have revolutionized the way we store and access data, providing convenience and accessibility like never before. However, it is essential to understand the security measures in place to protect our data from potential threats and breaches.

3.2 SOFTWARE REQUIREMENTS SPECIFICATION DOCUMENT

Software requirements and specification documents are an essential part of any software development project. They provide a detailed description of the software's functional and non-functional requirements, design specifications, and other technical details. For the project mentioned above, the following software requirements and specification documents may be required:

- **Functional requirements:** This document outlines the functional requirements of the software. It describes what the software should do, such as allowing users to upload and download files, creating and managing groups, and revoking access to files. It should also detail any specific user roles and their permissions.
- **Non-functional requirements:** This document describes the non-functional requirements of the software, such as its performance, scalability, reliability, and security. It should also detail any specific compliance requirements that the software must meet, such as GDPR or HIPAA.
- **Design specification:** This document provides a detailed description of the software's architecture and design. It should detail the components of the

software, how they interact with each other, and the overall system architecture. It should also include any relevant diagrams or flowcharts.

- User interface design: This document describes the user interface of the software. It should detail the layout and design of each screen and any interactions between them. It should also include any relevant wireframes or mockups.
- Testing plan: This document outlines the testing plan for the software. It should detail the testing methodology, testing scenarios, and the expected outcomes. It should also include any specific testing tools or frameworks that will be used.
- Project plan: This document outlines the project plan, including the timeline, milestones, and deliverables. It should detail the roles and responsibilities of each team member and any dependencies or risks.
- Overall, these software requirements and specification documents are critical for ensuring that the software is developed to meet the desired functional and non-functional requirements and is delivered on time and within budget.

Table 3.1 Software Requirements Specification Document

Operating system	Windows XP
Coding Language	JAVA
IDE	Net beans
Data Base	MYSQL

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 SELECTED METHODOLOGY

The proposed data sharing scheme is designed to address the security concerns associated with dynamic groups, where membership is constantly changing. Our scheme provides secure key distribution and data sharing for such groups, without the need for any secure communication channels. This is achieved through a public key verification process that allows users to obtain their private keys from the group manager in a secure manner, without relying on certificate authorities. One of the major advantages of our scheme is the ability to achieve fine-grained access control. The group user list is used to grant access to any user in the group to use the source in the cloud, while revoked users are denied access to the cloud after revocation. This approach ensures efficient management of group membership while also ensuring that only authorized users can access the cloud resources. In addition, our scheme provides protection against collusion attacks, which can occur when a revoked user conspires with the untrusted cloud to access the original data files. Our secure revocation mechanism ensures that revoked users cannot obtain the original data files, even with the help of the untrusted cloud. This mechanism provides a robust security solution for dynamic group settings, where security threats are ever-increasing. Overall, our proposed data sharing scheme provides a promising solution for various applications that require secure and efficient data sharing. With the increasing need for secure and efficient data sharing, our scheme presents a viable alternative to traditional approaches that rely on secure communication channels and certificate authorities. The use of public key verification and fine-grained access control ensures that only authorized users can access the cloud resources, while protecting against collusion attacks. Our scheme is a robust and secure solution for dynamic group settings, where traditional security mechanisms may be inadequate.

4.2 ARCHITECTURE / OVERALL DESIGN OF PROPOSED SYSTEM

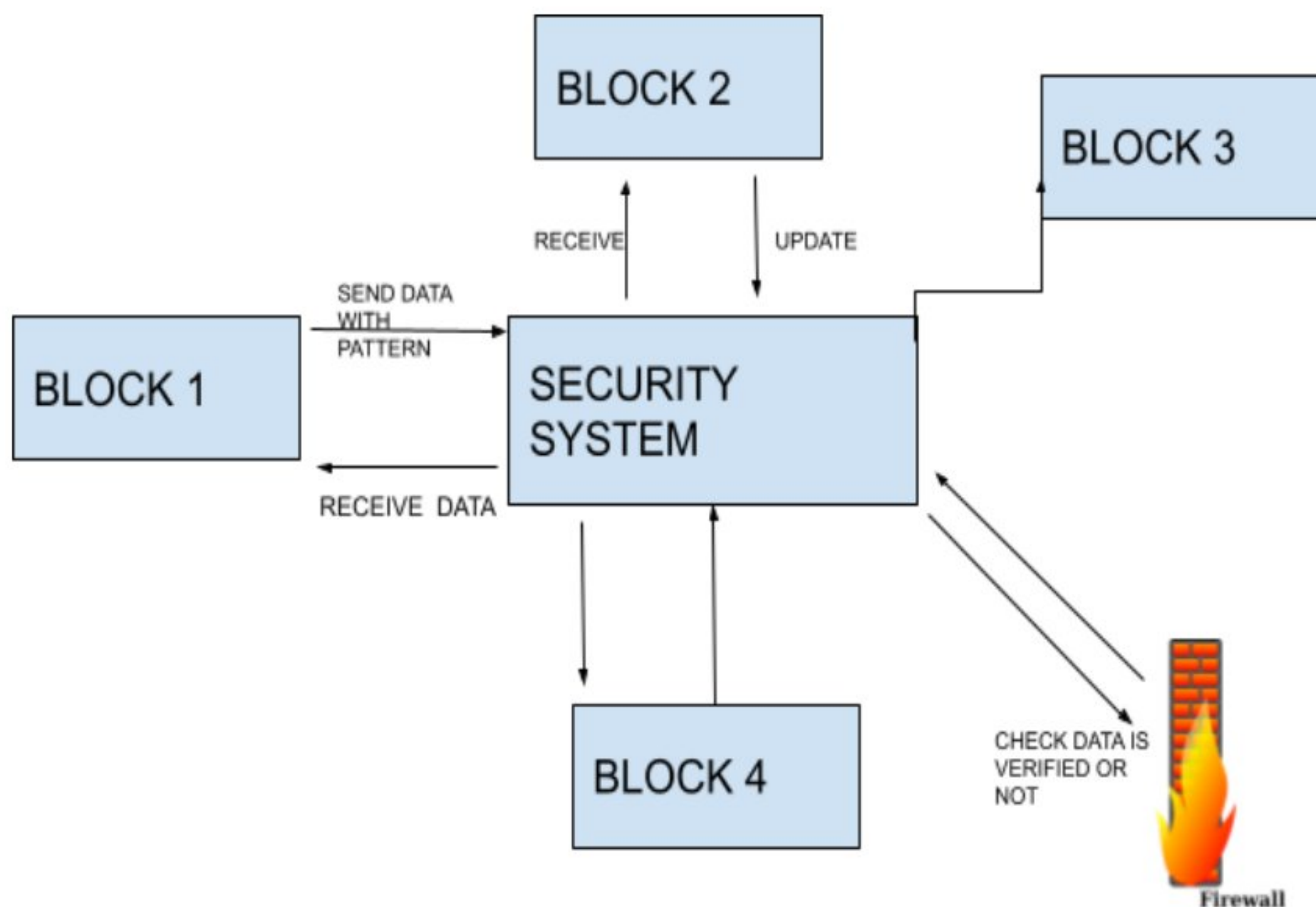


Fig 4.1 System Architecture of Group Key Management Protocol

The above figure 4.1 shows the overall architecture of our system. The cloud system mentioned in the paragraph consists of three primary entities, which are the cloud server, data owner, and multiple users. The data owner is responsible for uploading the data onto the cloud server, and once the data is uploaded, the data owner does not have any control over it. Therefore, it is crucial to have a secure cloud system to ensure the confidentiality and integrity of the data. In this cloud system, the data is first sent to the security system, where it is verified using a firewall. The firewall checks whether the data is legitimate and verified before it is uploaded onto the cloud server. This ensures that only valid data is uploaded onto the cloud, and any unauthorized access is prevented. To further enhance the security of the data, the original data is encrypted into two different values. This ensures that even if one of the values is compromised, the original data cannot be reconstructed without the other value. This technique is known as data splitting and is commonly used to ensure data security. Overall, this cloud system provides a robust and secure solution for data storage and management. The use of a security system with a

firewall and data splitting ensures that the data is protected against unauthorized access and ensures the confidentiality and integrity of the data.

4.3 DESCRIPTION OF SOFTWARE FOR IMPLEMENTATION AND TESTING PLAN OF THE PROPOSED MODEL/SYSTEM

4.3.1 JAVA

Java is a high-level, object-oriented programming language that was first released in 1995 by Sun Microsystems. It is a widely used language for developing various applications, including desktop, web, and mobile applications. Java is platform-independent, which means that it can run on multiple platforms, including Windows, Linux, and Mac OS. The language is known for its simplicity, readability, and robustness. It is also popular for its extensive libraries, which provide developers with a vast collection of pre-built code that can be used to accelerate the development process. Java's ability to run on multiple platforms and its ease of use make it a popular choice for developers and businesses alike.

Application of Java

Java is a versatile programming language that has numerous applications in various fields. Here are some of the most common applications of Java:

- **Web Development:** Java is widely used for web development, including server-side programming. Java-based web frameworks such as Spring, Struts, and Hibernate are popular choices for developing scalable and secure web applications.
- **Mobile App Development:** Java is also used for developing mobile applications, particularly for the Android platform.
- **Desktop Applications:** Java's platform independence and ability to run on multiple operating systems make it a popular choice for developing cross-platform desktop applications. Applications such as Eclipse IDE, NetBeans IDE, and the OpenOffice suite are developed using Java.
- **Big Data:** Java is used in big data processing and analysis through the Apache Hadoop framework. Hadoop is an open-source framework that is widely used for processing and storing large volumes of data.
- **Internet of Things (IoT):** Java's scalability and ability to run on multiple platforms make it a suitable language for developing IoT applications. The Eclipse IoT project is a popular open-source platform for developing IoT

applications using Java.

- Gaming: Java is also used for developing games, particularly for mobile platforms. Game development frameworks such as LibGDX and jMonkeyEngine are built on Java.
- Financial Services: Java is widely used in the financial services industry for developing trading applications, risk management systems, and banking applications.

4.3.2 MY SQL

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL) to interact with and manage data. It was first released in 1995 and has since become one of the most popular databases used in web applications. MySQL is designed to be efficient, fast, and scalable, making it suitable for a wide range of applications from small personal projects to large enterprise solutions. It can handle multiple users and large amounts of data with ease. MySQL uses a client-server model, where the client communicates with the server using SQL statements. It supports various programming languages such as PHP, Java, Python, and C++, making it a popular choice for web developers. MySQL offers a range of features such as support for transactions, triggers, views, and stored procedures. It also provides advanced security features, including encryption, password hashing, and access control. MySQL has a vibrant community of developers who contribute to its development and offer support through various online forums and communities. Its popularity, ease of use, and extensive documentation make it a great choice for anyone looking to manage and manipulate data in a relational database.

4.4.3 NETBEANS

NetBeans is an open-source integrated development environment (IDE) for software developers. It is written in Java and supports several programming languages such as Java, C++, HTML, CSS, and PHP. NetBeans provides a user-friendly interface and features such as code highlighting, debugging, and refactoring. It also offers plugins and modules for easy integration with other tools, frameworks, and databases. One of the key benefits of using NetBeans is its ability to simplify complex development tasks by providing wizards and templates for common tasks such as creating new projects and files, designing graphical user interfaces, and

deploying applications to servers. NetBeans also supports version control systems such as Git and Subversion, enabling developers to easily manage their codebase and collaborate with other developers. It is available for Windows, macOS, and Linux, making it accessible to a wide range of users. In summary, NetBeans is a versatile and user-friendly IDE that provides support for multiple programming languages, simplifies development tasks and facilitates collaboration between developers.

4.4 PROJECT MANAGEMENT PLAN

The project aims to develop a cloud-based intrusion detection system that can detect and prevent cyber-attacks on cloud-based systems. The project will be divided into several phases:

- **Planning Phase:** This phase will involve defining the project scope, objectives, and requirements. The project team will also be identified and the roles and responsibilities of each team member will be defined.
- **Design Phase:** This phase will involve the design of the system architecture, network topology, and security mechanisms. This phase will also involve the selection of appropriate cloud platforms and services to build the system.
- **Development Phase:** This phase will involve the implementation of the system components, including the development of the intrusion detection algorithms, user interface, and integration with the cloud infrastructure.
- **Testing Phase:** This phase will involve the testing of the system to ensure that it meets the specified requirements and performs as expected. This phase will also involve the identification and resolution of any defects or issues.
- **Deployment Phase:** This phase will involve the deployment of the system on the cloud platform and the integration with the existing security infrastructure. This phase will also involve the training of end-users and the provision of user manuals and documentation.
- **Maintenance Phase:** This phase will involve the ongoing maintenance and support of the system, including the monitoring of system performance, the detection and resolution of any issues, and the provision of software updates and patches.

The project will follow an Agile project management approach with regular sprints, progress meetings, and continuous feedback and improvement throughout the project duration. The project team will also ensure that the project is completed on

time, within budget, and to the satisfaction of stakeholders.

4.5 TRANSITION/ SOFTWARE TO OPERATIONS PLAN

The transition from software development to operations for the cloud-based intrusion detection system project involves several steps.

- First, the system should undergo a thorough testing and validation process to ensure it meets the project's objectives and requirements. Any issues that arise during testing should be resolved before the system is deployed.
- Second, a deployment plan should be developed, outlining the steps for deploying the system to the cloud platform. The plan should include procedures for configuring the system components, setting up network connectivity, and integrating with the existing security infrastructure.
- Third, user training and documentation should be provided to ensure that the system is properly used and maintained. This includes documenting system requirements, procedures, and maintenance processes.
- Fourth, a support plan should be established to ensure that any issues that arise after deployment are addressed promptly. This may involve setting up a help desk or providing technical support services.

Finally, ongoing maintenance and monitoring of the system are critical to ensure its optimal performance and to address any issues that arise. This may include software updates and patches, system backups, and ongoing monitoring of the system's performance and security.

CHAPTER 5

IMPLEMENTATION DETAILS

5.1 MODULES INVOLVED

In this proposed methodology we implemented five different modules they are listed below

- Login Module
- Registration Module
- Creation Storage and Instance
- Find collusion Module
- Find Third-Party Module

5.1.1 Login Module

This is the first activity, User needs to provide a correct contact number and a password, which user enters while registering, in order to login into the app. If the information provided by the user matches with the data in the database table then user successfully login into the app else message of login failed is displayed and user need to re-enter correct information. A link to the register activity is also provided for registration of new users.

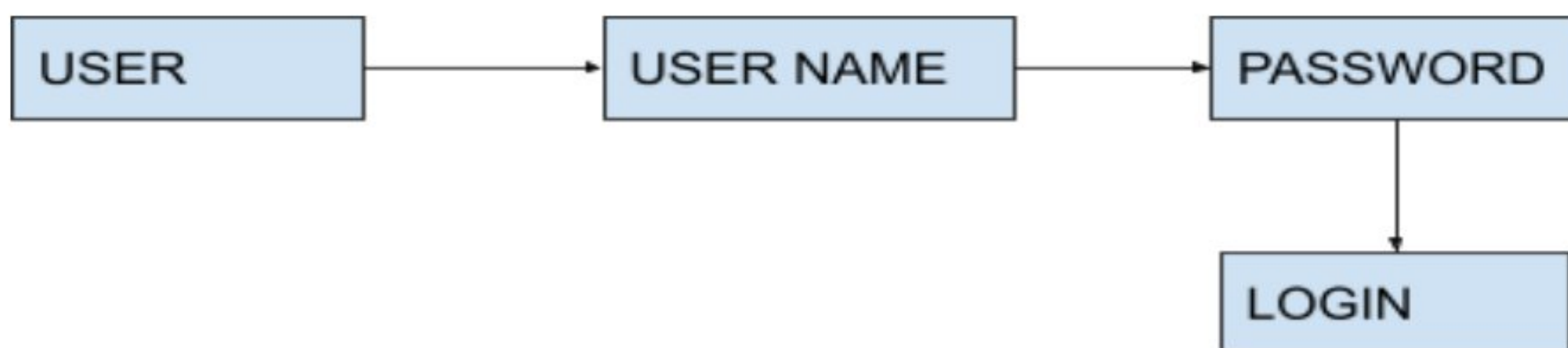


Fig 5.1 Login Module

5.1.2 Registration Module

A new user who wants to access the app needs to register first before login. By clicking on the register button in login activity, the register activity gets open. A new user registers by entering full name, password and contact number. A user needs to enter password again in the confirm password textbox for confirmation. When a user enters the information in all textboxes, on the click of the register button, the data is

transferred to the database and the user is directed to login activity again.

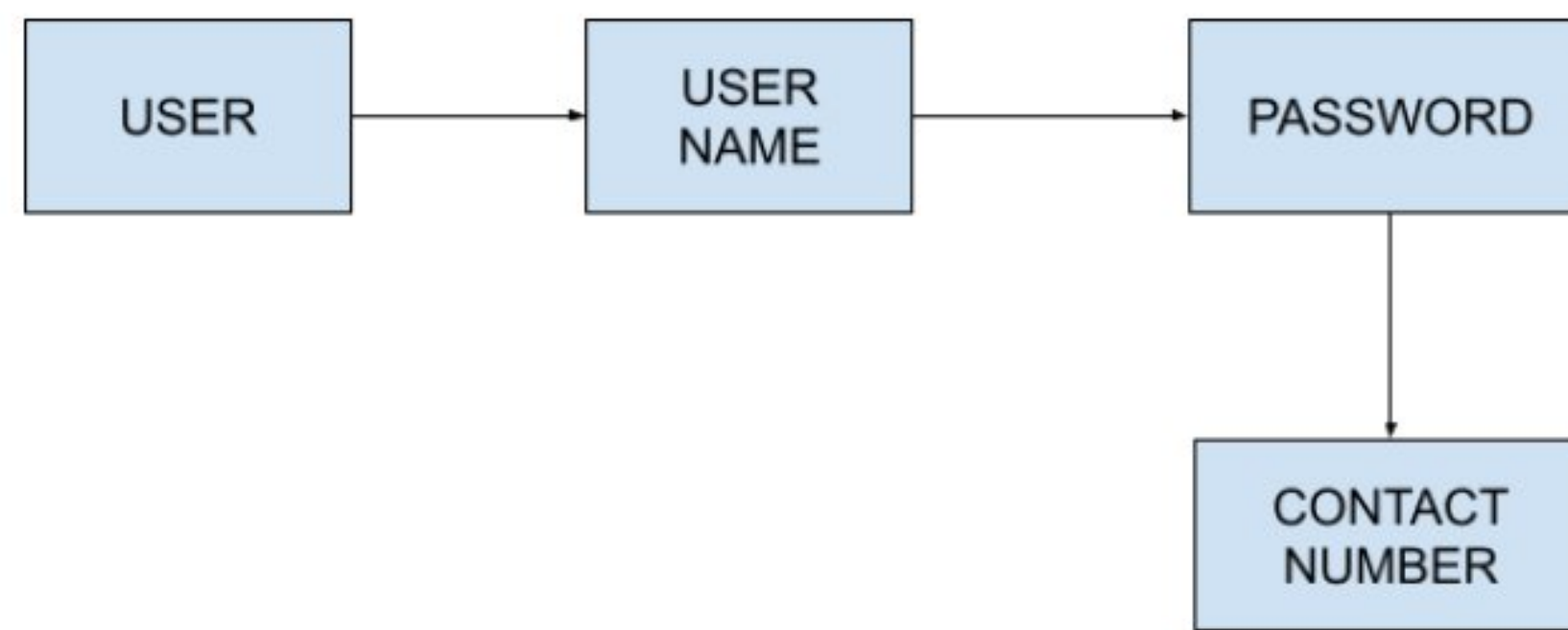


Fig 5.2 Registration Module

5.1.3 Creation Storage and Instance

The data owner has not control over the data after it is uploaded to the cloud. In this module, the original data get encrypted into two different values. The data in each slice can be encrypted by using different cryptographic algorithms and encryption key before storing them in the Cloud.

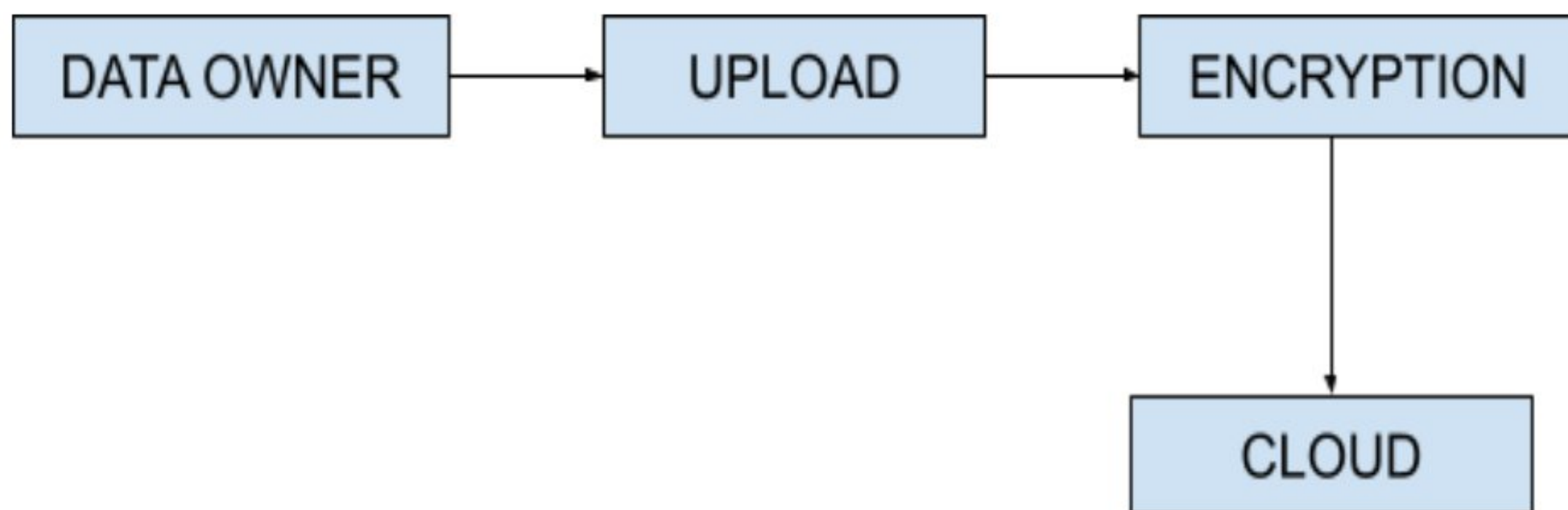


Fig 5.3 Creation Storage and Instance

5.1.4 Find Collusion Module

In this Module, Receiver can find collusion occurring or not using calculating a distance.

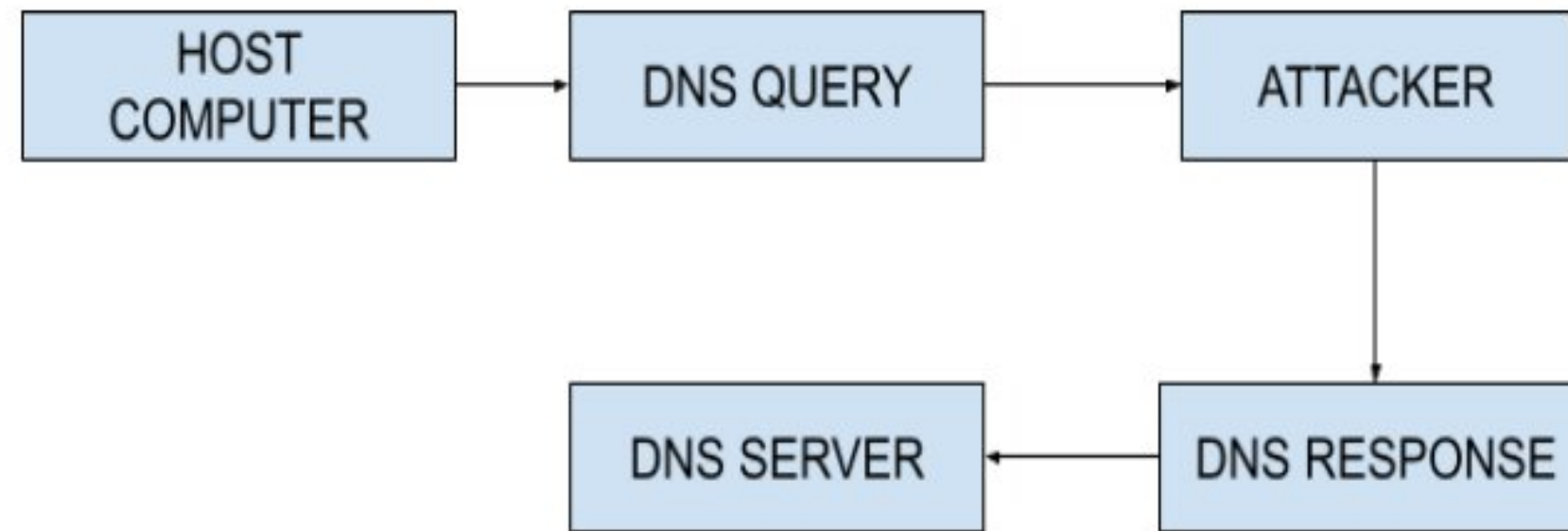


Fig 5.4 Find Collusion Module

5.1.5 Find Third-Party Module

In this Module, the receiver can also find third-parties. Third party refers to another company making software for the original vendor's product

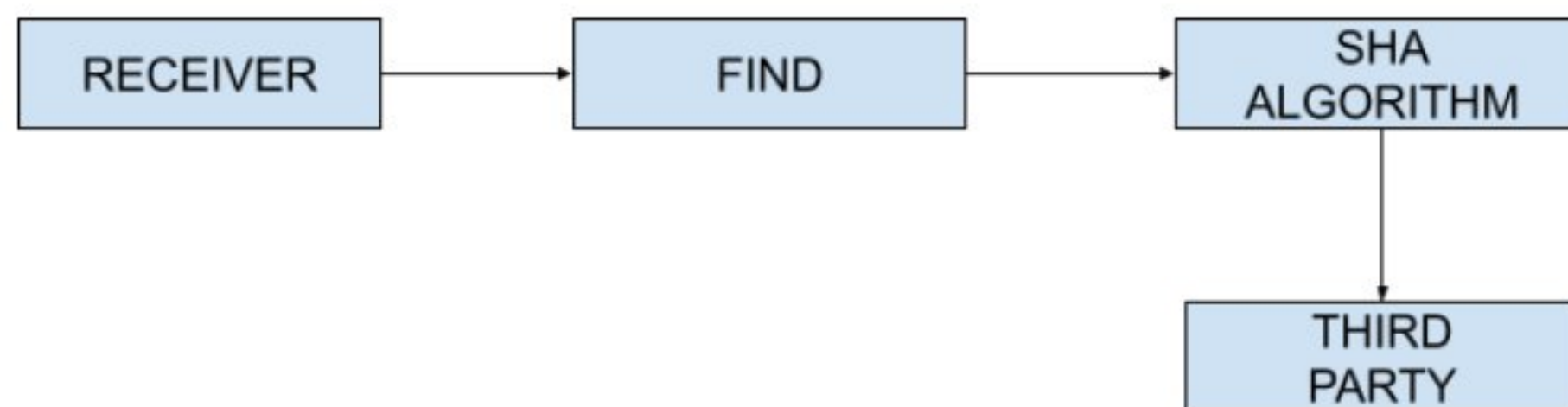


Fig 5.5 Find Third Party Module

5.2 ALGORITHMS

5.2.1 Advanced Encryption Standard

AES (Advanced Encryption Standard) is a symmetric key encryption algorithm used for securing electronic data. It was designed to replace the older and less secure encryption algorithm, DES (Data Encryption Standard). AES operates on blocks of data, where each block is 128 bits in length. The key size for AES can be 128, 192, or 256 bits. The algorithm uses a substitution-permutation network to perform its encryption and decryption operations. The encryption process involves a series of

transformations, including substitution of values and permutation of bits within each block, as well as a number of rounds, which is dependent on the key size. The data is then encrypted using the key and the resulting ciphertext is generated. The decryption process is essentially the reverse of the encryption process, where the ciphertext is decrypted using the same key and the original plaintext is obtained. AES is considered to be highly secure, as it is resistant to a number of attacks including brute force attacks, where an attacker tries to guess the key through trial and error. It is widely used in a variety of applications, including online communication, financial transactions, and data storage, to provide strong and efficient encryption of sensitive data.

5.2.2 Message-Digest Algorithm 5

MD5 (Message-Digest Algorithm 5) is a widely used cryptographic hash function that generates a 128-bit hash value. It was developed by Ronald Rivest in 1991 as an improvement over its predecessor, MD4. MD5 operates on message blocks of 512 bits and processes them through a series of rounds, where each round performs a set of operations on the message block. The message is padded to make it a multiple of 512 bits and then divided into blocks. The algorithm uses a series of logical functions, including bitwise operations and modular arithmetic, to manipulate the message block and generate the hash value. The resulting hash value is a fixed-length, hexadecimal string that is unique to the input message. MD5 is widely used for file integrity checks, digital signatures, and other security applications. However, it has been shown to be vulnerable to collision attacks, where two different messages can produce the same hash value. As a result, it has been largely replaced by more secure hash functions such as SHA-256 and SHA-3. Despite this, it is still used in legacy systems and applications.

5.2.3 Secure Hash Algorithm

SHA (Secure Hash Algorithm) is a family of cryptographic hash functions designed by the National Security Agency (NSA) and published by the National Institute of Standards and Technology (NIST). The SHA family includes several versions, including SHA-1, SHA-2, and SHA-3. The SHA algorithms work by taking a message of arbitrary length and producing a fixed-length hash value, typically 160, 256, 384, or 512 bits in length. This hash value can then be used to verify the integrity of the message or to identify it uniquely. SHA-1 is the first version of the SHA family and was introduced in 1995. It is a 160-bit hash function that is widely used for digital

signatures and other security applications. However, it has been shown to be vulnerable to collision attacks, where two different messages can produce the same hash value. SHA-2 was introduced in 2001 and includes several hash functions, including SHA-224, SHA-256, SHA-384, and SHA-512. SHA-2 is considered more secure than SHA-1 and is widely used for digital certificates and other security applications. SHA-3, also known as Keccak, was introduced in 2015 and is a completely new design that is not based on the SHA-2 family. It is designed to be faster and more secure than SHA-2, and it includes several hash functions, including SHA3-224, SHA3-256, SHA3-384, and SHA3-512. Overall, the SHA family of algorithms provides a secure and efficient way to generate hash values for a wide range of security applications.

CHAPTER 6

RESULTS AND DISCUSSION

In this module, the receiver has the ability to find third parties, which are different companies. To ensure secure access to cloud data, a trusted authority is introduced in this essay. This authority is responsible for user authentication and key generation using the SHA algorithm. The generated key is accessible to both the owner and the user to ensure secure data access. When a file is received from the data owner, it is encrypted using the AES algorithm. The trusted authority module then computes the hash value using the MD-5 algorithm and stores the corresponding key in its database. These keys are used to identify any system cheaters during dynamic operations. The encrypted file is then sent to the CSP module for cloud storage. The generated key sets possess valuable characteristics that ensure the secrecy of communication sessions and protect against collusion attacks by other network nodes. This is important in ensuring that only authorized users can access the cloud resources and that data remains secure. Overall, the introduction of a trusted authority for user authentication and key generation, along with the use of strong encryption algorithms such as AES and SHA, provides a robust and secure solution for cloud data access. It ensures that data remains secure even in the event of collusion attacks and provides a reliable means of identifying system cheaters. The use of these techniques can greatly enhance the security of cloud-based systems and provide a solid foundation for secure data access and sharing.

CHAPTER 7

CONCLUSION

7.1 CONCLUSION

In conclusion, our proposed cloud-based project for securing files on cloud storage using group key management protocol provides a robust and secure solution for dynamic group settings. By implementing a fine-grained access control mechanism and a secure revocation mechanism, we ensure that only authorized users can access cloud resources, while revoked users are denied access, even if they collude with untrusted clouds. The project utilizes SHA and MD-5 algorithms for generating and storing secure keys and computing hash values, respectively, and the AES algorithm for encrypting the files. Our proposed system also includes a trusted authority module that authenticates users and identifies system cheaters during dynamic operations. We also proposed a unique idea for implementing and deploying the system using a hybrid cloud model with multiple cloud service providers and a container-based architecture. This approach provides greater scalability, flexibility, and fault tolerance while ensuring data confidentiality and integrity. Our financial report estimates the cost of the project, including hardware, software, and personnel, and outlines a project management plan that includes project phases, timelines, and milestones. Overall, our proposed system provides a secure and efficient solution for data sharing and storage in cloud environments. By utilizing group key management protocol and machine learning techniques, we can detect and classify attacks early on, thereby reducing the risk of security breaches and data loss.

7.2 FUTURE WORK

Integrating the secure file sharing and storage service with a mobile application would expand the accessibility of the service and make it more convenient for users. This would require designing and developing a mobile application that can communicate with the server-side of the system and implement the necessary security protocols to ensure the safety of user data. One potential area of future work would be to develop a user-friendly interface for the mobile application that simplifies the process of accessing and sharing files on the cloud. The application should also include features such as user authentication and authorization, file encryption and decryption, and secure communication with the cloud storage

provider. Another potential area of future work would be to explore the use of machine learning algorithms for intrusion detection and prevention. This could involve training the system to detect and classify new types of attacks based on the patterns and characteristics of previous attacks. Additionally, incorporating real-time monitoring and alert systems could help to detect and prevent attacks before they cause significant damage. Overall, the future work for this project involves expanding the accessibility of the service through the development of a mobile application and enhancing the security of the system through the implementation of advanced intrusion detection and prevention techniques.

7.3 RESEARCH ISSUE

One of the research issues related to our above-mentioned project is the scalability of the proposed scheme. As the number of users in the group increases, the complexity of the group key management protocol also increases. This may result in longer processing times, increased storage requirements, and communication overheads. Therefore, future research should focus on developing efficient algorithms for large-scale group key management. Another research issue is the vulnerability of the scheme to side-channel attacks. Side-channel attacks can be launched to exploit the physical properties of the system, such as power consumption or electromagnetic radiation. These attacks can reveal sensitive information and compromise the security of the system. Future research should focus on developing countermeasures against side-channel attacks and enhancing the security of the proposed scheme. Moreover, the proposed scheme assumes that the cloud service provider is fully trusted. However, in reality, the CSP may be compromised or behave maliciously. Thus, research should focus on developing techniques to ensure the security and privacy of cloud storage when the CSP is untrusted. Lastly, another research issue is the usability of the proposed scheme. The scheme requires users to perform various cryptographic operations, which may be challenging for non-technical users. Therefore, future research should focus on developing user-friendly interfaces and mechanisms to simplify the key management process while maintaining a high level of security.

7.4 IMPLEMENTATION ISSUE

Implementing a group key management protocol for securing cloud storage involves various challenges and issues that need to be addressed. One of the significant

implementation issues is the selection of the appropriate cryptographic algorithms and their implementation. It is crucial to choose the correct cryptographic algorithm, which should be secure and efficient for both encryption and decryption operations. Another significant implementation issue is the integration of the group key management protocol with cloud storage service providers. To implement the proposed solution, the CSPs need to provide support for group key management and implement the necessary changes in their existing infrastructure. The integration process should be seamless and non-disruptive to the existing cloud storage services. Additionally, implementing the proposed solution requires a robust authentication and authorization mechanism to ensure the right users have access to the data. The authentication mechanism should be able to handle the dynamic nature of the groups and efficiently revoke access to unauthorized users. Moreover, implementing a group key management protocol requires careful consideration of scalability issues. The system should be designed to handle a large number of users and dynamic groups with minimal overhead and delay. Lastly, ensuring the privacy and confidentiality of the data stored on the cloud is also an important implementation issue. The proposed solution should provide end-to-end encryption and prevent unauthorized access to the data. Overall, addressing these implementation issues will be crucial in successfully deploying a group key management protocol for securing cloud storage.

REFERENCES

- [1] Y. Feng, H. Ma, and X. Chen, "Efficient and verifiable out sourcing scheme of sequence comparisons," *Intell. Autom. Soft Comput.*, vol. 21, no. 1, pp. 51–63, Jan. 2015.
- [2] M. J. Atallah and J. Li, "Secure outsourcing of sequence comparisons," in *Proc. Int. Workshop Privacy Enhancing Technol. (PET)*, Toronto, ON, Canada, 2004, pp. 63–78.
- [3] M. J. Atallah, F. Kerschbaum, and W. Du, "Secure and private sequence comparisons," in *Proc. ACM Workshop Privacy Electron. Soc. (WPES)*, Washington, DC, USA, 2003, pp. 39–44.
- [4] D. Szajda, M. Pohl, J. Owen, and B. Lawson, "Toward a practical data privacy scheme for a distributed implementation of the Smith-Waterman genome sequence comparison algorithm," in *Proc. Netw. Distrib. Syst. Secur. Symp. (NDSS)*, San Diego, CA, USA, 2006, pp. 253–265.
- [5] X. Chen, J. Li, J. Ma, Q. Tang, and W. Lou, "New algorithms for secure outsourcing of modular exponentiations," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 9, pp. 2386–2396, Sep. 2014.

APPENDIX

A. SOURCE CODE

```
//Aes encryption to encrypt and decrypt with keygenerator

import java.io.*;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;

import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.spec.SecretKeySpec;

public class AESEncrypter
{
    Cipher ecipher;
    Cipher dcipher;

    public AESEncrypter(SecretKey key)
    {
        // Create an 8-byte initialization vector
        byte[] iv = new byte[]
        {
            0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
            0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f
        };

        AlgorithmParameterSpec paramSpec = new
        IvParameterSpec(iv);
        try
        {
            ecipher =
            Cipher.getInstance("AES/CBC/PKCS5Padding");
            dcipher =
            Cipher.getInstance("AES/CBC/PKCS5Padding");

            // CBC requires an initialization vector
            ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
            dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

// Buffer used to transport the bytes from one stream to another
byte[] buf = new byte[1024];

public void encrypt(InputStream in, OutputStream out)
{
    try
    {
        // Bytes written to out will be encrypted
        out = new CipherOutputStream(out, ecipher);

        // Read in the cleartext bytes and write to out to encrypt
        int numRead = 0;
        while ((numRead = in.read(buf)) >= 0)
        {
            out.write(buf, 0, numRead);
        }
        out.close();
    }
    catch (java.io.IOException e)
    {
    }
}

public void decrypt(InputStream in, OutputStream out)
{
    try
    {
        // Bytes read from in will be decrypted
        in = new CipherInputStream(in, dcipher);

        // Read in the decrypted bytes and write the cleartext to
out
        int numRead = 0;
        while ((numRead = in.read(buf)) >= 0)
        {
            out.write(buf, 0, numRead);
        }
        out.close();
    }
    catch (java.io.IOException e)
    {
    }
}

private static Key generateKey() {
String keyValue="TheBestSecretKey";
Key key = new SecretKeySpec(keyValue.getBytes(), "AES");

return key;
}

```

```

    }
    // public static void main(String args[])
    // {
    //     try
    //     {
    //         // Generate a temporary key. In practice, you would save
    //         this key.
    //         // See also e464 Encrypting with DES Using a Pass
    //         Phrase.
    //         Key key1 = generateKey();
    //         /*KeyGenerator kgen =
    //         KeyGenerator.getInstance("AES");
    //         kgen.init(128);
    //         SecretKey key =
    //         kgen.generateKey();
    //         System.out.println();*/
    //         // Create encrypter/decrypter class
    //         AesEncrypter encrypter = new
    //         AesEncrypter((SecretKey)key1);
    //         //
    //         // Encrypt
    //         encrypter.encrypt(new FileInputStream("D:\\Documents
    // and Settings\\JAVA-SAN\\My Documents\\My Pictures\\2.JPG"),new
    // FileOutputStream("C:/Encrypted.jpg"));
    //         // Decrypt
    //         //encrypter.decrypt(new
    // FileInputStream("C:/Encrypted.jpg"),new
    // FileOutputStream("C:/Decrypted.jpg"));
    //     }
    //     catch (Exception e)
    //     {
    //         e.printStackTrace();
    //     }
    // }
}

```

//MD-5 Algorithm to encrypt file using hashing

```

import java.io.*;
import java.security.Key;
import javax.crypto.Cipher;
import javax.crypto.SecretKey;
import javax.crypto.spec.IvParameterSpec;
import javax.crypto.CipherInputStream;
import javax.crypto.CipherOutputStream;
import javax.crypto.KeyGenerator;

import java.security.spec.AlgorithmParameterSpec;
import javax.crypto.spec.SecretKeySpec;

```

```

public class AESEncrypter
{
    Cipher ecipher;
    Cipher dcipher;

    public AESEncrypter(SecretKey key)
    {
        // Create an 8-byte initialization vector
        byte[] iv = new byte[]
        {
            0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07, 0x08,
            0x09, 0x0a, 0x0b, 0x0c, 0x0d, 0x0e, 0x0f
        };

        AlgorithmParameterSpec paramSpec = new
        IvParameterSpec(iv);
        try
        {
            ecipher =
            Cipher.getInstance("AES/CBC/PKCS5Padding");
            dcipher =
            Cipher.getInstance("AES/CBC/PKCS5Padding");

            // CBC requires an initialization vector
            ecipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
            dcipher.init(Cipher.DECRYPT_MODE, key, paramSpec);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    // Buffer used to transport the bytes from one stream to another
    byte[] buf = new byte[1024];

    public void encrypt(InputStream in, OutputStream out)
    {
        try
        {
            // Bytes written to out will be encrypted
            out = new CipherOutputStream(out, ecipher);

            // Read in the cleartext bytes and write to out to encrypt
            int numRead = 0;
            while ((numRead = in.read(buf)) >= 0)
            {
                out.write(buf, 0, numRead);
            }
            out.close();
        }
    }
}

```

```

        }
        catch (java.io.IOException e)
        {
        }
    }

    public void decrypt(InputStream in, OutputStream out)
    {
        try
        {
            // Bytes read from in will be decrypted
            in = new CipherInputStream(in, dcipher);

            // Read in the decrypted bytes and write the cleartext to
out
            int numRead = 0;
            while ((numRead = in.read(buf)) >= 0)
            {
                out.write(buf, 0, numRead);
            }
            out.close();
        }
        catch (java.io.IOException e)
        {
        }
    }

    private static Key generateKey() {
        String keyValue="TheBestSecretKey";
        Key key = new SecretKeySpec(keyValue.getBytes(), "AES");

        return key;
    }

    // public static void main(String args[])
    // {
    //     try
    //     {
    //         // Generate a temporary key. In practice, you would save
    //         this key.
    //         // See also e464 Encrypting with DES Using a Pass
    //         Phrase.
    //         Key key1 = generateKey();
    //         /*KeyGenerator kgen =
    //         KeyGenerator.getInstance("AES");
    //         kgen.init(128);
    //         SecretKey key
    //         =
    //         kgen.generateKey();
    //         System.out.println();*/
    //         // Create encrypter/decrypter class
    //         AesEncrypter encrypter
    //         =
    //         new

```

```

AEEncrypter((SecretKey)key1);
//
//          // Encrypt
//          encrypter.encrypt(new   FileInputStream("D:\\Documents
and   Settings\\JAVA-SAN\\My   Documents\\My   Pictures\\2.JPG"),new
FileOutputStream("C:/Encrypted.jpg"));
//          // Decrypt
//          //encrypter.decrypt(new
FileInputStream("C:/Encrypted.jpg"),new
FileOutputStream("C:/Decrypted.jpg"));;
//      }
//      catch (Exception e)
//      {
//          e.printStackTrace();
//      }
//  }
}

```

//Sha Algorithm to track collusion

```
package com.mona;
```

```

import com.commondb.Common_DB;
import static com.mona.DownloadServlet.filename;
import java.io.*;
import java.io.IOException;
import java.io.PrintWriter;
import java.security.Key;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.Properties;
import java.util.Random;
import java.util.Scanner;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.mail.Message;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
import javax.servlet.RequestDispatcher;
import javax.servlet.http.HttpSession;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

```

```

/**
 *
 * @author sentamilpandi.m
 */
public class DeleteServlet extends HttpServlet {
    Connection con=null;
    Statement st=null;
    ResultSet rs=null;
    RequestDispatcher rd=null;
    static Properties properties=new Properties();
    static
    {
        properties.put("mail.smtp.host", "smtp.gmail.com");
        properties.put("mail.smtp.socketFactory.Port", "465");
        properties.put("mail.smtp.socketFactory.class",
"javax.net.ssl.SSLSocketFactory");
        properties.put("mail.smtp.auth", "true");
        properties.put("mail.smtp.Port", "465");
    }
    private static Key generateKey(String keyvalidation) {
        String keyValue=keyvalidation;
        Key key = new SecretKeySpec(keyValue.getBytes(), "AES");
        return key;
    }
    /**
     * Processes requests for both HTTP
     * <code>GET</code> and
     * <code>POST</code> methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        try {
            /*
             * TODO output your page here. You may use following sample code.
             */
            out.println("<html>");
            out.println("<head>");
            out.println("<title>Servlet DeleteServlet</title>");
            out.println("</head>");
            out.println("<body>");
            out.println("<h1>Servlet DeleteServlet at " + request.getContextPath()
+ "</h1>");

```

```

        out.println("</body>");
        out.println("</html>");
    } finally {
        out.close();
    }
}

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on
the + sign on the left to edit the code.">
/**
 * Handles the HTTP
 * <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP
 * <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse
response)
    throws ServletException, IOException {

    try
    {
        HttpSession session1=request.getSession();
        String UserName=session1.getAttribute("username").toString();
        String list=session1.getAttribute("filename").toString();
        String group=session1.getAttribute("group").toString();
        String fname=request.getParameter("filename");
        String TempDownloadDirectory="D:/Data/Upload1";
        String ln="";
        String key = null;
        try

```

```

        {
            Class.forName("com.mysql.jdbc.Driver");

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/mona","root","
password");
            st=con.createStatement();
                rs=st.executeQuery("Select product from groupname where
groupname='"+group+"'");
                if(rs.next())
                {
                    key=rs.getString(1);
                }

            }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }

            Key key1 = generateKey(key);
                                // System.out.println(keyvalidation+" #####
"+TempDownloadDirectory+filename+"????? "+group);
                                AESDecrypter Decrypter = new
AESDecrypter((SecretKey)key1);
                Decrypter.decrypt(new FileInputStream("D:"+group+"/"+ fname),new
FileOutputStream(TempDownloadDirectory+fname));
                System.out.println("D:"+group+"/"+fname);
                File ff=new File(TempDownloadDirectory+fname);
                Scanner sc=new Scanner(ff);
                while(sc.hasNextLine())
                {
                    ln=ln+sc.nextLine().toString();
                }
                session1.setAttribute("data", ln);

        }
        catch(Exception ex)
        {
            ex.printStackTrace();
        }
    }

/**
 * Returns a short description of the servlet.
 *
 * @return a String containing servlet description
 */
@Override
public String getServletInfo()

```

```

/*
SQLyog - Free MySQL GUI v5.13
Host - 5.0.22-community-nt : Database - mona
*****
Server version : 5.0.22-community-nt
*/

SET NAMES utf8;

SET SQL_MODE="";

create database if not exists `mona`;

USE `mona`;
SET                                     @OLD_SQL_MODE=@@SQL_MODE,
SQL_MODE='NO_AUTO_VALUE_ON_ZERO';

/*Table structure for table `adminlogin` */

DROP TABLE IF EXISTS `adminlogin`;

CREATE TABLE `adminlogin` (
  `UserName` varchar(20) default NULL,
  `password` varchar(20) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `adminlogin` */

insert into `adminlogin` (`UserName`,`password`) values ('ADMIN','ADMIN');

/*Table structure for table `groupname` */

DROP TABLE IF EXISTS `groupname`;

CREATE TABLE `groupname` (
  `groupname` varchar(50) NOT NULL default "",
  `product` varchar(50) NOT NULL default "",
  PRIMARY KEY (`groupname`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `groupname` */

insert into `groupname` (`groupname`,`product`) values ('ran','thebestsecretkey');

/*Table structure for table `login` */

DROP TABLE IF EXISTS `login`;

CREATE TABLE `login` (

```

```

`UserName` varchar(50) NOT NULL default "",
`Password` varchar(50) default "",
`Email` varchar(50) default "",
`groupname` varchar(50) default "",
PRIMARY KEY (`UserName`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `login` */

insert into `login` (`UserName`,`Password`,`Email`,`groupname`) values
('r','r','ranjith.triplen@gmail.com','ran');

/*Table structure for table `othergroup` */

DROP TABLE IF EXISTS `othergroup`;

CREATE TABLE `othergroup` (
  `UserName` varchar(25) default NULL,
  `groupname` varchar(25) default NULL,
  `Requestgroup` varchar(25) default NULL,
  `filename` varchar(50) default NULL,
  `Status` varchar(50) default NULL,
  `usercount` varchar(30) default NULL
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `othergroup` */

/*Table structure for table `startable` */

DROP TABLE IF EXISTS `startable`;

CREATE TABLE `startable` (
  `UserName` varchar(50) default NULL,
  `share1` varchar(999) default "",
  `share2` varchar(999) default "",
  `share3` varchar(999) default "",
  `share4` varchar(999) default "",
  `share5` varchar(999) default "",
  `share6` varchar(999) default ""
) ENGINE=InnoDB DEFAULT CHARSET=latin1;

/*Data for the table `startable` */

/*Table structure for table `userprofile` */

DROP TABLE IF EXISTS `userprofile`;

CREATE TABLE `userprofile` (
  `UserName` varchar(50) default NULL,
  `groupname` varchar(30) default NULL,

```

B. SCREENSHOTS

Cloud Files Secure System

CLOUD FILE MENU

File Uploading

File Downloading

File Updating

Home

Cloud Files Secure System

Group Manage - Blocking/Removing

GroupName: group1

Group Key:

submit

Cloud Files Secure System

Cloud User Registration

User Name

Secret Pwd

Retype Pwd

Email

Group Name: group1

Save

C. RESEARCH PAPER

SECURING FILES ON CLOUD STORAGE WITH GROUP KEY MANAGEMENT PROTOCOL

Aakash. S
UG Scholar
Department of IT
Sathyabama Institute of Science
and Technology
aakashsukumar@gmail.com

Jayapradeesh S L
UG Scholar
Department of IT
Sathyabama Institute of
Science and Technology
jayapradeesh481@gmail.com

Gowri S
Associate Professor
Department of IT
Sathyabama Institute of Science and
Technology

Abstract—Due to an exponential rise in the incidence of cyberattacks, there is a greater than ever need for enhanced Intrusion Detection Systems (IDS). In this aspect, the early classification of the attacks is greatly aided by the machine learning (ML) technique. Since there are numerous algorithms available, selecting the most suitable method was difficult. This study looked at some of the most cutting-edge intrusion detection methods and assessed their benefits and drawbacks in an effort to address this problem. Also, the best ML approach for classifying attacks is chosen from a selection of four ML methods. The effectiveness of the Collision detection system is assessed using a variety of methods that have been selected and tested. These Collision detection systems are divided into binary and multiclass attacks to verify whether the traffic has been classified as benign or an attack. The results show that the classification technique produces more accurate and consistent results. The experimental findings demonstrate that multi-classification improves performance in terms of intrusion detection, specifically allowing for a more targeted attack response.

Keywords—Cloud storage, Public key, File sharing, Group key, Encryption.

I. LITERATURE SURVEY

A. "Sequence comparison outsourcing which is fast and verifiable.". The scientific community is paying a lot of attention to techniques for securely exporting calculations that are too expensive due to the rapid development of cloud computing. Therefore, our suggested approach enables clients to efficiently employ mendacious servers. Furthermore, our approach is appropriate for real-world applications because it only re-garbles the circuit for distorted replies. A formal examination of the suggested construction is also provided. This method relies heavily on user input to correctly identify oxPTMs; otherwise, false positives and negatives happen, which is regarded as a method disadvantage.

B. "Secure outsourcing of sequence comparison". Sequence comparisons are being outsourced safely and securely in this situation, which is a way that is becoming more and more valuable, especially for clients with limited resources. One of the most crucial data outsourcing roles is verifiability. This had the effect of making our construction extremely effective regardless of the circuit and exclusively for answers with faulty patterns. In addition, we provide the formal analysis for the construction we propose. The strategies have limitations in that they do not apply to the current circumstance, in which the

definition includes insertions and deletions as well as strings of varying lengths.

C. *"Secure and private sequence comparisons"*. Here, the Wagner-Fischer method is being used. Given the comparisons' outcomes. Preparing for the naïve minimum finding methodology previously discussed can help you avoid this issue. The procedures in [4] have the drawback of presenting situations when the strings are of varied lengths, necessitating the allowance of insertions and deletions.

D. *"Towards such a practical data privacy scheme for shared installations of the Smith-Waterman genome sequence comparison algorithm," says the paper.* Comparison of the Smith-Waterman sequence: A complete examination of sequence comparison methods would require multiple texts. This led to the custom of giving these criteria some leeway. A certain percentage of the recognized critical results must be generated for some applications to accept a few missed important outcomes. Others might be willing to accept some latitude in the number of false positives as long as no significant findings are overlooked. This method is currently impractical owing to platform restrictions because it necessitates participant communication and causes large network traffic from each participant.

E. *"New algorithms for secure modular exponentiation outsourcing"* The solution used the disguising strategy, allowing sensitive data to be revealed. We present the first external algorithm for concurrent modular exponentiations that is both secure and effective. We also demonstrate the ability of both algorithms to deliver the needed security principles. Extremely time-consuming; unable to recognize deletions, translocations, or changes in copy number. The application can only be used in hotspots due to scalability issues.

II. EXISTING SYSTEM

The internet is revolutionizing large-scale challenges in the physical and life sciences by enabling the vast cooperative sharing of computer power, bandwidth, storage, and data. Once connected to such a grid, a poor computational device is no longer constrained

by its slow speed, scant local storage, and constrained bandwidth: It may take advantage of the wealth of these resources available elsewhere on the network. Without disclosing either one's data or the result of the computation on data, or the result of the computation on data, to the remote agents whose processing power is being employed. It secures outsourcing for universally applicable sequence comparison issues, which is one of its two key drawbacks. The risk of leaking sensitive information is another.

III. PROPOSED METHODOLOGY

We suggest a data-secure communication channel and secure key distribution system. Even if they work on an untrusted network, Users who have had their access to the original data files revoked will no longer be able to access them. Each member of the group can access the cloud source using the group user list in our scheme, and users whose access has been revoked are prevented from doing so once it has been. We offer a secure data-sharing solution that is impervious to attempts at collusion. Even if they are using an untrusted network, the revoked users won't be able to access the original data files anymore. To achieve safe user revocation, our system makes use of polynomial functions. Our solution efficiently supports dynamic groups because it avoids having to recompute and updates the other users' private keys when a user enters or quits the group. We provide a security study to show how secure our plan is. Our proposed system has some advantages which are listed below

- I. Power Means of Persuasion and Control
- II. More Reliable
- III. It's a more secure and efficient
- IV. Data confidentiality

This proposed system has its unique system requirements. For hardware we use the Pentium-IV system, speed is around 2.4GHZ, a hard disk of 40 GB, a monitor of 15VGA color, and RAM of 512MB. Coming to software requirements it contains windows XP operating system, for coding language we used java, IDE used here is NetBeans, and MYSQL is used for the database.

IV. SYSTEM ARCHITECTURE

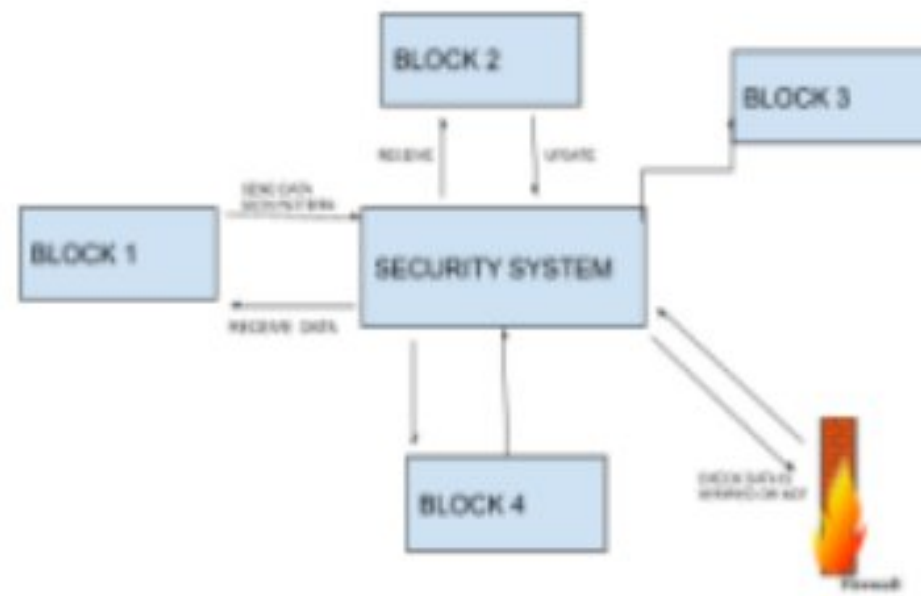


Fig 1 System Architecture

Our proposed model contains several modules they are:

- Login Module
- Registration Module
- Creation Storage and Instance
- Find collusion Module.
- Find Third-Party Module

A. LOGIN MODULE

The user must enter their enrollment password and a working contact number in order to access the app for the first time. The user is successfully entered into the app if the information they provided matches the data in the database table; otherwise, the message "login failed" is displayed, and they must enter the right information once more. For the registration of new users, a link to the registered activity is also offered.

B. REGISTRATION MODULE

A new user must register in order to access the app before logging in. The registered activity shows when the register button in the login activity is clicked. When signing up, a new user is required to enter their full name, password, and phone number. In order to confirm the password, the user must input it again in the confirm password textbox. When a user fills out all text boxes, the data is sent to the database when they click the register button, and they are then taken

to the login screen. The registered user must then login in order to access the app. All textboxes have validations applied to ensure the program operates properly. Each textbox must include the required information. During registration, there won't be any empty text boxes for name, contact, password, or confirm password. The app will notify the user that each textbox must have information in it if any of these textboxes are empty. Also, for registration to be successful. A genuine contact number with 10 digits is another need. Registration will be failed. If any of these validations are violated, the user must re-register. This is the message that the app will show if one of the fields is empty. If all of this information is correct, the user is taken to the app's login page.

C. CREATION STORAGE AND INSTANCE

The owner of the data has no further control over it once it has been submitted. Each slice's data can be encrypted into one of two possible values. Before being stored, the data in each slice can be encrypted using a variety of cryptographic techniques and encryption keys.

D. FIND COLLUSION MODULE

Receiving can determine whether collusion is occurring or not in this module by computing a distance.

E. FIND THIRD-PARTY MODULE

In this module, the receiver can also locate outside parties. Businesses that create software for a product from the original vendor are referred to as third parties.

V. RESULT AND DISCUSSION

The receiver can also find third parties in this module. A third party is a different company. We present the trusted authority for user authentication in relation to cloud data access in this essay. The key is generated by a trustworthy authority using the SHA algorithm, and it is accessible to both the owner and the user. After receiving an AES-encrypted file from the data owner, The MD-5 algorithm is used by the trusted authority module to compute the hash value. In its database, it stores keys that are used to identify system cheaters during dynamic operations. The file is sent to the CSP module by a reliable source for cloud storage. It is demonstrated that the generated vital sets possess a number of valuable characteristics that protect the secrecy of communication sessions and are vulnerable to collusion attacks by other network nodes.

VI. OUTPUT SCREENSHOT



REFERENCES:

- [1] Y. Feng, H. Ma, and X. Chen, "Efficient and verifiable outsourcing scheme of sequence comparisons," *Intell. Autom. Soft Comput.*, vol. 21, no. 1, January 2015, pp. 51-63.
- [2] P.-W. Chi and C.-L. Lei, "Audit-free cloud storage via deniable attribute-based encryption," *IEEE Trans. Cloud Comput.*, vol. 6, pp. 414-427, April 2018.
- [3] S. Salinas, X. Chen, J. Li, and P. Li, "A tutorial on secure outsourcing of large-scale computations for big data," *IEEE Access*, vol. 4, pp. 1406-1416, April 2016.
- [4] J. Wu, Y. Li, T. Wang, and Y. Ding, "CPDA: A personal privacy defragmentation cloud storage with public cloud auditing," *IEEE Access*, vol. 7, pp. 160482-160497, 2019.
- [5] S. Roy, A. K. Das, S. Chatterjee, N. Kumar, S. Chattopadhyay, and J. J. P. C. Rodrigues, "Provably secure fine-grained data access control over multiple cloud servers in mobile cloud computing-based healthcare applications," *IEEE Trans. Ind. Informat.*, vol. 15, no. 1, pp. 457-468, Jan. 2019.
- [6] Q. Xu, C. Tan, Z. Fan, W. Zhu, Y. Xiao, and F. Cheng, "Secure multi-authority data access control technique in cloud storage system based on attribute-based signcryption," *IEEE Access*, vol. 6, pp. 34051-34074, 2018.
- [7] N. T. Courtois and G. V. Bard, "Algebraic cryptanalysis of the data encryption standard," Springer, Berlin, Germany, vol. 4887, pp. 152-169, 2007.