Data Access Control in Cloud Computing Flexible and Receiver

AT
KMMIPS, Tirupathi

Submitted in partial fulfillment of the requirement for the award of Degree of
Master of Computer Applications

By

UGRANAM JAHNAVI
Regd. No: 2322157

Department of Computer Applications

KMM INSTITUTE OF POST GRADUATE STUDIES
(Affiliated to Sri Venkateswara University, Tirupati) RamiReddiPalle,
TIRUPATI-517 102

July/August- 2024

Data Access Control in Cloud Computing Flexible and Receiver

AT
KMMIPS, Tirupathi

Submitted in partial fulfillment of the requirement for the award of Degree of
Master of Computer Applications

By

UGRANAM JAHNAVI
Regd. No: 2322157

Under the Guidance of: S. MUNIKUMAR

Department of Computer Applications

KMM INSTITUTE OF POST GRADUATE STUDIES
(Affiliated to Sri Venkateswara University, Tirupati) RamiReddiPalle,
TIRUPATI-517 102

July/August- 2024

KMM INSTITUTE OF POST GRADUATE STUDIES

(Affiliated to Sri Venkateswara University, Tirupati) RamiReddiPalle, TIRUPATI – 517 102

DEPARTMENT OF COMPUTER APPLICATIONS

CERTIFICATE

REGISTERED NO: 2322157

This is to certify that the report entitled Data Access Control in Cloud Computing Flexible and Receiver submitted by U.JAHNAVI in partial fulfillment of the requirements for the award of the degree of Master of Computer Applications under our supervision and guidance in the IV semester of course, during the academic year 2023 – 2024.

HEAD

Project Guide

Attended and submitted for the University Viva –Voce Examination.

Signature (s) of the Examiners: 1._____

(External Examiner)

2._____

(Internal Examiner)

3._____

(Chairman)

Acknowledgements

The satisfaction and euphoria accompany the successful completion of task and would be incomplete without the mention of the people who made it possible, whose constant guidance and encouragement crown all the success.

I wish to express my deepest sense of gratitude and pay my sincere thanks to our Guide **S. Munikumar**, Associate Professor, and department of MCA who evinced keen interest in my effort and provided his valuable guidance throughout my project work.

I owe my gratitude to our principal, **Dr. K. Venkataramana** Principal, for his kind attention and valuable guidance given to me throughout this course.

I am also thankful to all staff members of MCA Department., and Lab Technicians for helping me complete this project work by giving valuable suggestions.

The last, but not least, I express my sincere thanks to all my friends who have supported me in the accomplishment of this project.

U. JAHNAVI

Abstract

This paper introduces EIBBE, a novel approach to enhance data access control in cloud computing through broadcast encryption. Addressing the challenge of accommodating collaboration, EIBBE leverages identity-based cryptosystems. It introduces the concept of flexible data access control with receiver extension, enabling authorized users to expand the receiver set without re-encryption. This extension, denoted as S0, complements the existing set S, granting decryption rights to both. The data uploader has control over the maximum number

of extended receivers. The paper includes a detailed construction of EIBBE, a rigorous security analysis, and demonstrates the scheme's efficiency and feasibility.

The existing system faces challenges in achieving efficient and secure data access control in cloud computing, particularly when dealing with dynamic receiver sets and collaboration. The proposed potential solution involves the use of identity-based broadcast proxy re-encryption (IBPRE). This technique allows a cloud proxy to convert the initial ciphertext for one set of users into a new ciphertext for another set using a re-encryption key. However, drawbacks include the necessity for the data uploader to generate a re-encryption key for the cloud and the cloud's ability to re-encrypt all ciphertexts, which may not be practical.

Keywords: Broadcast Encryption, Receiver Extendable, Cloud Computing, Access Control

CONTENTS

1. Introduction

The paper addresses the challenge of enhancing data access control in cloud computing environments, focusing on accommodating collaborative scenarios. Existing approaches often struggle to efficiently manage access rights, especially when collaborators need to be added dynamically without re-encryption of data. In response, the paper introduces EIBBE, a novel scheme leveraging identity-based cryptosystems. EIBBE allows for flexible data access control through receiver extension, denoted as S0, which enables authorized users to expand the set of receivers (S) without requiring re-encryption. This capability empowers data uploaders to manage and control access dynamically while ensuring efficient and secure data sharing. The paper provides a detailed construction of EIBBE, conducts a rigorous security analysis, and demonstrates the scheme's effectiveness and feasibility in real-world cloud computing scenarios.

The main reason to select this topic is to avoid the manual work and maintenance of attendance and reports. Those works required lot of time and man power, so that we are proposing a web based application to maintain the attendance and marks reports along with we shared the student attendance and mark reports to concern parents through SMS.

1.1 Organization Profile

KMMIPS is an educational institute established in 2001 by a group of retired civil servant.
This educational institute which serves for the betterment of the students.
Initially this college offers MCA course.

KMM Institute of Post Graduate Studies:

- Provides good higher-level educational services.
- Develop good communication skills to students.
- Provide good managerial skills MCA students.

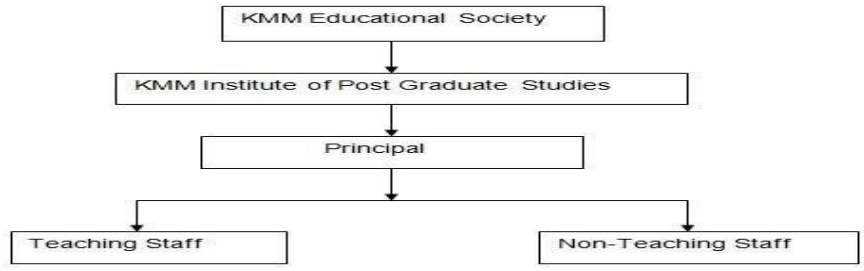
"Sathyameva jayathe" -truth always triumph is the motto of our KMM social and educational development society, Tirupati.

Nothing is permanent except change. The ongoing advances in computer communications technology continue to have profound effect on the way people work and play. Both the technology itself and the expectations of the people who use it are altering the features of the information system that analysis, design and the widespread deployment of information systems in changing the very nature of the society in which the systems are used. The development of the information systems has played a dominant role in evolution of information economy.

Kmm institute of postgraduate studies very popularly known as kmmips has emerged as a major technological institute managed by kmm social and educational development society, tirupati. The kmm society has taken the lead role to establish the institute in academic year 2001-2002. Sprawled over an area of 25 acres, permanent infrastructural facilities are Being developed near tirupati-madanapalli state highway at ramireddipalle. Kmmips has already secured the approval from the all india council of technical education (aicte), new delhi, and government of Andhra Pradesh and is affiliated to sri Venkateshwara university, tirupati.

Kmmips offers admission into the professional course mca with annual intake of 180 seats. The institution is governed by the chairman sri s. srinivasulu garu, retired irs (Indian revenue service) officer with the support of an advisory body consisting of the eminent personalities form different fields.

1.2 Organization chart



1.3 Project Report Layout

This project consists of nine chapters with an overview of computerization of Data Access Control in Cloud Computing Flexible and Receiver

Chapter 1 Introduction deals with the overview of the organization and its administrative hierarchy.

Chapter 2 Genesis of the Study outlines the Existing problem, the solution proposed, methodology used and looks at how the preliminary investigation is carried out, what are all its scope and objectives and limitations of existing manual system.

Chapter 3 Feasibility Study is used to test the feasibility of the project i.e., Operational, Technical and Economical feasibilities of the system.

Chapter 4 System Analysis deals with software requirement analysis. The various entities and their relationships are discussed using E-R Diagrams.

Chapter 5 System Requirements explains about the various hardware and software requirements and their features.

Chapter 6 System Design contains the description of all the tables and its attributes, design principles, HIPO charts and user interface design.

Chapter 7 System Testing presents software-testing strategies and techniques like white box testing, black box testing.

Chapter 8 Implementation gives the software and hardware implementation details of the system.

Chapter 9 Conclusion.

Appendix contains User Manual, Test Screens and Reports, Base Paper, Conference Paper, Journal Paper.

Bibliography gives detailed information of references i.e., books, sites, guides etc.

2. Genesis of Study

This chapter deals with the study of the existing system and describes the need for the proposed system to overcome the drawbacks in the existing system. It also specifies the objectives, scope of the system and also the methodology for the system development.

Hence, the genesis of the study clearly depicts the factors regarding the beginning of the existing system and its extension to the proposed system and it includes the following.

2.1 Aim

The aim of data access control in cloud computing is to develop advanced algorithms capable of accurately identifying and flagging manipulated or synthetic images. By leveraging machine learning techniques, this initiative seeks to protect the integrity of visual media, enhance trust in digital content, and prevent the spread of misinformation. The ultimate goal is to create a robust system that can be easily integrated into various platforms, providing real-time verification and ensuring the authenticity of images shared online.

2.2 Problem Description

The paper addresses the challenge of enhancing data access control in cloud computing environments, focusing on accommodating collaborative scenarios. Existing approaches often struggle to efficiently manage access rights, especially when collaborators need to be added dynamically without re-encryption of data. In response, the paper introduces EIBBE, a novel scheme leveraging identity-based cryptosystems. EIBBE allows for flexible data access control through receiver extension, denoted as S0, which enables authorized users to expand the set of receivers (S) without requiring re-encryption. This capability empowers data uploaders to manage and control access dynamically while ensuring efficient and secure data sharing. The paper provides a detailed construction of EIBBE, conducts a rigorous security analysis, and demonstrates the scheme's effectiveness and feasibility in real-world cloud computing scenarios.

2.2.1 Objective of the Project:

The objective of this paper is to introduce EIBBE, a novel approach for enhancing data access control in cloud computing using broadcast encryption. EIBBE addresses collaboration challenges by leveraging identity-based cryptosystems, allowing flexible data access control with receiver extension. This extension, denoted as S0, enables authorized users to expand the receiver set without re-encryption, while the data uploader retains control over the maximum number of extended receivers. The paper provides a detailed construction of EIBBE, conducts a rigorous security analysis, and demonstrates the scheme's efficiency and feasibility in real-world applications.

2.2.2 Scope**:**

The scope of this paper encompasses the introduction and detailed construction of EIBBE, a novel broadcast encryption approach designed to enhance data access control in cloud computing environments. By leveraging identity-based cryptosystems, EIBBE addresses the challenge of accommodating collaboration by introducing flexible data access control with receiver extension. This extension, denoted as S0, allows authorized users to expand the receiver set without the need for re-encryption, under the control of the data uploader who sets limits on the number of extended receivers. The paper includes a rigorous security analysis and demonstrates the efficiency and feasibility of the proposed scheme, highlighting its potential impact on secure data sharing in cloud-based collaboration scenarios.

2.2.2 Project Introduction:

In the rapidly evolving landscape of cloud computing, ensuring robust data security and flexible access control mechanisms is paramount. Traditional approaches often face challenges when accommodating dynamic collaboration scenarios where users need to securely share and access sensitive information. Addressing these challenges, this paper introduces a novel approach named EIBBE (Enhanced Identity-Based Broadcast Encryption). EIBBE leverages identity-based cryptosystems to facilitate efficient and secure data access control in cloud environments.

One of the key innovations of EIBBE is its capability to support flexible data access control with receiver extension. This feature allows authorized users to dynamically expand the receiver set without the need for re-encryption, thereby enhancing the collaborative nature of cloud-based data sharing. This extension, denoted as S0, complements the existing set S, enabling decryption rights for both sets while maintaining control over the maximum number of extended receivers.

2.3 Literature survey

2.3.1 Related Work:

[1] Gentry, C. (2003). Identity-based encryption from the Weil pairing. Gentry's paper introduces a groundbreaking approach to identity-based encryption (IBE) using the Weil pairing. This method allows cryptographic keys to be generated directly from user identities, eliminating the need for a traditional public key infrastructure (PKI). By leveraging bilinear maps, the scheme enables efficient and scalable encryption where messages can be securely sent to any arbitrary identity without prior key exchange. This work lays the theoretical foundation for practical implementations of IBE systems, influencing the development of secure communication protocols in distributed environments.

[2] Boneh, D., & Franklin, M. (2001). Identity-based encryption from the Weil pairing. Boneh and Franklin present a seminal paper on identity-based encryption utilizing the Weil pairing. Their approach revolutionizes cryptographic key management by allowing public keys to be derived directly from user identities. This eliminates the overhead associated with traditional PKI systems, offering a scalable solution for secure communication and access control in large-scale networks. The paper provides both theoretical insights into bilinear pairings and practical applications of their encryption scheme.

[3] Waters, B. (2005). Efficient identity-based encryption without random oracles. Waters contributes an efficient identity-based encryption scheme that does not rely on random oracles, addressing previous theoretical limitations. This advancement enhances both the security and practicality of identity-based encryption in real-world applications. The paper includes a detailed construction of the scheme and a rigorous security analysis, demonstrating

its effectiveness compared to earlier proposals. Waters' work significantly advances the field by providing a more robust framework for secure communication and access management.

[4] Shamir, A. (1984). Identity-based cryptosystems and signature schemes. Shamir's paper introduces the concept of identity-based cryptosystems, where cryptographic keys and signatures are derived from user identities rather than traditional key pairs. This foundational work lays the groundwork for modern identity-based encryption and digital signature schemes, influencing subsequent research in public-key cryptography. Shamir's innovative approach paved the way for new paradigms in secure communication and authentication based on user identities.

[5] Boneh, D., & Franklin, M. (2001). Identity-based encryption from the Weil pairing and its applications. Building on their previous work, Boneh and Franklin explore practical applications and theoretical implications of identity-based encryption using bilinear pairings. They demonstrate how their scheme can be effectively applied to various cryptographic tasks such as secure email communication and fine-grained access control. The paper consolidates the theoretical foundation and practical implementations of identity-based encryption, highlighting its versatility and robustness in cryptographic protocols.

[6] Gentry, C., & Silverberg, A. (2002). Hierarchical ID-based cryptography. Gentry and Silverberg propose hierarchical identity-based cryptography, extending the basic IBE framework to support hierarchical structures within organizations. This extension allows for efficient delegation of decryption rights across organizational levels, enhancing flexibility and scalability in access control systems. Their work addresses the complexities of managing cryptographic keys in large-scale environments, offering a practical solution for secure communication and access management in hierarchical settings.

2.4 Methodology

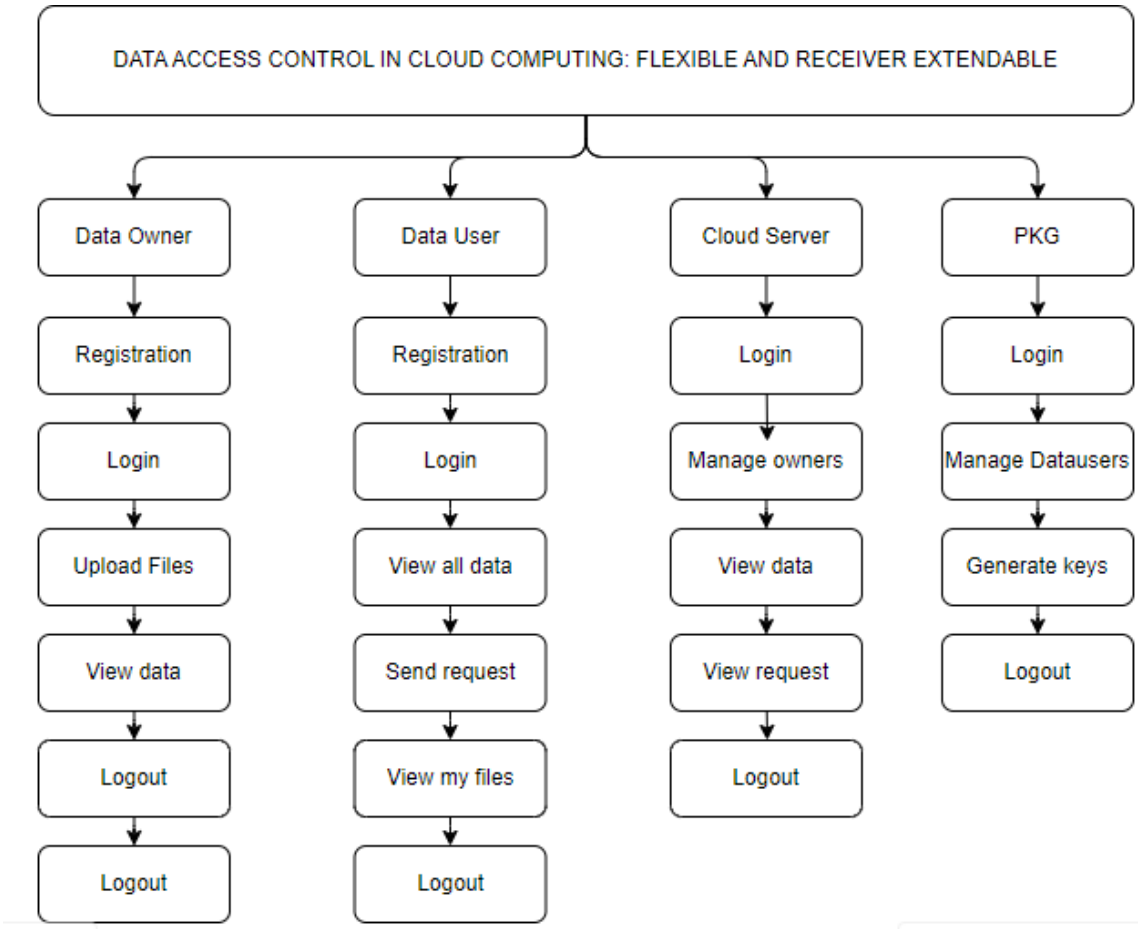
The system faces challenges in achieving efficient and secure data access control in cloud computing, particularly when dealing with dynamic receiver sets and collaboration. The

proposed potential solution involves the use of identity-based broadcast proxy re-encryption (IBPRE). This technique allows a cloud proxy to convert the initial ciphertext for one set of users into a new ciphertext for another set using a re-encryption key. However, drawbacks include the necessity for the data uploader to generate a re-encryption key for the cloud and the cloud's ability to re-encrypt all ciphertexts, which may not be practical. Additionally, generating a re-encryption key requires knowledge of the new receivers' identities, posing difficulties in scenarios where this information is determined externally, such as by a company. Existing methods in the literature do not adequately address these challenges.

2.5 Proposed System

The proposed system, called EIBBE (Extended Identity-Based Broadcast Encryption), addresses the limitations of existing broadcast encryption systems in cloud computing. Leveraging the advantages of identity-based cryptosystems, EIBBE introduces a flexible data access control mechanism that supports receiver extension without re-encryption. Authorized users can extend the initial receiver set \(S\) in the Identity-Based Broadcast Encryption (IBBE) ciphertext by adding a new set \(S_0\), allowing both sets to access the data seamlessly. The data uploader has control over the maximum number of extended receivers, providing adaptability to collaborative environments. The paper includes a detailed construction of EIBBE, a rigorous security analysis, and a demonstration of its efficiency and feasibility

PROJECT FLOW:



2.6 Objectives of Proposed system

The objective of this paper is to introduce EIBBE, a novel approach for enhancing data access control in cloud computing using broadcast encryption. EIBBE addresses collaboration challenges by leveraging identity-based cryptosystems, allowing flexible data access control with receiver extension. This extension, denoted as S0, enables authorized users to expand the receiver set without re-encryption, while the data uploader retains control over the maximum number of extended receivers. The paper provides a detailed construction of EIBBE, conducts a rigorous security analysis, and demonstrates the scheme's efficiency and feasibility in real-world applications.

2.7 Advantages of Proposed System

The proposed solution using identity-based broadcast proxy re-encryption (IBPRE) has several disadvantages:

1. Limited Control Over Re-Encryption: The data uploader has to generate a re-encryption key for the cloud, but this grants the cloud the ability to re-encrypt all ciphertexts. This lack of granularity in control may pose security and privacy concerns.

2. Dependency on New Receiver Set Information: Generating a re-encryption key requires knowledge of the set \(S_0\) containing the identities of new receivers. This may be challenging in situations where the new receiver set is determined externally, making it impractical.

. Potential Privacy Concerns: Allowing the cloud to re-encrypt all ciphertexts poses potential privacy risks, as it may have access to sensitive information that the data uploader might want to restrict.

4. Complexity in Implementation: The described approach involves a complex process of generating re-encryption keys and managing different sets of users, potentially making it more challenging to implement and maintain.

5. Incompatibility with Company-Defined Sets: The solution may face difficulties in scenarios where the new receiver set is determined by an external entity (e.g., a company), limiting the practicality of the proposed method in certain real-world use cases.

2.8 Limitations of Proposed System

. Limited Control Over Re-Encryption: The data uploader has to generate a re-encryption key for the cloud, but this grants the cloud the ability to re-encrypt all ciphertexts. This lack of granularity in control may pose security and privacy concerns.

2. Dependency on New Receiver Set Information: Generating a re-encryption key requires knowledge of the set \(S_0\) containing the identities of new receivers. This may be challenging in situations where the new receiver set is determined externally, making it impractical.

3. Potential Privacy Concerns: Allowing the cloud to re-encrypt all ciphertexts poses potential privacy risks, as it may have access to sensitive information that the data uploader might want to restrict.

4. Complexity in Implementation: The described approach involves a complex process of generating re-encryption keys and managing different sets of users, potentially making it more challenging to implement and maintain.

5. Incompatibility with Company-Defined Sets: The solution may face difficulties in scenarios where the new receiver set is determined by an external entity (e.g., a company), limiting the practicality of the proposed method in certain real-world use cases.

3. Feasibility Study

Generally, the feasibility study is used for determining the resources, cost, benefits and whether the proposed system is feasible with respect to the organization or not. The feasibility of proposed "Data Access Control in Cloud Computing" for Investigators could be evaluated as follows. There are three types of feasibility which are equally important are:

- Operational feasibility
- Technical feasibility
- Economical feasibility

3.1. Operational Feasibility:

A feasibility study is a comprehensive evaluation of a proposed project that evaluates all factors critical to its success in order to assess its likelihood of success. Business success can be defined primarily in terms of ROI, which is the number of profits that will be generated by the project. In a feasibility study, a proposed plan or project is evaluated for its practicality. As part of a feasibility study, a project or venture is evaluated for its viability in order to determine whether it will be successful. "It is an iterative approach that separates the unlabeled dataset into k distinct clusters, each of which contains just one dataset and shares a set of characteristics." It provides a straightforward approach to categories the groups in the unlabeled dataset on its own without the requirement for any training. It also enables us to cluster the data into multiple groups.

3.2 Technical Feasibility:

Technical feasibility deals with the existing technology, software & hardware requirements for the proposed system. The proposed system "Data Access Contol in Cloud Computing" is planned apache tomcat can be used to deploy and run the system. The backend database is MySQL which is an open source and easily be managed with minimum knowledge of SQL. The work for the project can be done with current equipment, existing software technology and with available personnel itself. Hence the proposed system is technically feasible.

3.3 Economical Feasibility:

This method is most frequently used for evaluating the effectiveness of a system. In this project “Data Access Control in Cloud Computing”, the development of the system required apache tomcat and above, MYSQL which are freely available and minimum memory. So, the system can be developed and can run with the current equipment, with existing software technology. Since the required Hardware and software for developing the system is already available in the organization, it does not cost much for developing the proposed system. Thus, this project is economically feasible.

4. System Analysis

System analysis is an important activity that takes place when we are building a new system or changing existing one. Analysis helps to understand the existing system and the requirements necessary for building the new system. If there is no existing system, then analysis defines only the requirements.

One of the most important factors in system analysis is to understand the system and its problems. A good understanding of the system enables designer to identify and correct problems.

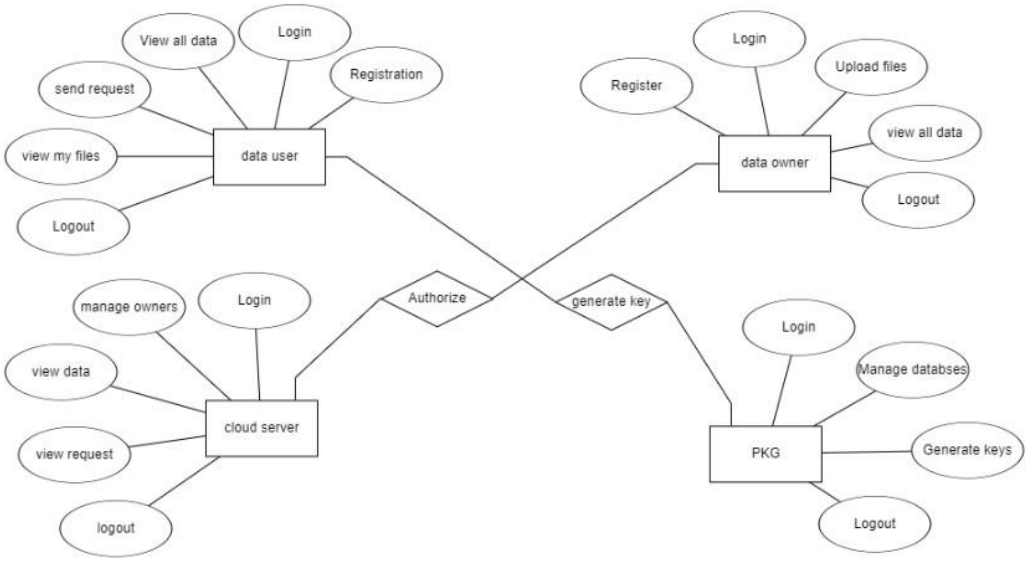
4.1 Entity-Relationship Diagrams

The Entity-Relationship Diagram depicts a relationship between data objects. The ERD is the notation that is used to conduct the data modeling activity. The attributes of each data object noted in the ERD can be described using a data object description.

At first a set of primary components are identified for ERD i.e., Data objects, Attributes, Relationships and Various type indicators. Data objects are represented by labeled rectangles. Relationships are indicated with labeled lines connecting objects.

Data modeling and the entity-relationship diagram provide the analyst with a concise notation for examining data within the context of data processing application.

The below diagram explains about the various database tables and the relation among those tables almost all the tables have one-to-many relationship. By using the exam DB Database, we can select the information of each record from the database, which can be useful for entering various details regarding students, exams, results.



4.2 Data Flow Diagram (DFD)

A Data Flow Diagram (DFD) is a traditional way to visualize the information flows within a system. A neat and clear DFD can depict a good amount of the system requirements graphically. It can be manual, automated, or a combination of both. It shows how information enters and leaves the system, what changes the information and where information is stored. The purpose of a DFD is to show the scope and boundaries of a system as a whole. It may be used as a communications tool between a systems analyst and any person who plays a part in the system that acts as the starting point for redesigning a system.

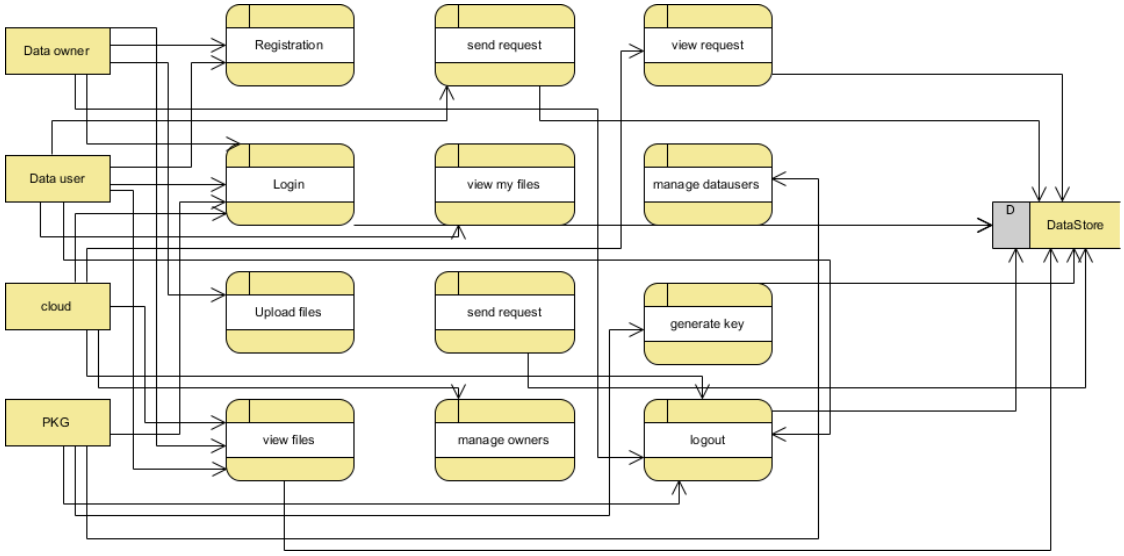
Logical Flow Diagrams:

As implementation-independent view of the system, focusing on the flow of data between processes without regard for the specific devices, storage location and people in the system.

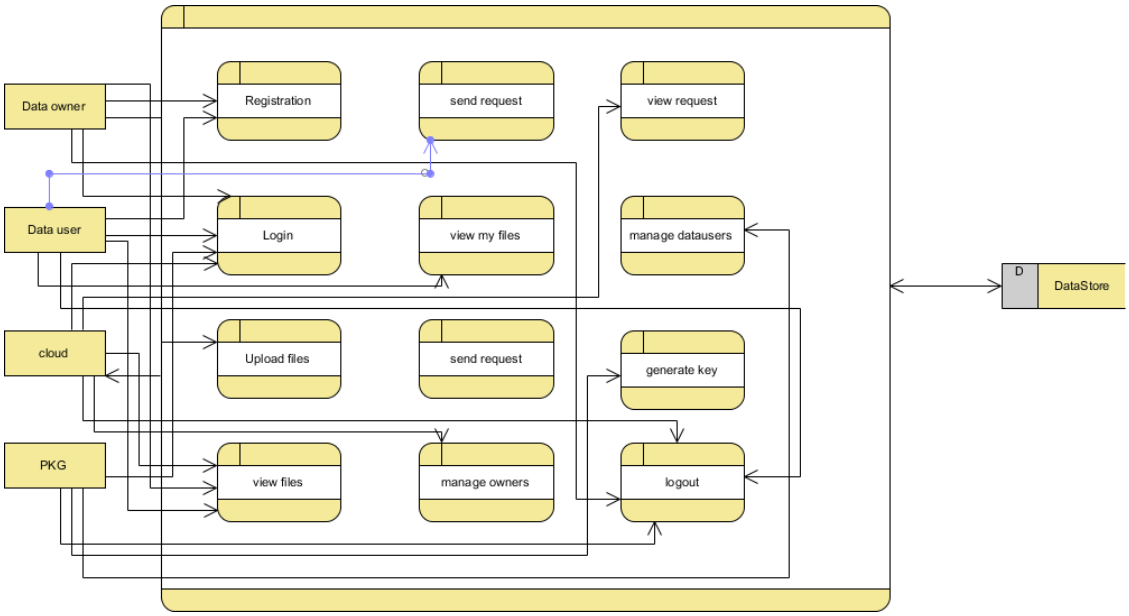
At level 0 DFD, also called as the context diagram, represents the entire system as a single module with input and output data indicated y incoming and outgoing arrows respectively.

A level 1 DFD, also called as top-level DFD, represent the system with major modules and data stores. The other levels will show each module in the top-level DFD in a more detailed fashion.

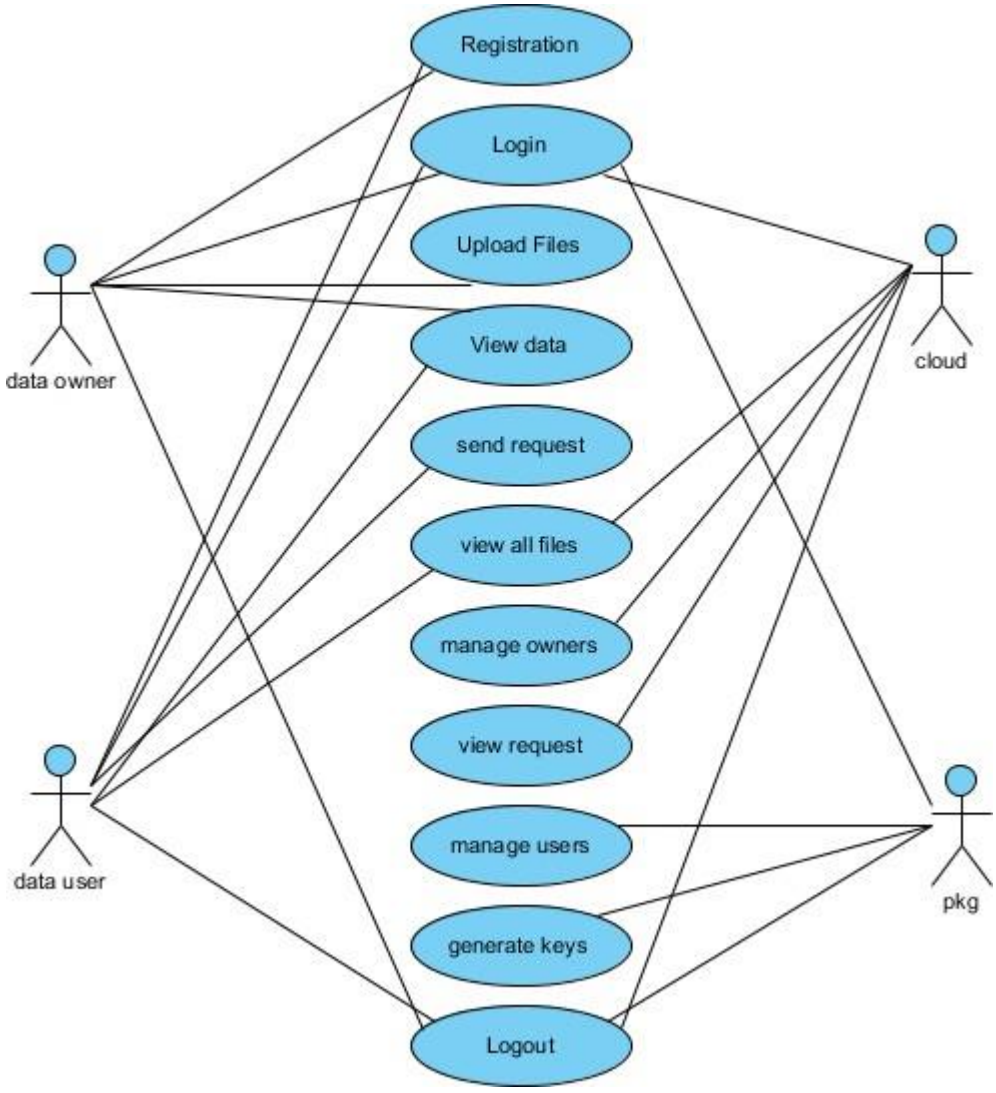
Level 1 Diagram:



Level 2 Diagram:



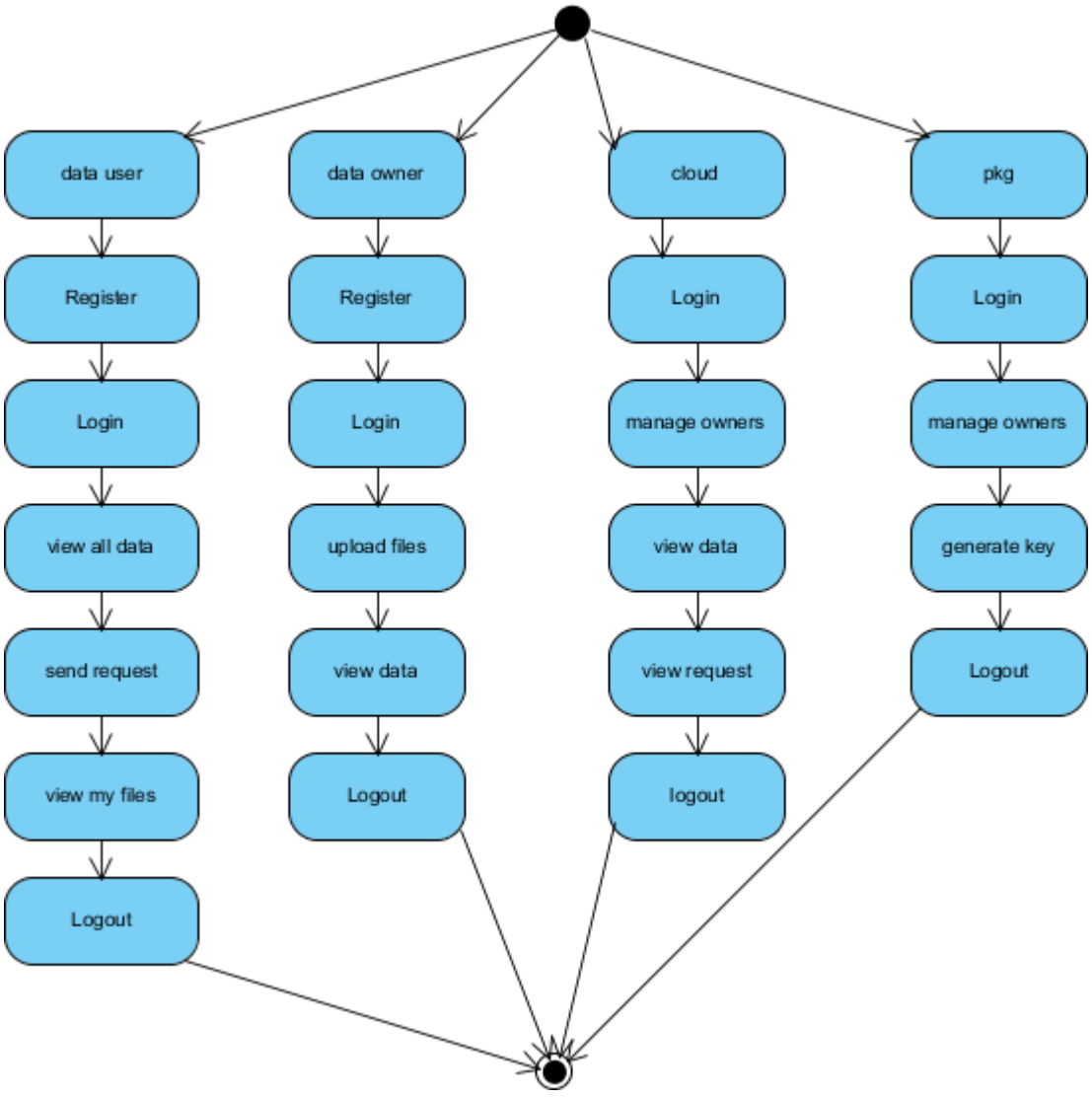
4.3 Use case Diagrams



A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

4.4 Activity Diagram:

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modelling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.



4.5 Data Dictionary:

A Data dictionary as the name implies, is a repository of information about data. In some database systems, the stored definitions of data (called schemas) provide all necessary data dictionary information.

In others, the data dictionary is supplementary. The information in the data dictionary is about types of data and uses of data.

Data dictionary is used:

- To manage the details in large system.
- To communicate a common meaning for all system developers.
- To document the features of the system.
- To facilitate analysis of the details in order to evaluate characteristics and determine where system changes to be made.

Component may be a data field or user variable used in the program.

Field Names	Description
Username	Admin Username
Gender	Admin Gender
Email Id	Admin Email Id
Mobile No	Admin Mobile No
Address	Admin Address
Password	Admin Password
Whatc	What is c-language
Whyc	Why use c language
Pointer	What is pointers in c
Whatp	What is python
Whyp	Why use python
Advp	Advantages of python

5. System Requirements

System requirements gives the idea about what are the necessary things that are needed for proposed system, which plays very important role in development of any system. This chapter deals with what are hardware components that are needed for the system, application software that are required for the development of the system.

The environment deals with the features of software JSP is used as the front-end tool and MY SQL as a backend. Front end tools help to visualize the system through naked eyes while back end helps in activities which are unseen to the end user.

5.1 : Hardware Requirements

Processor	- I3/Intel Processor
Hard Disk	- 160GB
Key Board	- Standard Windows Keyboard
Mouse	- Two or Three Button Mouse
Monitor	- SVGA
RAM	- 8GB

5.2 : Software Requirements

- Performance: Operating System : Windows 7/8/10
- Server side Script : HTML, CSS, Bootstrap & JS
- Programming Language : Python
- Libraries : Flask, Pandas, Mysql.connector, Os, Smtplib, Numpy
- IDE/Workbench : PyCharm
- Technology : Python 3.6+
- Server Deployment : Xampp Server

- 5.3: Features of Software
- 5.3.1 Features in CSS

CSS is executed much faster than those scripts which are written in other languages such as JSP and ASP. PHP uses its own memory, so the server workload and loading time is automatically reduced, which results in faster processing speed and better performance.

Open Source:

CSS source code and software are freely available on the web. You can develop all the versions of CSS according to your requirement without paying any cost. All its components are free to download and use.

Familiarity with syntax:

CSS has easily understandable syntax. Programmers are comfortable coding with it.

Embedded:

JS code can be easily embedded within HTML tags and script.

Platform Independent:

CSS is available for WINDOWS, MAC, LINUX & UNIX operating system. A CSS application developed in one OS can be easily executed in other OS also.

Database Support:

CSS supports all the leading databases such as MySQL, SQLite, ODBC, etc.

Error Reporting -

CSS has predefined error reporting constants to generate an error notice or warning at runtime. E.g., E_ERROR, E_WARNING, E_STRICT, E_PARSE.

Loosely Typed Language:

CSS allows us to use a variable without declaring its datatype. It will be taken automatically at the time of execution based on the type of data it contains on its value.

Web servers Support:

CSS is compatible with almost all local servers used today like Apache, Netscape, Microsoft IIS, etc.

Security: CSS is a secure language to develop the website. It consists of multiple layers of security to prevent threads and malicious attacks.

Control:

Different programming languages require long script or code, whereas HTML can do the same work in a few lines of code. It has maximum control over the websites like you can make changes easily whenever you want.

A Helpful CSS Community:

It has a large community of developers who regularly updates documentation, tutorials, online help, and FAQs. Learning CSS from the communities is one of the significant benefits.

5.3.2 MySQL Server

MySQL is the most popular Open-Source Relational SQL Database Management System. MySQL is one of the best RDBMS being used for developing various webbased software applications.

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 PHP, 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 Enterprise Transparent Data Encryption (TDE)

MySQL Enterprise Transparent Data Encryption (TDE) enables data-at-rest encryption by encrypting the physical files of the database. Data is encrypted automatically, in real time, prior to writing to storage and decrypted when read from storage.

MySQL Enterprise Backup

MySQL Enterprise Backup reduces the risk of data loss by delivering online "Hot" backups of your databases. It supports full, incremental and partial backups, Pointin-Time Recovery and backup compression.

MySQL Enterprise High Availability

MySQL Inno DB Cluster delivers an integrated, native, HA solution for your databases. It tightly integrates MySQL Server with Group Replication, MySQL Router, and MySQL Shell, so you don’t have to rely on external tools, scripts or other components.

MySQL Workbench

It is a unified visual tool for database architects, developers, and DBAs. It provides data modeling, SQL development, database migration and comprehensive administration tools for server configuration, user administration, and much more.

5.3.3 PYTHON

Python is a general purpose, dynamic, high-level, and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and

easy to learn and provides lots of high-level data structures. Python is easy to learn yet powerful and versatile scripting language, which makes it attractive for Application Development. Python's syntax and dynamic typing with its interpreted nature make it an ideal language for scripting and rapid application development. Python supports multiple programming pattern, including object-oriented, imperative, and functional or procedural programming styles. Python is not intended to work in a particular area, such as web programming. That is why it is known as multipurpose programming language because it can be used with web, enterprise, 3D CAD, etc. We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable. Python makes the development and debugging fast because there is no compilation step included in Python development, and edit-test-debug cycle is very fast.

Python is –

Python provides many useful features which make it popular and valuable from the other programming languages. It supports object-oriented programming, procedural programming approaches and provides dynamic memory allocation. We have listed below a few essential features.

Easy to Learn and Use

Python is easy to learn as compared to other programming languages. Its syntax is straightforward and much the same as the English language. There is no use of the semicolon or curly-bracket, the indentation defines the code block. It is the recommended programming language for beginners.

Expressive Language

Python can perform complex tasks using a few lines of code. A simple example, the hello world program you simply type print ("Hello World"). It will take only one line to execute, while Java or C takes multiple lines.

Interpreted Language

Python is an interpreted language; it means the Python program is executed one line at a time. The advantage of being interpreted language, it makes debugging easy and portable.

Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, UNIX, and Macintosh, etc. So, we can say that Python is a portable language. It enables programmers to develop the software for several competing platforms by writing a program only once.

Free and Open Source

Python is freely available for everyone. It is freely available on its official website www.python.org

. It has a large community across the world that is dedicatedly working towards make new python modules and functions. Anyone can contribute to the Python community. The open-source means, "Anyone can download its source code without paying any penny."

Object-Oriented Language

Python supports object-oriented language and concepts of classes and objects come into existence. It supports inheritance, polymorphism, and encapsulation, etc. The object-oriented procedure helps to programmer to write reusable code and develop applications in less code.

Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our Python code. It converts the program into byte code, and any platform can use that byte code.

Large Standard Library

It provides a vast range of libraries for the various fields such as machine learning, web developer, and also for the scripting. There are various machine learning libraries, such as Tensor flow, Pandas, Numpy, Keras, and Pytorch, etc. Django, flask, pyramids are the popular framework for Python web development.

GUI Programming Support

Graphical User Interface is used for the developing Desktop application. PyQT5, Tkinter, Kivy are the libraries which are used for developing the web application.

Integrated

It can be easily integrated with languages like C, C++, and JAVA, etc. Python runs code line by line like C, C++ Java. It makes easy to debug the code.

Embeddable

The code of the other programming language can use in the Python source code. We can use Python source code in another programming language as well. It can embed other language into our code.

Dynamic Memory Allocation

In Python, we don't need to specify the data-type of the variable. When we assign some value to the variable, it automatically allocates the memory to the variable at run time. Suppose we are assigned integer value 15 to x, then we don't need to write int x = 15. Just write x = 15.

The Python Platform

The platform module in Python provides functions that access information of the underlying platform (operating system). The Platform () method returns a single string containing as much information about the underlying platform as feasible. The output is meant to be viewable by humans rather than machines. Python is crossplatform and will work on Windows, macOS, and Linux.

Parameters

- aliased: Setting this parameter to true indicates the function to use aliases for different platforms than their common names. For example, SunOS will be reported as Solaris.
- terse: Setting this parameter to true will make the function return minimal information about the platform.

Return value

This method returns the human readable single string about the underlying platform.

5.3.4 JDBC in Python

In this section of the tutorial, we will discuss the steps to connect the python application to the database.

There are the following steps to connect a python application to our database.

1. Import MySQL. Connector module.
2. Create the connection object.
3. Create the cursor object
4. Execute the query

Creating the connection

To create a connection between the MySQL database and the python application, the connect () method of MySQL. Connector module is used.

Pass the database details like Hostname, username, and the database password in the method call. The method returns the connection.

1. Connection-Object= mysql.connector.connect(host = <hostname> , user = <username> , passwd = <password>)

Creating a cursor object

The cursor object can be defined as an abstraction specified in the Python DB-API 2.0. It facilitates us to have multiple separate working environments through the same connection to the database. We can create the cursor object by calling the 'cursor' function of the connection object. The cursor object is an important aspect of executing queries to the databases.

The syntax to create the cursor object is given below.

1. <my_cur> = conn. cursor ()

5.3.5 HTML

HTML stands for Hyper Text Markup Language, which is the most widely used language on Web to develop web pages. HTML was created by Berners-Lee in late 1991 but "HTML 2.0" was the first standard HTML specification which was published in 1995. HTML 4.01 was a major version of HTML and it was published in late 1999. Though HTML 4.01 version is widely used but currently we are having HTML-5 version which is an extension to HTML 4.01, and this version was published in 2012.

- Hypertext refers to the way in which Web pages (HTML documents) are linked together. Thus, the link available on a webpage is called Hypertext.
- As its name suggests, HTML is a Markup Language which means you use HTML to simply mark up a text document with tags that tell a Web browser how to structure it to display. Originally, HTML was developed with the intent of defining the structure of documents like headings, paragraphs, lists, and so forth to facilitate the sharing of scientific information between researchers.

These are some basic tags of html-----

<!DOCTYPE...>: This tag defines the document type and HTML version.

<html>: This tag encloses the complete HTML document and mainly comprises of document header which is represented by <head>...</head> and document body which is represented by <body>...</body> tags.

<head>: This tag represents the document's header which can keep other HTML tags like <title>, <link> etc.

<title>: Defines a title for the document.

<body>: This tag represents the document's body which keeps other HTML tags like <h1>, , <p> etc.

<h1> to <h6>: This tag represents the heading.

<p>: This tag represents a paragraph.

: Inserts a single line break.

6. System Design

6.1 Introduction

Design is the first step in the development phase for any system. It may be defined as the "process of applying various techniques and principles for the purpose of designing a device, a process, or a system".

Software design is an iterative process through which requirements are translated into a 'Blue Print' for constructing the software. Preliminary design is concerned with the transformation of requirements in to data and software architecture.

6.2 Design Principles

Basic design principles that enable the software engineer to navigate the design process are:

- The design should be traceable to the analysis model.
- The design should minimize the intellectual distance between the software and the problem, as it exists in the real world.
- The design should exhibit uniformity and integrity.
- The design should be structured to accommodate changes.
- The design is not coding, the coding is not a design.
- The design should be reviewed to minimize the conceptual errors.

6.3 Database Design

The goal of database design is to generate a set of relation schemes that allow us to store information without necessary redundancy and allows us to retrieve information easily. We can achieve optimization, ease of use in which data is stored in the form of tables and there exists a relation between or among tables.

The design objectives must be:

- To reduce redundancy.
- To arrive at loss-less join.
- To reduce the time as compared to the present system.
- To reduce the number of errors.

6.4 Normalization

Normalization of relation schema is done to eliminate insertion and deletion anomalies that exist in databases. Normalization is a step-by-step reversible process of converting given collection of relations to some desirable form in which the relations have a progressively simpler and regular structure.

The objectives of Normalization are:

- To make it feasible to represent any relation in the database.
- To obtain powerful retrieval algorithms based on a simpler collection of relational operations.
- To free relations from undesirable insertions, update and deletion dependencies.

A relation R is said to be in 1NF if all underlying domains contain atomic values only.

A relation R is said to be in 2NF if and only if it is in 1NF and every non-key attribute is non-transitively dependent on the primary key.

A relation R is said to be in 3NF if it is in 2NF and its non-key attribute is nontransitively dependent on its primary key.

All the tables that have been designed for developing this system follow 2nd Normalization form.

6.4.1 Database Tables

Registration table:

Field id	Type	Null	Key	Default	Extra
Username	Varchar (20)	Yes		Null	Auto_increment
Email Id	Varchar (30)	Yes	Pri	Null	
Password	Int (20)	Yes		Null	
Confirm Password	Varchar (50)	Yes		Null	
Password	Varchar (30)	Yes		Null	

Admin table:

Field id	Type	Null	Key	Default	Extra
Email Id	Varchar (30)	Yes		Null	
Password	Varchar (30)	Yes		Null	

6.5 Module design

User Modules:

1. **User Registration and Authentication Module:**
    - Allows new users to register by providing necessary details.
    - Secure login functionality with password encryption.
    - Implements brute force protection by blocking IP addresses after multiple failed login attempts.
2. **File Upload and Encryption Module:**
    - Facilitates secure file uploads by encrypting files during the upload process.
    - Stores encrypted files securely in the system.
3. **File Search and Request Module:**
    - Enables users to search for files by filename.
    - Allows users to request access to files from the uploader.
    - Notifies the uploader of access requests for their files.
4. **File Access Approval and Notification Module:**
    - Uploader reviews and approves file access requests.
    - Sends automated emails to requesters with the decryption key for downloading the approved files.

Admin Modules:

1. **Admin User Management Module:**
    - Allows admins to view and approve new user registrations.
    - Ensures only verified users can access the system.
2. **Admin File Monitoring Module:**

- Enables admins to monitor all uploaded files.
- Provides tools to track file transaction activities, ensuring a comprehensive overview of the file-sharing ecosystem.

3. **System Dashboard and Reporting Module:**
   - Provides a centralized dashboard for admins to view user activities, file uploads, and transactions.

6.6 ALGORITHMS:

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Advanced Encryption Standard (AES) is a specification for the encryption of electronic data established by the U.S National Institute of Standards and Technology (NIST) in 2001. AES is widely used today as it is a much stronger than DES and triple DES despite being harder to implement.

Working of the cipher :

AES performs operations on bytes of data rather than in bits. Since the block size is 128 bits, the cipher processes 128 bits (or 16 bytes) of the input data at a time.

The number of rounds depends on the key length as follows :

128 bit key – 10 rounds

192 bit key – 12 rounds

256 bit key – 14 rounds

7. System Testing

Introduction:

Testing is the process of detecting errors. Testing performs a very critical role for quality assurance and for ensuring the reliability of software. The results of testing are used later on during maintenance also.

Psychology of Testing

The aim of testing is often to demonstrate that a program works by showing that it has no errors. The basic purpose of testing phase is to detect the errors that may be present in the program. Testing is the process of executing a program with the intent of finding errors.

Testing Objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time.

- Testing is a process of executing a program with the intent of finding an error.
- A successful test is one that uncovers an as yet undiscovered error.
- A good test case is one that has a high probability of finding error, if it exists.
- The tests are inadequate to detect possibly present errors.
- The software more or less confirms to the quality and reliable standards.

Testing case design:

Functional testing

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.

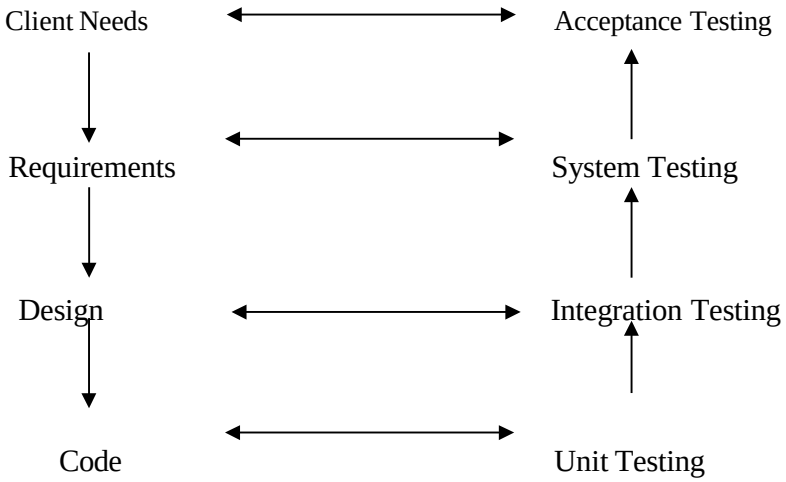
Invalid Input : identified classes of invalid input must be rejected.

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.



7.1 White Box Testing

This is a unit testing method where a unit will be taken at a time and tested thoroughly at a statement level to find the maximum possible errors. We tested step wise every piece of code, taking care that every statement in the code is executed at least once. The white box testing is also called Glass Box Testing.

We have generated a list of test cases, sample data which is used to check all possible combinations of execution paths through the code at every module level.

7.2 Black Box Testing

This testing method focuses on the functional requirements of the software. Here each module will be treated as a black box that will take some input and generate output. Output for a given set of input combinations are forwarded to other modules.

Black box testing attempts to find the following types of errors

- Incorrect or missing functions.
- Interface errors.
- Errors in data structures or external database access.
- Performance errors
- Initialization errors and termination errors.

7.3 Unit Testing:

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

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.

7.4 Integration Testing:

The goal here is to see whether the modules are integrated properly, the emphasis being on testing interfaces between modules. This testing activity can be considered as testing the design and hence the emphasis on testing module interactions.

In this project the main system is formed by integrating all the modules. When integrating all the modules, we have checked whether the integration effects working of any of the services

by giving different Combinations of inputs with which the two services run perfectly before Integration.

7.5 Test Cases

Input	Output	Result
Input data	Data access control in cloud computing flexible and receiver	Success

❑ TEST CASES MODEL BUILDING:

• S.NO	Test cases	I/O	Expected O/T	Actual O/T	P/F
1	Register users	Enter , name, email, and password	Data added successfully	Added successfully	P
2	Login users	Enter email and password	Login successfully	Login success	P
3	Login users	Enter email and password	Password not matched	Login fail	F
4	Upload file	Upload file	Data added successfully	Added successfully	P
5	Login Admin	Enter email and password	Login successfully	Login success	P

8. Implementation

Implementation is the process of converting a new or revised system design into an operational one. Apart from planning, the major tasks of preparing for implementation or education and training of users. Implementation includes following activities:

- Obtaining and installing the system hardware
- Providing user access to the system
- Creating and updating the database
- Training the users on the new system
- Documenting the system for its users ❑ Evaluating the operation and use of the system

8.1 Sample Code Implementation and output:

```
import sys
import traceback
from codeop import CommandCompiler, compile_command

__all__= ["InteractiveInterpreter", "InteractiveConsole",
          "interact", "compile_command"]

class InteractiveInterpreter:
    """Base class for InteractiveConsole.

    This class deals with parsing and interpreter state (the user's
    namespace); it doesn't deal with input buffering or
    prompting or input file naming (the filename is always
    passed in explicitly).

    """
def __init__(self, locals=None):
        """Constructor.
```

```
    The optional 'locals' argument specifies the dictionary
    in which code will be executed; it defaults to a newly
    created
    dictionary with key "__name__" set to "__console__" and
    key  "__doc__" set to None.
    """
    if locals is None:
       locals = {"__name__": "__console__", "__doc__": None}
    self.locals = locals
    self.compile = CommandCompiler()

  def runsource(self, source, filename="<input>",
    symbol="single"): """Compile and run some source in the
    interpreter.

    Arguments are as for

    compile_command(). One of several

    things can happen:

    1) The input is incorrect; compile_command() raised an
    exception (SyntaxError or OverflowError).  A syntax
    traceback will be printed by calling the
    showsyntaxerror() method.
    2) The input is incomplete, and more input is
    required; compile_command() returned None.
    Nothing happens.
    3) The input is complete; compile_command() returned a
    code object.  The code is executed by calling
    self.runcode() (which
    also handles run-time exceptions, except for SystemExit).
    The return value is True in case 2, False in the other cases
    (unless an exception is raised).  The return value can be
    used to
    decide whether to use sys.ps1 or sys.ps2 to prompt the
    next line.
```

```
    """
    try:
        code = self.compile(source, filename, symbol)
    except (OverflowError, SyntaxError, ValueError):
        # Case 1
        self.showsyntaxerror(filename)
        return False

    if code is None:
        # Case 2
        return True

    # Case 3
    self.runcode(code)
    return False

def runcode(self, code):
    """Execute a code object.

    When an exception occurs, self.showtraceback() is called
    to display a traceback.  All exceptions are caught except
    SystemExit, which is reraised.

    A note about KeyboardInterrupt: this exception may
    occur elsewhere in this code, and may not always be
    caught.  The caller should be prepared to deal with it.
    """
    try:
        exec(code,
    self.locals) except
    SystemExit:
        rais
    e
    except
    :
        self.showtraceback()
```

```
ef showsyntaxerror(self, filename=None):
    """Display the syntax error that just occurred.

    This doesn't display a stack trace because there isn't one.

    If a filename is given, it is stuffed in the exception instead
    of what was there before (because Python's parser always
    uses "<string>" when reading from a string).

    The output is written by self.write(), below.

    """
    type, value, tb = sys.exc_info()
    sys.last_type = type
    sys.last_value =
    value
    sys.last_traceback =
    tb
    if filename and type is SyntaxError:
      # Work hard to stuff the correct filename in the
      exception try:
        msg, (dummy_filename, lineno, offset, line) =
      value.args except ValueError:
        # Not the format we expect; leave it alone
        pass
      else:
        # Stuff in the right filename
        value = SyntaxError(msg, (filename, lineno, offset,
        line)) sys.last_value = value
    if sys.excepthook is sys.__excepthook__:
      lines = traceback.format_exception_only(type,
      value) self.write(''.join(lines))
    else:
      # If someone has set sys.excepthook, we let that take
      precedence # over self.write
      sys.excepthook(type, value, tb)
```

```
  def showtraceback(self):
    """Display the exception that just occurred.

    We remove the first stack item because it is our own

    code. The output is written by self.write(), below.

    """
    sys.last_type, sys.last_value, last_tb = ei = sys.exc_info()
    sys.last_traceback = last_tb
    try:
      lines = traceback.format_exception(ei[0], ei[1],
      last_tb.tb_next) if sys.excepthook is sys._excepthook_:
        self.write(''.join(lines))
      else:
        # If someone has set sys.excepthook, we let that take
        precedence # over self.write
        sys.excepthook(ei[0], ei[1], last_tb)
    finally:
      last_tb = ei = None

 def write(self, data):
    """Write a string.

    The base implementation writes to sys.stderr; a subclass
    may replace this with a different implementation.

    """
    sys.stderr.write(data)

class InteractiveConsole(InteractiveInterpreter):
  """Closely emulate the behavior of the interactive Python interpreter.
```

```
  This class builds on InteractiveInterpreter and adds
  prompting using the familiar sys.ps1 and sys.ps2, and
  input buffering.

  """

  def __init__(self, locals=None,
    filename="<console>"): """Constructor.

    The optional locals argument will be passed to
    the InteractiveInterpreter base class.

    The optional filename argument should specify the
    (file)name of the input stream; it will show up in
    tracebacks.
    """
    InteractiveInterpreter.__init__(self, locals)
    self.filename = filename
    self.resetbuffer()

  def resetbuffer(self):
    """Reset the input buffer."""
    self.buffer = []

  def interact(self, banner=None, exitmsg=None):
    """Closely emulate the interactive Python
    console.
    The optional banner argument specifies the banner to
    print before the first interaction; by default it prints a
    banner
    similar to the one printed by the real Python interpreter,
    followed by the current class name in parentheses (so
    as not to confuse this with the real interpreter -- since
    it's so close!).

    The optional exitmsg argument specifies the exit message
```

```
printed when exiting. Pass the empty string to
suppress printing an exit message. If exitmsg is not
given or None, a default message is printed.

"""
try:
   sys.ps1
except AttributeError:
   sys.ps1 = ">>>
" try:
   sys.ps2
except AttributeError:
   sys.ps2 = "... "
cprt = 'Type "help", "copyright", "credits" or "license" for more
information.' if banner is None:
   self.write("Python %s on %s\n%s\n(%s)\n" %
           (sys.version, sys.platform, cprt,
            self._class__.__name__
)) elif banner:
   self.write("%s\n" %
str(banner)) more = 0
while
   1:
   try:
      if more:
         prompt =
      sys.ps2 else:
         prompt =
      sys.ps1 try:
         line = self.raw_input(prompt)
      except EOFError:
         self.write("\n")
         break
      else:
         more = self.push(line)
```

```
        except KeyboardInterrupt:
            self.write("\nKeyboardInterrupt\n")
            self.resetbuffer()
            more = 0
        if exitmsg is None:
            self.write('now exiting %s...\n' % self.__class__.
        ____________________________________________name__)
        elif exitmsg != '':
            self.write('%s\n' % exitmsg)

    def push(self, line):
        """Push a line to the interpreter.

        The line should not have a trailing newline; it may have
        internal newlines.  The line is appended to a buffer and
        the interpreter's runsource() method is called with the
        concatenated contents of the buffer as source.  If this
        indicates that the command was executed or invalid, the
        buffer is reset; otherwise, the command is incomplete,
        and the buffer is left as it was after the line was
        appended.  The return
        value is 1 if more input is required, 0 if the line was
        dealt with in some way (this is the same as
        runsource()).
        """
        self.buffer.append(line)
        source = "\n".join(self.buffer)
        more = self.runsource(source,
        self.filename) if not more:
            self.resetbuffer
        () return more

    def raw_input(self, prompt=""):
        """Write a prompt and read a line.

        The returned line does not include the trailing newline.
```

```
    When the user enters the EOF key sequence, EOFError is raised.

    The base implementation uses the built-in
    function input(); a subclass may replace this
    with a different implementation.

    """
    return input(prompt)
def interact(banner=None, readfunc=None, local=None,
  exitmsg=None): """Closely emulate the interactive Python
  interpreter.

  This is a backwards compatible interface to the
  InteractiveConsole class.  When readfunc is not specified, it
  attempts to import the readline module to enable GNU
  readline if it is available.

  Arguments (all optional, all default to None):

  banner -- passed to InteractiveConsole.interact()
  readfunc -- if not None, replaces
  InteractiveConsole.raw_input() local -- passed to
  InteractiveInterpreter._init_()
  exitmsg -- passed to InteractiveConsole.interact()

  """
  console =
  InteractiveConsole(local) if
  readfunc is not None:
    console.raw_input =
  readfunc else:
    try:
      import readline
    except ImportError:
      pass
  console.interact(banner, exitmsg)
```

```
if __name__== "__main__":
   import argparse

   parser = argparse.ArgumentParser()
   parser.add_argument('-q', action='store_true',
               help="don't print version and copyright
   messages") args = parser.parse_args()
   if args.q or
      sys.flags.quiet:
      banner = ''
   else:
      banner = None
   interact(banner)
```

9. Conclusions

In conclusion, the paper introduces EIBBE, an innovative solution for enhancing data access control in cloud computing through broadcast encryption. By leveraging identity-based cryptosystems, EIBBE addresses collaboration challenges by allowing flexible data access control with receiver extension (S0), enabling authorized users to expand the receiver set without re-encryption. The scheme empowers data uploaders with control over the maximum number of extended receivers, ensuring efficient and secure data sharing. A detailed construction of EIBBE, rigorous security analysis, and practical demonstrations underscore its effectiveness, making it a promising approach for secure and scalable data access control in cloud environments.

Appendices

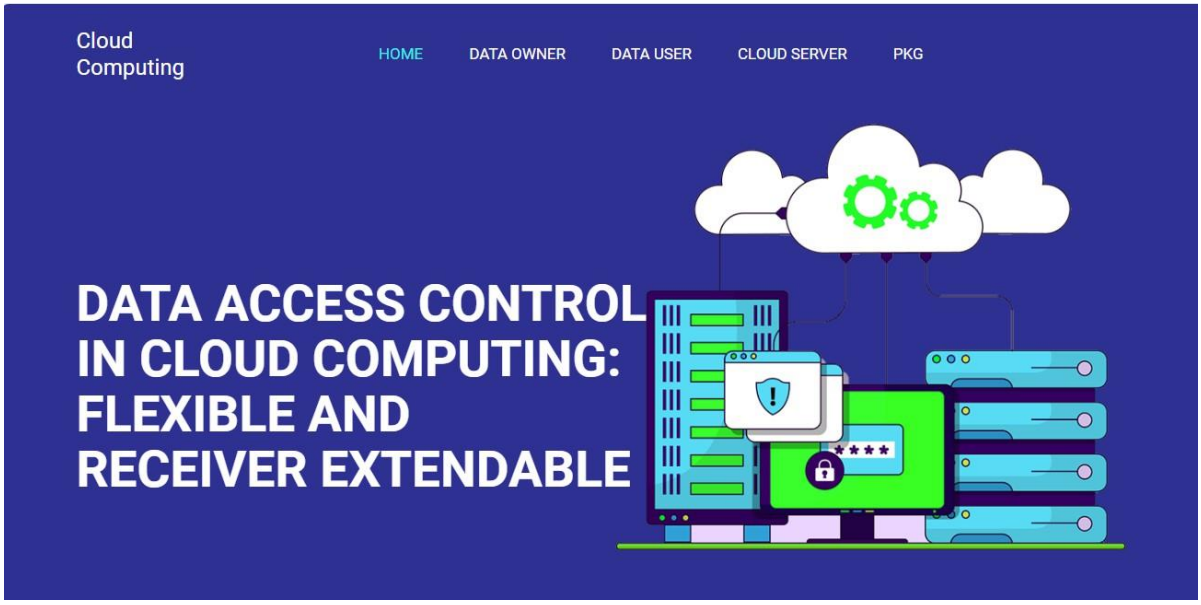
APPENDIX-A: User Manual

User Manual is the guide to the users of the system. It paves a path to the corresponding user to help him how to proceed further in the proper understanding of the system. The Interfaces of the system gets familiar to the user, based on this manual only. The first form is the login form where user has to enter his username and password, here the types of users are admin and customer.

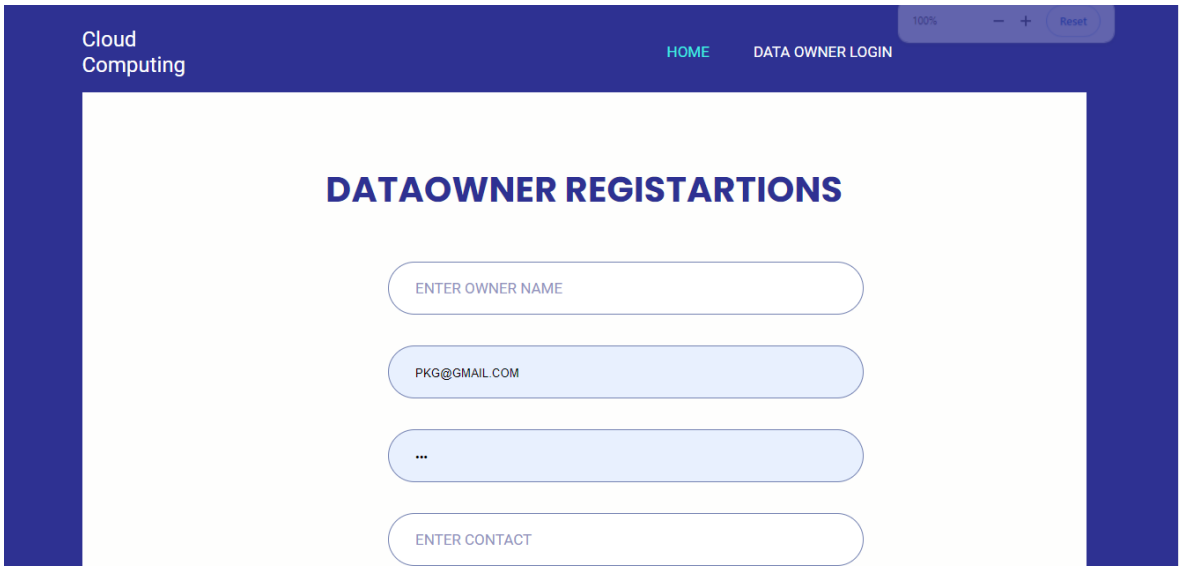
If the person has connected as an admin, he will have rights to authorize to create User-id and passwords, Update the items and prices details in database. And also have authority to provide a solution for feedback problems. If the user has been connected as customer, he/she can enter the items details, quantity details, Feedback details and can view reply reports. If the user has been connected as customer, he/she can view the hearing details.

APPENDIX-B: Test Screens

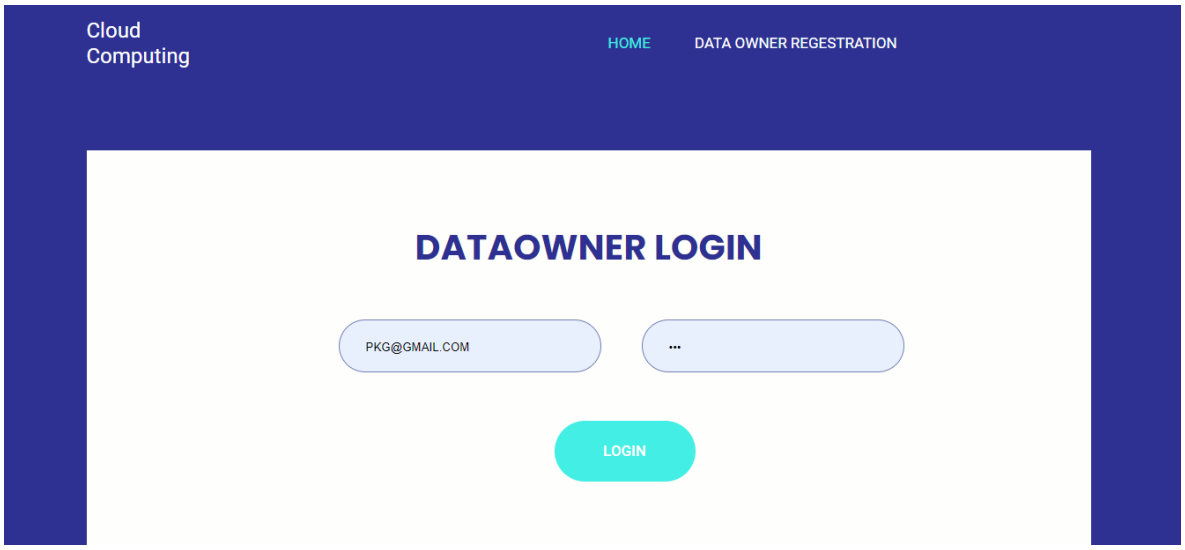
Screen1: This screen shows home page



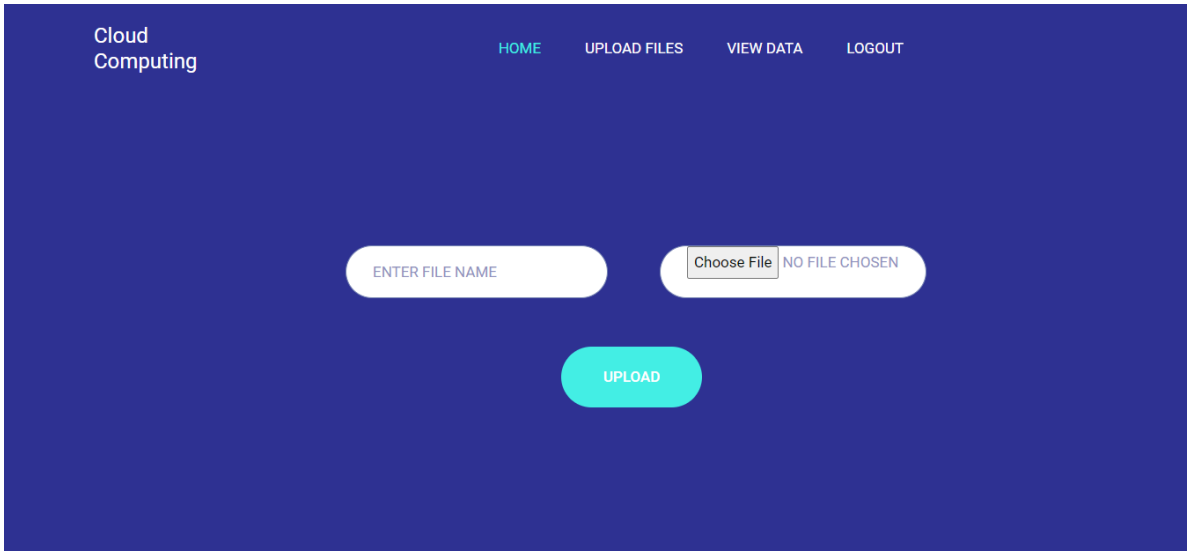
Screen 2: This screen shows Registration page



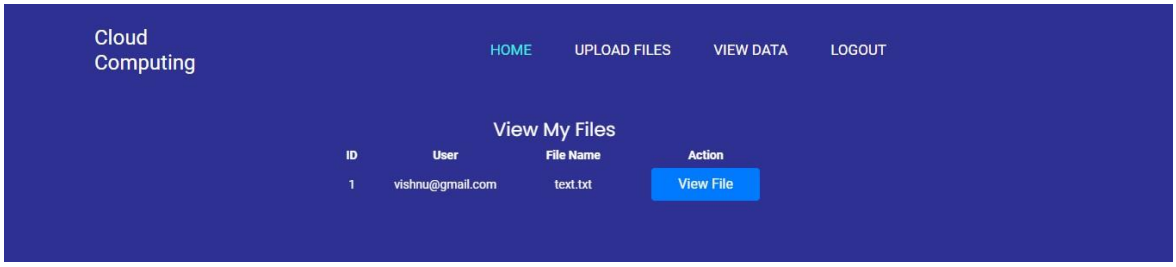
Screen 3: This screen shows Login page



Screen 4: This screen shows Upload page



Screen 5: This screen shows Result page



APPENDIX-C: Base Paper

Data Access Control in Cloud Computing: Flexible and Receiver Extendable

Jianchang Lai, Fuchun Guo, Willy Susilo, *Senior Member, IEEE*, Xinyi Huang, Peng Jiang, Futai Zhang

Abstract—Broadcast encryption provides a promising technique of data access control for specified users in cloud computing. A data uploader can generate a ciphertext for a set of chosen users such that only the intended users are able to access the data. However, with the rapidly increasing of collaboration between users, it is desired to extend the receiver set to grant decryption right for more users. The existing broadcast encryption systems cannot support receiver extension. In this paper, we for the first time take this problem into consideration and give a solution. We take the merits of identity-based cryptosystem and propose a notion of EIBBE: a flexible data access control with receiver extendable for cloud computing based on broadcast encryption. It allows the authorized user to extend the receiver set S stated in the IBBE ciphertext by adding a new receiver set S^0 without re-encryption. Both the users in S and S^0 can access the data successfully. Moreover, the data uploader determines the maximum number of extended receivers. We then give a concrete construction of EIBBE and provide a rigorous security analysis of our proposed scheme. Finally, we demonstrate the scheme's efficiency and feasibility.

Index Terms—Broadcast Encryption, Receiver Extendable, Cloud Computing, Access Control

1. INTRODUCTION

CLOUD computing has been regarded as the most promising technique for the rapidly increasing of user data. The cloud usually features powerful storage and calculation capability, flexibility and economy. A user can outsource its data to a remote cloud server to save the cost on local data management and further share them with its cooperators. When the data is sensitive such as military data or personal health records, it is desirable to prevent them from exposing. Using cryptographic tools to encrypt such kind of data before uploading them to the cloud server has been
viewed as a significant approach to protect data confidentiality. Nevertheless, as the data is encrypted into cipher data, it is inconvenient to utilize them as plaintext. Flexible data access control mechanisms are needed for encrypted data in cloud computing, such that only the users who satisfy some specified conditions are allowed to access
the encrypted data. While the core of designing such scheme is to construct a flexible key encapsulation mechanism, which encapsulates the message encryption key.

---

*This work was supported in part by the National Natural Science Foundation of China under Grant No.61902191, No.61822202, and in part by the Natural Science Foundation of Jiangsu Province under Grant No.
BK20190696.(Corresponding author: Xinyi Huang)*

- *J. Lai and X. Huang are with the School of Mathematics and Computer Science, Fujian Normal University, Fuzhou, China E-mail: jclai@finu.edu.cn, xyhuang81@gmail.com*
- *F. Guo and W. Susilo is with the Centre for Computer and Information Security Research, School of Computing and Information Technology, University of Wollongong, NSW 2522, Australia. E-mail: {fuchun,wsusilo}@uow.edu.au*
- *P. Jiang is with the School of Computer Science and Technology, Beijing institute of technology, Beijing, China. E- mail: pennyjiang0301@gmail.com*
- *F. Zhang is with the School of Computer Science and Technology, Nanjing Normal University, Nanjing, China E-mail: zhangfutai@njnu.edu.cn*

Broadcast encryption (BE) [1] provides a useful approach to achieve data access control for multi-user. In such a system, the access condition is a set of users chosen by the data
uploader who will use it to encrypt the data to be protected. Only the chosen users can retrieve the encryption key from ciphertext and then obtain plaintext. Other users who are not chosen in the encryption phase learn nothing about the encrypted data even they collude. This merit results in that BE
has been widely used in the reallife applications, such as pay- TV and radio subscriptions. The BE system designed in the identity-based setting is called identitybased broadcast
encryption (IBBE) [2], [3]. It inherits the merits of identity-based cryptosystem which eliminates the verification and management of certificates associated with users public keys. IBBE has been studied extensively and in many flavors in the literature [4], [5], [6], [7]. In IBBE, when a data uploader wants to share its data with some participants, it uses all these participants' identities to encrypt the data before outsourcing them to the cloud server. The participants can access the data via the cloud server using their private keys issued by a private key generator (PKG). The user whose identities are not specified in the encryption phase learn nothing about the data.

With the rapidly increasing of collaborations, data sharing among different participants becomes more frequent than
ever before. The authorized user might enable other users who do not satisfy the original access condition to access the same data for further collaborations in different stages. This situation was first taken into consideration in [8]. The authors proposed an extendable access control system with integrity protection based on attribute-based encryption (ABE). It
allows a data uploader to encrypt a data under an access policy P_1 such that any user satisfied P_1 is able to add a new
access policy P_2 for accessing the same data. The user satisfies either P_1 or P_2 can retrieve the encryption key. The downside of this approach is that not only the uploader has no control over the extended access policy, but also it does not be capable of the scenarios where receivers are determinate.

A motivation example can be illustrated as follow. Suppose a data supplier Alice has shared some particular industrial data M with a set S of companies who have paid for her via a cloud platform. The company X is one of the paid companies and hence it is able to access the data. Now, assume that the company X needs to further collaborate with several manufacturers (set S^0) to generating some products, and M is one of the most important data for that process. Hence, all these manufacturers are desired to access and decrypt the cipher data. Therefore, there is a need for company X to extend the receiver set stated in the ciphertext to allow manufacturers to access data via the cloud without reencrypting M. On the other hand, Alice, as the data supplier, is unwilling to see that the shared data is distributed without any restriction and would like to control the maximum
number of the extended receivers to protect her benefit.

One potential solution is to use the technique of identity- based broadcast proxy re-encryption (IBPRE) [9]. In a nutshell, it allows a proxy (cloud) with a re-encryption key to convert
the initial ciphertext for a set of user into a new one for a new
set of intended users. However, in such primitive, not only the data uploader needs to generate a re-encryption key for the cloud, but also the cloud can re-encrypt all ciphertexts
generated by the data uploader, which is not desirable in practice. Furthermore, to generate a re-encryption key, the data uploader must know the set S^0 of all new receivers' identities. It is not easy to achieve and seems impossible as the new receiver set is determined by the company X in the above example. To the best of our knowledge, no existing methods in the literature are capable of the above scenario.

Our Contribution. In this paper, we give a solution for the above problem by proposing a flexible data access control with receiver extendable. Our contributions are summarized as follows:

- We introduce a new primitive of EIBBE: Extendable Identity-Based Broadcast Encryption. This primitive is a variant of identity-based broadcast encryption (IBBE) which supports receiver extension. Any user whose identity belongs to the authorized set S can add a new receiver set S^0 for accessing the cipher data created by the data uploader. Both the users in S and S^0 can obtain the identical data.
- We give a formal definition of EIBBE and define three practical security notions for EIBBE, namely semantic security, soundness and accountability. Roughly speaking, non-authorized users learn nothing about the data hidden in the EIBBE ciphertext even they collude; the receiver set can be extended by the authorized user only; if the number of extended receivers is larger than that set by the data uploader, the ciphertext updating will be rejected. We formally define these security requirements via gamebased security models. For the semantic security, we define it via the standard indistinguishability against selective chosen plaintext attacks (IND-sCPA).
- We present a concrete construction of EIBBE that provides flexible data access control in cloud computing without reencryption. Not only the receiver set can be extended, but also the data uploader can decide the maximum number of extended receivers, and it guarantees that the extended receiver access the identical data. Prior to this work, all IBBE schemes does not support receiver extension. The proposed scheme is provably secure in the defined security models and proved to be IND-sCPA secure in the random oracle model under a q-type assumption. Finally, we give a performance analysis and demonstrate its efficiency and feasibility via experiments.

Remark. After the data user obtains the encrypted data M
and stored it locally, she definitely can share it with other users as needed and the data uploader cannot stop it. In thiswork, we do not consider this case. Otherwise, all data access control schemes are meaningless.

Organization. The rest of paper is organized as follows. In the next section, we review some related work in the literature that is close to our work. Section 3 gives the framework and formal definition of EIBBE and defines the security models for EIBBE. The concrete construction of EIBBE is given in Section 4 and the security analysis is showed in Section 5. Section 6 gives the performance evaluation of our proposed EIBBE
scheme and we conclude the paper in Section 7.

2. RELATED WORK

Broadcast encryption (BE) [1] has been introduced for enabling efficient access control on encrypted data for multi- user. It allows a data encryptor to share a single data with a group of specified users such that only the intended users are allowed to retrieve the encapsulated key and then access data. Other users learn nothing about it. Compared to
traditional public key encryption for access control in single user setting, BE greatly improves its efficiency in the multi- user setting. Identity-based broadcast encryption (IBBE) [2] eliminates the certificate appeared in the traditional public
key encryption system, and has received intensive attention in the literature [2], [10], [11], [12]. BE is particular useful for
providing access control for the scenarios where receivers are known before encryption.

Attribute-based encryption (ABE) introduced by Sahai and Waters [13] and later extended by Goyal *et al.* [14] views user identity as a set of attributes. ABE provides scalability and
finegrained access control on encrypted data. ABE is classified into key-policy ABE (KP-ABE) and ciphertext-policy ABE (CP- ABE) in terms of whether the access policy is associated with user private keys generation or encapsulated ciphertext
generation. Ciphertext decryption is possible if and only if attributes satisfy the underlying access structure (embedded in the private key for KP-ABE and in ciphertext for CP-ABE).
Due to ABE features error-tolerance property, it has been widely used to construct flexible key encapsulation
mechanisms for data access control. There are various ABE schemes developed in different models and applications to achieve different performances [15], [16], [17], [18], [19],
[20]. Nevertheless, compared to IBBE, the receivers in ABE system are fuzzy as one attribute might be used to feature several users.

Proxy re-encryption (PRE) [21] plays an important role in cloud data sharing. A date uploader can store its data in the remote cloud server in an encrypted form to protect its confidentiality. At some point in time, when the receiver is determined, the data uploader delegates a re-encryption key associated with the receiver to the cloud which is viewed as a proxy. The proxy uses the given re-encryption key to re-encrypt the initial ciphertext into a new ciphertext. This process does not reveal the data to the cloud and only the intended receiver is able to access the data. Efforts have been scheme in the identitybased setting. Whatever in the single user setting or multi-user setting, all these PRE schemes require the uploader to know the receiver(s) before generating the re-encryption key. However, in the aforementioned scenario, the new extended receiver set is determined by company X instead of the uploader. Therefore, the primitive of PRE cannot be applied to solve the above problem.

Lai *et al.* [26] focused on how to anonymously revoke a

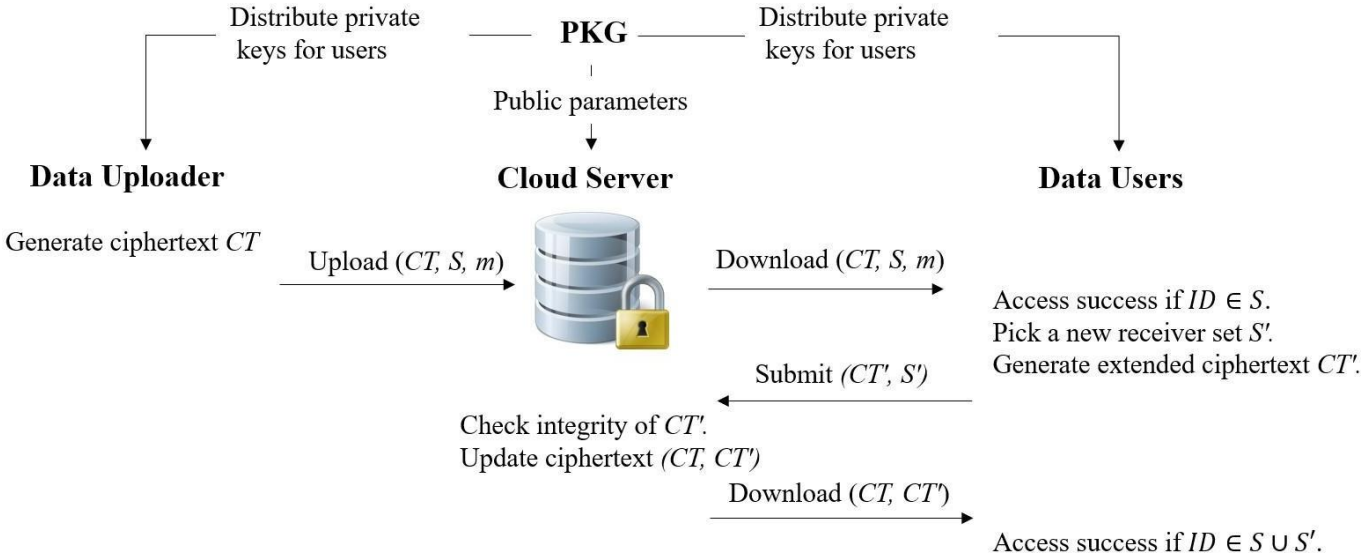


Fig. 1: Framework of EIBBE.

made to equip PRE to meet new requirements. Green and Ateniese [22] presented a PRE scheme in the identitybased
setting (IPRE) to eliminate the certificate issue. However, both PRE and IPRE are capable of a single receiver only. When the number of intended receivers are more than one, it needs to perform the corresponding system multiple times. To address this problem, Chu *et al.* [23] introduced the concept of broadcast proxy re-encryption (BPRE)(some papers also call it proxy broadcast re-encryption), which features the properties of both broadcast encryption and proxy re-encryption. In such system, the uploader generates a re-encryption key under a set of intended receivers instead of a single receiver. The new ciphertext generated under the re-encryption key can be
decrypted by all intended receivers.

One might note that in the PRE schemes, the proxy is able to encrypt either the delegator's all ciphertexts or none of
them. To address this issue, the concept of conditional proxy re-encryption (CPRE) [23], [24] was introduced. The data
uploader can decide which kind of ciphertexts are allowed to be re-encrypted when generating the re-encryption key
instead of all ciphertexts. CPRE is naturally extended to
conditional proxy broadcast re-encryption (CPBRE) [23], [25] to capture multiple receivers. Xu *et al.* [9] proposed a CPBRE
large set of receivers instead of receiver extension and proposed an identity-based broadcast encryption with
authorization scheme. A third party is allowed to update an initial ciphertext with a new receiver set anonymously. Only the user satisfying both identity sets can decrypt the
ciphertext. The work in [27] combined the technique of homomorphic encryption and proxy re-encryption to achieve access control for multi-user. However, the ciphertext
generation requires the cooperation of two servers: access control server (ACS) and data service provider (DSP). For each data user, DSP first re-encrypts the ciphertext using user's public key and sends the result to ACS. ACS then uses the public key of user to re-encrypt the received result again. For a set of receivers, the DSP and ACS need to perform the above process for each receiver.

Additionally, the ACS is able to decrypt the cipher data. From the above discussion, we note that there is no existing work can provide a solution for the aforementioned scenario.

3. SYSTEM MODEL

In this section, we describe the system model of extendable identity-based broadcast encryption (EIBBE). We first present some notations and conventions used in the rest of paper
below.

TABLE 1: Notations

Symbols	Description
λ	System security parameter
N	Maximum number of receivers
SP	System public parameter
msk	Master secret
$\mathbb{BG}$	key Bilinear
$\mathbb{G}$, $\mathbb{G}_T$	group Source
p	group
ID	Large prime number, the order of
ID_0	source groups
sk_{ID}	User identity
M	The identity of cloud server
MLE	Private key associated with identity ID
K, K'	Message to be shared
m	Message-locked encryption
CT	Message encryption key
CT'	Maximal number of extended receivers
S	Original ciphertext
S'	Extended ciphertext
$\|S\|$	Original receiver set
n	Extended receiver set
l	The number of elements in set S
$\|\mathbb{G}\|$	The number of original receivers, namely
	$n = \|S\|$
	The number of extended receivers,
	namely $l = \|S^0\|$ The size of elements in
	group G

If A is a probability or stateful algorithm, then $y \leftarrow \mathrm{A}(x)$ denotes the assignment to y of the output of A on input x. Let N denote the set of natural numbers. A function $\epsilon : \mathbb{N} \rightarrow [0,1]$ is say to be *negligible* if for every $d \in$ N, there exists a $\lambda_d \in$ N such that $\epsilon(\lambda) \leq \lambda^{-d}$ for all $\lambda > \lambda_d$.

3.1 Framework of EIBBE

The framework of EIBBE is showed in Fig. 1. It involves four kinds of entities: private key generator (PKG), cloud server, data uploader (encryptor) and data user (decryptor).

- *The PKG* is a trusted entity in the EIBBE system who manages the whole system. PKG will firstly establish the system and generates system public parameter. PKG also grants new data users capability of data decryption by generating a private

key for each user after checking the validity of user identity information. In practical applications, the PKC can be the entity which the involved user believe to be honest, such as our government, the data center of a company.

- *The cloud (server)* is assumed to be curious-buthonest(That is it will process the procedure by following algorithms honestly, but it tries to obtain the information of encrypted data.) and has powerful storage and calculation ability. It provides a platform for user data storage. The cipher data stored in the cloud server can be freely downloaded by any users. Additionally, the cloud performs integrity checking once receiving an extended ciphertext for receiver extension. If the extended ciphertext is valid, it will be added to the server.
- *The data uploader* holds the original data (message) for sharing and converts them into cipher data by specifying an authorized receiver set before uploading them to the cloud. The cipher data contains the information for restricted receiver set extension.
- *The data user* can freely obtain the cipher data that he/she interests from the cloud server. Prior to access data, each user should first register into the system and receives a private key associated with its identity from PKG. The intended (authorized) receiver is allowed to extend the receiver set by adding a new user set without encrypting data again.

3.2 Syntax of EIBBE

An EIBBE system consists of six phases: system initialization, user registration, data sharing, data access, receiver extension and integrity check. Each phase is specified by the following polynomial-time algorithms respectively.

Setup(λ,N) $\rightarrow$ (SP,msk). Taking as input a system security parameter λ and an integer N the maximal number of receivers, the setup algorithm run by the PKG outputs a master secret key msk and system public parameter SP. The msk is kept secretly and SP is made public.

KeyGen(SP,msk,ID) $\rightarrow$ sk_{ID}. Taking as input the system key pair (SP,msk) and an identity ID, the key generation algorithm run by the PKG outputs a private key sk_{ID} for ID.

Encrypt(SP,M,S) $\rightarrow$ CT. Taking as input the system public parameter SP and a set of receiver identities S, the encryption algorithm run by the data uploader outputs a tuple of (CT,K,m), where K is a message encryption key and m is the maximal number of extended receivers. Here, we always assume that m is included in CT.

When M needs to be broadcasted to users in S, the data uploader first generates (CT,K,m) by running Encrypt(SP,M,S). Then it computes a cipher data C_M of M using a symmetric encryption scheme under key K and broadcasts (CT,C_M,S). The real broadcast body is C_M and CT can be viewed as a header (In this paper, we also refer it to ciphertext).

Decrypt (SP,CT,S,sk_{ID},ID) $\rightarrow M/\perp$. Taking as input the system public parameter SP, a ciphertext CT associated with an identity set S and a private key sk_{ID} associated with identity ID. If $ID \in S$, the decryption algorithm run by the data user outputs a data encryption key K. Intuitively, the user can use K to decrypt C_M and obtains the data M. If $ID \notin S$, it outputs $\perp$.

Extend (SP,CT,S^0) $\rightarrow$ CT^0. Taking as input the public parameter SP, a ciphertext CT with an identity set S and m, and an extended identity set S^0 with $|S^0| \leq m$, the extension algorithm run by the authorized data user outputs an extended ciphertext CT^0 for $S \cup S^0$, which can be used to retrieve a key K^0.

Check (SP,CT,S,CT^0,S^0,sk_0) $\rightarrow 1/\perp$. Taking as input the system public parameter SP, a ciphertext CT associated with an identity set S, an extended ciphertext CT^0 associated with an extended identity set S^0, and cloud's private key sk_0, the integrity checking algorithm run by the cloud outputs 1 to denote that $K^0 = K$ and accept CT^0. Otherwise, it outputs $\perp$ to reject CT^0.

The *correctness* of an EIBBE scheme requires that for any (SP,msk) by calling the algorithm Setup(λ,N), sk_{ID} for ID and (CT,K) for a set S by calling the algorithm KeyGen(SP,msk,ID) and Encrypt(SP,M,S) respectively, and (CT^0,S^0) by calling Extend (SP,CT,S^0), we have, if $ID \in S$, Decrypt(SP,CT,S,sk_{ID},ID) $= K$, if $ID \in S \cup S^0$, Decrypt(SP,CT^0,$S \cup S^0$,sk_{ID},ID) $= K$, and if $K^0 = K$, Check(SP,CT,S,CT^0,S^0,sk_0) = 1.

3.3 Thread Model and Security Model

Based on the motivated example, we now present the thread models and security models for an EIBBE scheme in the cloud environment.

3.3.1 Thread models

- After the data uploader uploads the encrypted data to the cloud, the cloud and unauthorized data users might be interested in learning the content of encrypted data for some benefit.
- The data users might try to create an extended

ciphertext CT' associated with another encryption key K' such that CT' can pass the cloud's integrity checking.

- As the maximum number m of extended receivers is determined by the data uploader, the authorized data user might try to create an extended ciphertext CT' for more than m additional receivers such that CT' can pass the cloud's integrity checking.

Remark. Here we always assume that the cloud and attacker will not collude.

3.3.2 Security Models

According to the threat models, we define three security models for an EIBBE scheme: semantic security, soundness and accountability. Semantic security guarantees that only the user whose identity is specified in the encryption phase or in the extended receiver set can access the encrypted data. Soundness ensures that a valid ciphertext and an extended ciphertext will correspond to the same encapsulated key. The accountability guarantees that any extended ciphertext for more than m receivers will be rejected.

Three security models are defined via a game play between a challenger and an adversary. Suppose both the adversary and challenger are given as input N the maximum number of receivers for one encryption.

Game 1. For semantic security, we define indistinguishability against selectively chosen plaintext attacks(IND-sCPA, for short) for key encapsulation mechanism. The challenger provides adversary two encryption keys K_0, K_1 and a challenge ciphertext. The goal of the adversary is to tell apart the challenge ciphertext is generated under which key. The IND-sCPA game is defined as follows.

Init. The adversary first commits a set of identities S^* with $|S^*| < N$ that it wants to challenge.

Setup. The challenger runs the Setup algorithm to generate the system key pair (SP, msk) and sends SP to the adversary.

Phase 1. In this phase, the adversary is allowed to query private keys as needed. Upon receiving a private key query on $ID_i \in / S^*$, challenger runs the key generation algorithm KeyGen to generate a private key sk_{IDi} and responses with sk_{IDi}.

Challenge. Once the adversary decides that phase 1 is over, the challenger runs Encrypt algorithm to generate a challenge ciphertext CT^* associated with key K under S^*. It then randomly picks a bit $\mu \in \{0,1\}$, sets $K_\mu = K$ and returns (CT^*, K_0, K_1) to the adversary, where $K_{1-\mu}$ is randomly picked from the key space.

Phase 2. Same as phase 1.

Guess. Finally, the adversary outputs its guess $\mu' \in \{0,1\}$ of μ and wins the game if $\mu' = \mu$.

We defined the advantage of adversary A in wining the INDsCPA game as $\mathrm{Adv}_A = |\Pr[\mu' = \mu] - 1/2|$.

Definition 1. *An EIBBE scheme is IND-sCPA secure if for any probabilistic polynomial-time (PPT) adversary* A*, the advantage* Adv_A *is negligible.*

Game 2. For soundness, the adversary's goal is to create a valid extended ciphertext CT' with different encryption key.

Here, we present a stronger security model where the adversary is allowed to access the system master key. Therefore, there is no private key query in this model since the adversary can generate user private key by itself. The soundness game is defined below.

Setup. The challenger runs the Setup algorithm to generate system master key pair (SP, msk) and then sends (SP, msk) to the adversary.

Win. Given a valid ciphertext CT associated with an encryption key K from a honest data uploader, the adversary creates an extended ciphertext CT' related to an encryption key K' satisfying that CT' can pass cloud's integrity checking. The adversary wins the game if $K' \neq K$.

Game 3. In the accountability model, the adversary's goal is to create a ciphertext CT' for more than m additional receivers for the same data, where CT' can pass cloud's integrity checking. For simplicity, we assume that CT' is for $n + m + 1$ receivers. We define the accountability game as follows.

Setup. The challenger runs Setup algorithm to generate system master key pair (SP, msk) and sends SP to the adversary.

Win. Given a valid ciphertext CT for n receivers from an honest data uploader, where the encryption key is K and the maximum number of extended receivers is m, the adversary generates an extended ciphertext CT' with the same encryption key K for $n + m + 1$ receivers. The adversary wins the game if CT and CT' can pass the cloud's integrity checking.

4. THE PROPOSED SCHEME

In this section, we present a concrete construction of EIBBE in the cloud environment. Our proposed scheme is designed from pairing. To ensure the identical data for extended receivers, we adopt a secure message-locked encryption (MLE) [28] in the encryption key generation.

Let G and GT be two cyclic groups of prime order p. We say that a map $e: \mathrm{G} \times \mathrm{G} \rightarrow \mathrm{G}T$ is a bilinear map if it satisfies:
(1) for any $a,b \in \mathrm{Z}p$ and $g,h \in \mathrm{G}$, $e(g^a,h^b) = e(g,h)^{ab}$; (2) $e(g,h) \neq 1$ and there exists an efficient algorithm to compute $e(g,h)$. A bilinear group BG = (G,GT,e,p) is composed of the above defined objects.System Initialization

The EIBBE system is established by PKG who initializes the system public parameter. Let λ be the system security parameter and N be the maximum number of receivers, where N is an integer. Taking as input (λ,N), PKG runs the Setup algorithm to generate the system public parameter SP and master secret key msk. The algorithm is described as follows.

It first chooses a bilinear group $\mathrm{BG} = (\mathrm{G}, \mathrm{G}T, p, e)$ and a cryptographic hash function $H: \{0,1\}^* \rightarrow \mathrm{Z}p$. It then picks four generators g,h,u,v of G, two random numbers $\alpha, \beta \in \mathbb{Z}_p^*$ and computes $e(g,u)$, $g_1 = g^\alpha$. For each $i \in [1,N]$, it computes $h^{\alpha i}$ and $h^{\beta\alpha i}$. The output system public parameter SP and master secret key msk are

$$SP = \left(\mathbb{BG}, h, u, v, g_1, \left\{h^{\alpha^i}, h^{\beta\alpha^i}\right\}_{i=1}^{N}, N, e(g,u), H\right)$$

$$msk = (g, \alpha, \beta).$$

,

Any entity in the system can access SP and the master secret key msk is kept by PKG secretly which is used to user private key generation.

4.1 User Registration

In this phase, we divide the user registration into two parts. One is for the cloud's private key generation and another is for users' key generation.

For cloud. Let the identity of cloud server be ID_0. Taking as input the public parameter SP and master secret key msk. It picks a random number $t_0 \in \mathbb{Z}_p^*$ and computes the private key as

$$sk_{ID_0} = \left(g^{\frac{\beta}{\alpha+H(ID_0)}},\ g^\beta v^{t_0},\ h^{t_0},\ u^{t_0}\right).$$

Finally, PKG sends sk_{ID0} to the cloud server via a secure channel.

For users. To active access service, each user needs to register to the system and be assigned a private key associated with its identity from PKG. Upon receiving a registration request from user, PKG first checks the user's identity information(The concrete checking processes is out scope of this paper, we omit it here.).

Let a user's identity be ID. After checking the validity of ID, PKG takes as input (SP,msk,ID) and runs key generation algorithm KeyGen to generate a private key sk_{ID} for ID. The algorithm KeyGen is described as follows.

It randomly picks $t \in \mathbb{Z}_p^*$ and computes the private key as

$$sk_{ID} = (d_0, d_1, d_2, d_3) = \left(g^{\frac{1}{\alpha+H(ID)}},\ gv^t,\ h^t,\ u^t\right).$$

PKG then assigns sk_{ID} to the user via a secure channel.

Remark. Here, we stress that as the key sk_{ID0} is kept secretly by the cloud, only the cloud can verify the extended ciphertext. To achieve public verification, we can view ID_0 as a dummy identity and view sk_{ID0} as part of system public parameter SP in the system initialization phase.

4.2 Data Sharing

This phase is run by the data uploader independently, which is used to generate cipher data for sharing under some specified users. Each piece of ciphertext consists of two parts: an encapsulation ciphertext (ECT) and a suffix. The ECT is used for the encryption key retrieval and the suffix is used for receiver extension. The data is encrypted under a symmetric encryption algorithm (e.g. AES or 3-DES) by using the encryption key.

In the aforementioned scenario (in Section 1), suppose Alice wants to share a message M via the cloud platform with a set of users $S = (ID_1, ID_2, \cdots, ID_n)$ who have paid for her, and decides the maximum number of extended receivers is m with $n + m < N$. Then Alice takes as input (SP,S,M) and runs the Encrypt algorithm described as follows.

First, Alice performs the MLE to compute $s = \mathrm{MLE}(M)$ and computes the encryption key as $K = e(g,u)^s$. Then it randomly picks $k \in \mathbb{Z}_p^*$ and computes the ECT components as

$$C_1 = g_1^{-k}, \qquad C_2 = h^{k \cdot \mathrm{Q}n_{i=0}(\alpha+H(ID_i))},$$

$$C3 = ushk, \qquad C4 = vs, \qquad C5 = vk.$$

It then computes the suffix as follows. For each $j \in [1,m]$,

compute
cloud. While only the authorized user (whose identity is specified in S) can access the data successfully.

Suppose company X with identity $ID = ID_i \in S$ wants to obtain the data hidden in

which is associated with set S. The company X takes as input (SP,CT,S,ID_i,sk_{IDi}) and runs the Decrypt algorithm to retrieve the symmetric encryption key K, where $sk_{IDi} = (d_0,d_1,d_2,d_3)$.
The algorithm is described as follows. It first computes

where ID_0 is the identity of the cloud server and we always assume that ID_0 is publicly known (ID_0 can be viewed as part of the system public parameter.), and

$$p_{i,S}(\alpha) = \frac{1}{\alpha} \cdot \left(\prod_{j=0, j\neq i}^{n} (\alpha + H(ID_j)) - \prod_{j=0, j\neq i}^{n} H(ID_j) \right)$$

.

Here $h^{pi,S(\alpha)}$ is computable from the system public parameter.
With equation (1) and its private key, the company X can retrieve the encryption key by computing

$$K = \frac{e(C_3, d_1)}{e(C_4, d_3) \cdot e(C_5, d_2) \cdot e(g,h)^k} = e(g,u)^s \quad (2)$$

.

Correctness. The correctness of decryption process can be illustrated as follows. Suppose that CT is well-formed for S, then we have

$$D_j = h^{\alpha^j \cdot k \cdot \prod_{i=0}^{n}(\alpha + H(ID_i))}$$

.

The output ciphertext is

$$CT = \left((C_1, C_2, C_3, C_4, C_5),\ (D_1, D_2, \cdots, D_m),\ m\right)$$

.

We assume that CT automatically contains the identity set S. At the end of this phase, Alice uploads CT to the cloud for storage and sharing.

4.3 Data Access

Once the ciphertext is uploaded to the cloud server, it is
available to all users in the EIBBE system. Each user is allowed to freely query any ciphertext that he/she interests from theEquation (3) shows that equation (1) holds. After obtaining
the value $e(g,h)^k$, it retrieves the decryption key by computing

4)

From equation (4), we have that the decryption result is equal to the message encryption key K. It then performs the
decryption algorithm of the symmetric encryption scheme under key K to obtain M. We omit the details here.

4.4 Receiver Extension

$$CT = \left((C_1, C_2, C_3, C_4, C_5),\quad (D_1, D_2, \cdots, D_m),\quad m\right)$$

$$e(g,h)^k = \left(e\left(C_1, h^{p_{i,S}(\alpha)}\right) \cdot e\left(d_0, C_2\right)\right)^{\frac{1}{\prod_{j=0,j\neq i}^{n} H(ID_j)}}, \quad (1)$$

$$\begin{aligned}
&\left(e\left(C_1, h^{p_{i,S}(\alpha)}\right) \cdot e\left(d_0, C_2\right)\right)^{\frac{1}{\prod_{j=0,j\neq i}^{n} H(ID_j)}} \\
&= \left(e\left(g^{-\alpha k}, h^{p_{i,S}(\alpha)}\right)\right. \\
&\quad \left. \cdot \left(g^{\frac{1}{\alpha + H(ID_i)}}, h^{k \cdot \prod_{j=0}^{n}(\alpha + H(ID_j))}\right)\right)^{\frac{1}{\prod_{j=0,j\neq i}^{n} H(ID_j)}} \\
&= \left(e(g,h)^{-k \cdot \left(\prod_{j=0,j\neq i}^{n}(\alpha + H(ID_j)) - \prod_{j=0,j\neq i}^{n} H(ID_j)\right)}\right. \quad (3) \\
&\quad \left. \cdot\, e(g,h)^{k \cdot \prod_{j=0,j\neq i}^{n}(\alpha + H(ID_j))}\right)^{\frac{1}{\prod_{j=0,j\neq i}^{n} H(ID_j)}} \\
&= \left(e(g,h)^{k \cdot \prod_{j=0,j\neq i}^{n} H(ID_j)}\right)^{\frac{1}{\prod_{j=0,j\neq i}^{n} H(ID_j)}} \\
&= e(g,h)^k.
\end{aligned}$$

$$\begin{aligned}
&\frac{e(C_3, d_1)}{e(C_4, d_3) \cdot e(C_5, d_2) \cdot e(g,h)^k} \\
&= \frac{e(u^s h^k, g v^t)}{e(v^s, u^t) \cdot e(v^k, h^t) \cdot e(g,h)^k} \\
&= e(g,u)^s \\
&= K.
\end{aligned}$$

ciphertext is generated under original receiver set and the same symmetric encryption key before updating the

As one of main functionalities of EIBBE system, it allows the authorized data user to extend the receiver set such that more users can access the same data.

Suppose at some point later in time, company X needs to further share the identical message hidden in CT to a set of manufactures $S' = (ID'_1, ID'_2, \cdots, ID'_l)$ for collaboration. The company X runs algorithm Extend described below.

Let $CT = \left((C_1, C_2, C_3, C_4, C_5),\ (D_1, D_2, \cdots, D_m),\ m\right)$ associated with $S = (ID_1, ID_2, \cdots, ID_n)$, and set $S^0 = (ID'_1, ID'_2, \cdots, ID'_l) = (ID_{n+1}, ID_{n+2}, \cdots, ID_{n+l})$ with $l \leq m$ and $S \cap S^0 = \emptyset$. It first obtains s as in data sharing phase, picks a random $k^0 \in Z^*_p$, and computes

$$C'_1 = C_1^{k'},\ C'_2 = h^{k' \cdot k \cdot \prod_{i=0}^{n+l}(\alpha + H(ID_i))},$$
$$C'_3 = u^s \left(\frac{C_3}{u^s}\right)^{k'},\ C'_4 = C_4,\ C'_5 = C_5^{k'}$$

The extended ciphertext is $CT' = (C'_1, C'_2, C'_3, C'_4, C'_5)$.

Finally, company X uploads (CT^0,S^0) to the could for receiver extension. Note that in this stage, we do not re-encrypt M, but update the encapsulated ciphertext.

The correctness of extended ciphertext can be showed as follows. Let $\tilde{k} = k \cdot k^0$, we have

$$C'_1 = C_1^{k'} = \left(g_1^{-k}\right)^{k'} = g_1^{-\tilde{k}}, \quad (5)$$
$$C'_2 = h^{k' \cdot k \cdot \prod_{i=0}^{n+l}(\alpha + H(ID_i))} = h^{\tilde{k} \cdot \prod_{i=0}^{n+l}(\alpha + H(ID_i))}, \quad (6)$$
$$C'_3 = u^s \left(\frac{C_3}{u^s}\right)^{k'} = u^s \cdot \left(h^k\right)^{k'} = u^s h^{\tilde{k}}, \quad (7)$$
$$C'_4 = C_4 = v^s, \quad (8)$$
$$C'_5 = C_5^{k'} = \left(v^k\right)^{k'} = v^{\tilde{k}}. \quad (9)$$

As $l < m$, C'_2 is computable from C_2 and D_j stated in CT. If $|S^0| > m$, the value of C'_2 cannot be computed successfully. In order to compute C'_3, it has to obtain s and remove u^s from C_3 firstly which requires to know M. Therefore, the extended ciphertext CT^0 can be computed successfully by company X and CT^0 can be viewed as a valid ciphertext generated under $S \cup S^0$ and random number $\tilde{k}$ such that the manufactures in S^0 can access the identical data.

$$\tilde{K}' = \frac{e(C'_3, d_1)}{e(C'_4, d_3) \cdot e(C'_5, d_2) \cdot e(g, h)^{\beta \tilde{k}}} = e(g, u)^{\beta s}. \quad (13)$$

ciphertext. This process is completed without learning the encrypted data.

$$CT' = (C'_1, C'_2, C'_3, C'_4, C'_5)$$

Suppose $CT = \left((C_1, C_2, C_3, C_4, C_5), (D_1, D_2, \cdots, D_m), m\right)$ is associated with S^0 and is related to S. The cloud takes as input (SP,CT,CT^0,S,S^0,sk_{ID0}) and runs algorithm Check described below.

It first uses its private key sk_{ID0} to partially decrypt CT by computing

$$e(g, h)^{\beta \cdot k} = \left(e\left(C_1, h^{p_{0,S}(\alpha)}\right) \cdot e(d_0, C_2)\right)^{\frac{1}{\prod_{j=1}^{n} H(ID_j)}}, \quad (10)$$

where

$$p_{0,S}(\alpha) = \frac{\beta}{\alpha} \cdot \left(\prod_{j=1}^{n}(\alpha + H(ID_j)) - \prod_{j=1}^{n} H(ID_j)\right),$$

which is computable from the system public parameter. With equation (10) and its private key, it can compute

4.5 Integrity Check

After receiving an updating request for receiver extension, the cloud makes integrity checking to ensure the extended

$$\tilde{K} = \frac{e(C_3, d_1)}{e(C_4, d_3) \cdot e(C_5, d_2) \cdot e(g, h)^{\beta \cdot k}} = e(g, u)^{\beta s}. \quad (11)$$

It then uses sk_{ID0} to partially decrypt CT^0 by computing

$$e(g, h)^{\beta \cdot \tilde{k}} = \left(e\left(C'_1, h^{p_{0,S+S'}(\alpha)}\right) \cdot e(d_0, C'_2)\right)^{\frac{1}{\prod_{j=1}^{n+l} H(ID_j)}}. \quad (12)$$

With equation (12) and sk_{ID0}, it can compute

If $\tilde{K} = \tilde{K}^0$, the cloud outputs 1 to accept the extended ciphertext and update the corresponding information. Otherwise, it outputs 0 to reject the extended ciphertext.

Next, we show the integrity checking works. Suppose CT is well-formed for S and CT^0 is well-formed for $S \cup S^0$. We have

$$\begin{aligned}
&\left(e\left(C_1, h^{p_{0,S}(\alpha)}\right) \cdot e(d_0, C_2)\right)^{\frac{1}{\prod_{j=1}^{n} H(ID_j)}} \\
&= \left(e\left(g^{-\alpha k}, h^{p_{0,S}(\alpha)}\right)\right. \\
&\quad \left. \cdot \left(g^{\frac{\beta}{\alpha + H(ID_0)}}, h^{k \cdot \prod_{j=0}^{n}(\alpha + H(ID_j))}\right)\right)^{\frac{1}{\prod_{j=1}^{n} H(ID_j)}} \\
&= \left(e(g, h)^{-\beta k \cdot \left(\prod_{j=1}^{n}(\alpha + H(ID_j)) - \prod_{j=1}^{n} H(ID_j)\right)}\right. \\
&\quad \left. \cdot e(g, h)^{\beta k \cdot \prod_{j=1}^{n}(\alpha + H(ID_j))}\right)^{\frac{1}{\prod_{j=1}^{n} H(ID_j)}} \\
&= \left(e(g, h)^{\beta k \cdot \prod_{j=1}^{n} H(ID_j)}\right)^{\frac{1}{\prod_{j=1}^{n} H(ID_j)}} \\
&= e(g, h)^{\beta k}, \quad (14)
\end{aligned}$$

$$\begin{aligned}
&\left(e\left(C_1', h^{p_{0,S+S'}(\alpha)}\right)\cdot e\left(d_0, C_2'\right)\right)^{\frac{1}{\prod_{j=1}^{n+l} H(ID_j)}}\\
&=\left(e\left(g^{-\alpha\tilde{k}}, h^{p_{0,S+S'}(\alpha)}\right)\right.\\
&\left.\cdot\left(g^{\frac{\beta}{\alpha+H(ID_0)}}, h^{\tilde{k}\cdot\prod_{j=0}^{n+l}(\alpha+H(ID_j))}\right)\right)^{\frac{1}{\prod_{j=1}^{n+l} H(ID_j)}}\\
&=\left(e(g,h)^{-\beta\tilde{k}\cdot\left(\prod_{j=1}^{n+l}(\alpha+H(ID_j))-\prod_{j=1}^{n+l} H(ID_j)\right)}\right.\\
&\left.\cdot e(g,h)^{\beta\tilde{k}\cdot\prod_{j=1}^{n+l}(\alpha+H(ID_j))}\right)^{\frac{1}{\prod_{j=1}^{n+l} H(ID_j)}}\\
&=\left(e(g,h)^{\beta\tilde{k}\cdot\prod_{j=1}^{n+l} H(ID_j)}\right)^{\frac{1}{\prod_{j=1}^{n+l} H(ID_j)}}\\
&=e(g,h)^{\beta\tilde{k}}.
\end{aligned} \tag{15}$$

Equation (14) and (15) show equation (10) and (12) hold respectively. After obtaining the value $e(g,h)^{\beta k}$ and $e(g,h)^{\beta\tilde{k}}$, we have

$$\begin{aligned}
K' &= \frac{e(C_3, d_1)}{e(C_4, d_3)\cdot e(C_5, d_2)\cdot e(g,h)^{\beta k}}\\
&= \frac{e(u^s h^k, g^\beta v^{t_0})}{e(v^s, u^{t_0})\cdot e(v^k, h^{t_0})\cdot e(g,h)^{\beta k}}\\
&= e(g,u)^{\beta s},
\end{aligned} \tag{16}$$

$$\begin{aligned}
\tilde{K} &= \frac{e(C_3', d_1)}{e(C_4', d_3)\cdot e(C_5', d_2)\cdot e(g,h)^{\beta\tilde{k}}}\\
&= \frac{e(u^s h^{\tilde{k}}, g^\beta v^{t_0})}{e(v^s, u^{t_0})\cdot e(v^{\tilde{k}}, h^{t_0})\cdot e(g,h)^{\beta\tilde{k}}}\\
&= e(g,u)^{\beta s}\\
&= K'.
\end{aligned} \tag{17}$$

From the setting of encryption algorithm and receiver extension algorithm, if a user can decrypt CT and C_2' is generated under $S \cup S^0$, K^0 must equal to $\tilde{K}$. Otherwise, $K^0 = \tilde{K}$ holds with a negligible probability. Note that if sk_{ID0} is public, the proposed scheme also supports public verification.

5. SECURITY ANALYSIS

In this section, we give formal security proof of our proposed scheme. Before starting to analyze the security, we show a complexity assumption which the security of our proposed scheme bases on.

5.1 Complexity Assumption

The security of our proposed scheme bases on a variant of
(*f*,*g*,*F*)-GDDHE assumption [2]. We call it augment general decisional Diffie-Hellman exponent assumption and denote it as (*f*,*g*,*F*)-aGDDHE. The (*f*,*g*,*F*)-aGDDHE problem is described as follows. Let BG = (G, G$_T$, *p*, *e*) be a bilinear group, f and g are two coprime polynomials with pairwise distinct roots of
respective orders t and N. For a random $l \in [2, N]$, the given instance I is

$$g_0,\ g_0^a,\ g_0^{a^2}, \cdots,\ g_0^{a^{t-1}},\ g_0^{af(a)},\ g_0^{kaf(a)} \tag{18}$$

$$g_0^o,\ g_0^{oa},\ g_0^{oa^2}, \cdots,\ g_0^{oa^{-1}},\ g_0^{ka^{-1}},\ g_0^{sa^{-1}} \tag{19}$$

$$h_0,\ h_0^a,\ h_0^{a^2}, \cdots,\ h_0^{a^{2N}}, \tag{20}$$

$$h_0^{ba},\ h_0^{ba^2}, \cdots,\ h_0^{ba^{2N}}, \tag{21}$$

$$h_0^k,\ h_0^{ka},\ h_0^{ka^2}, \cdots,\ h_0^{ka^{l-1}}, \tag{22}$$

$$h_0^{kg(a)},\ h_0^{kag(a)}, \cdots, h_0^{ka^N g(a)} \tag{23}$$

$$u_0,\ u_0^a,\ u_0^{ab},\ u_0^s h_0^{ka^l} \tag{24}$$

as well as $Z \in G_T$ which is either equal to $e(g_0,u_0)^{sf(a)}$ or some random element in G$_T$.

An algorithm D that outputs a bit $\mu \in \{0,1\}$ has advantage in solving (*f*,*g*,*F*)-aGDDHE problem in G if

$$\left|\Pr\left[\mathcal{D}\left(\mathcal{I}, Z = e(g_0,u_0)^{sf(a)}\right) = 1\right]\right.$$
$$\left. - \Pr\left[\mathcal{D}\left(\mathcal{I}, Z \neq e(g_0,u_0)^{sf(a)}\right) = 1\right]\right| \geq \epsilon.$$

Definition 2. *We say that the* (*f*,*g*,*F*)-aGDDHE *assumption holds if no PPT algorithm has non-negligible advantage in solving the* (*f*,*g*,*F*)-aGDDHE *problem.*

5.2 Security Proof

The security of our proposed EIBBE scheme is captured by the following theorems. We stress that in our proofs, we assume the underlying message locked encryption scheme is secure.

Theorem 1 (Semantic Security). *Let H is a random oracle, the proposed EIBBE scheme is IND-sCPA secure under the* (*f*,*g*,*F*)- aGDDHE *assumption.*

Proof. Suppose there exists an adversary A who can break the proposed scheme in the IND-sCPA game with non-negligible

advantage ϵ. Then we are able to construct a simulator B to solve the (*f*,*g*,*F*)-aGDDHE problem by running A. For simplicity, we assume ID_0 is given as the system parameter and publicly known. We also assume both A and B are given as input N, the maximum number of receivers, and t the total number of private key queries and random oracle queries issued by A.

The simulator B then is given as input a bilinear group BG
= (G, G$_T$, *p*, *e*) and a (*f*,*g*,*F*)-aGDDHE instance I. Thus we have f and g are two coprime polynomials with pairwise distinct roots of respective orders t and N, and $l \in [2, N]$. It works by interacting with A as follows.

In the (*f*,*g*,*F*)-aGDDHE instance I, f and g are unitary polynomials. B sets $f(x)$ and $g(x)$ as

$$f(x) = \prod_{i=1}^{t}(x + x_i) = \sum_{j=0}^{t} a_i x^i,$$

$$g(x) = \prod_{i=t+1}^{t+N}(x + x_i),$$

with known x_i and then sets

$$f_i(x) = \frac{f(x)}{x + x_i}, i \in [1, t].$$

Init: A outputs a challenge identity set $S^* = \{ID_1^*, ID_2^*, \cdots, ID_{s^*}^*\}$ with $s^* < N$. If $N - s^* 6= l$, B aborts. Otherwise, it performs as follow.

Setup: B simulates the master key pairs as follow. It picks a random $w \in Z^*_p$, implicitly sets $\alpha = a, \beta = b, g = g_0^{f(a)}$ and computes

$$h = h_0^{\prod_{i=t+s^*+1}^{t+N}(a+x_i)}, \ g_1 = g_0^{af(a)}, \ u = u_0, \ v = g_0^{wa^{t-1}},$$

$$\begin{aligned} e(g,u) &= e\left(g_0^{f(a)}, u_0\right) \\ &= \prod_{i=0}^{t-1} e\left(\left(g_0^{a^i}\right)^{a_i}, u_0\right) \cdot e\left(\left(g_0^{a^{t-1}}\right)^{a_t}, u_0^a\right), \end{aligned}$$

which are computable from (18) and (24).

Then for each $i \in [1,N]$, compute

$$h^{\alpha^i} = h_0^{a^i \cdot \prod_{i=t+s^*+1}^{t+N}(a+x_i)}, \ h^{b\alpha^i} = h_0^{ba^i \cdot \prod_{i=t+s^*+1}^{t+N}(a+x_i)}$$

which can be computed from (20) and (21). Finally, B sends the system public parameter below to A.

$$SP = \left(\mathbb{BG}, h, u, v, g_1, \left\{h^{\alpha^i}, h^{\beta\alpha^i}\right\}_{i=1}^{N}, N, e(g,u), ID_0\right).$$

Here, the cryptography hash function H is set as a random oracle controlled by B.

Hash Query: For a hash query on ID_i (at most $t–q_E$ times, with q_E the number of private key queries), to respond to these queries, B maintains a list L of tuples (ID_i, x_i, sk_{IDi}) that contains at the beginning:

$$\{(ID_0, x_0, *)\}, \{(*, x_i, *)\}_{i=1}^{t}, \ \{(ID_i, x_i, *)\}_{i=t+1}^{t+s^*}$$

where $*$ denotes an empty entry in L. Upon receiving a hash query on ID_i, if ID_i already appears in L, B responds with the corresponding x_i. Otherwise, B sets $H(ID_i) = x_i$ and adds the tuple $(ID_i, x_i, *)$ to L.

Phase 1: A is allowed to adaptively issue private key queries. For a private key query on $ID_i \notin S^*$. If A has already issued a private key query on ID_i, B responds with the corresponding sk_{IDi} in L. Otherwise, we have $H(ID_i) = x_i$ (If ID_i has not been submitted to the hash query, B runs the hash query on ID_i to get x_i). Then B simulates the private key sk_{IDi} as follows.

- If $ID_i = ID_0$, randomly choose $t' \in \mathbb{Z}_p^*$ and implicitly set $t_0 = -\frac{1}{w}ab + t'$, compute

$$d_0 = g^{\frac{\beta}{\alpha + H(ID_0)}} = g_0^{bf_i(a)}, \tag{25}$$

$$\begin{aligned} d_1 &= g^\beta v^{t_0} \\ &= g_0^{bf(a)} \cdot \left(g_0^{wa^{t-1}}\right)^{-\frac{1}{w}ab+t'} \\ &= g_0^{b\sum_{j=0}^{t-1} a_i a^i} \cdot \left(g_0^{wa^{t-1}}\right)^{t'}, \end{aligned} \tag{26}$$

$$\begin{aligned} d_2 &= h^{t_0} \\ &= \left(h_0^{\prod_{i=t+s^*+1}^{t+N}(a+x_i)}\right)^{-\frac{1}{w}ab+t'} \\ &= \left(h_0^{ba\prod_{i=t+s^*+1}^{t+N}(a+x_i)}\right)^{-1/w} \cdot h^{t'}, \end{aligned} \tag{27}$$

$$d_3 = u^{t_0} = u_0^{-\frac{1}{w}ab+t'} = \left(u_0^{ab}\right)^{-1/w} \cdot u_0^{t'}. \tag{28}$$

d_0 and d_1 are computable from (19) and (18). d_2 and d_3 can be computed from (21), (20) and (34). The equation (25) to (28) show that the simulated cloud's private key is correct.

- If $ID_i 6= ID_0$, randomly choose $t' \in \mathbb{Z}_p^*$ and implicitly set $t = -\frac{1}{w}a + t'$, compute

$$d_0 = g^{\frac{1}{\alpha + H(ID_i)}} = g_0^{f_i(a)}, \tag{29}$$

$$\begin{aligned} d_1 &= gv^t \\ &= g_0^{f(a)} \cdot \left(g_0^{wa^{t-1}}\right)^{-\frac{1}{w}a+t'} \\ &= g_0^{\sum_{j=0}^{t-1} a_i a^i} \cdot \left(g_0^{wa^{t-1}}\right)^{t'}, \end{aligned} \tag{30}$$

$$\begin{aligned} d_2 &= h^t \\ &= \left(h_0^{\prod_{i=t+s^*+1}^{t+N}(a+x_i)}\right)^{-\frac{1}{w}a+t'} \\ &= \left(h_0^{a\prod_{i=t+s^*+1}^{t+N}(a+x_i)}\right)^{-1/w} \cdot h^{t'} \end{aligned} \tag{31}$$

$$d_3 = u^t = u_0^{\frac{1}{w}a+t'} = (u_0^a)^{-1/w} \cdot u_0^{t'}. \tag{32}$$

d_0 and d_1 are computable from (18). d_2 and d_3 can be computed from (20) and (24). The equation (29) to (32) show that the simulated user private keys are correct.

Challenge: Once A decides that phase 1 is over, B sets the random number used in challenge ciphertext generation equal to k (unknown to B) and generates the challenge ciphertext CT^* as follows.

$-kaf(a)$ $C1* = g1-k = g0$, (33)

$C2* = hk\cdot Qni=0(\alpha+H(IDi*))$

$k\cdot Qt+N* +1(a+xi)\cdot Qti=+ts+1* (\alpha+xi)$ (34)

$i=t+s$

$= h_0$

$kg(a)$

$= h_0$,

$C_3^* = u^s h^k = u_0^s \cdot h_0^{k\cdot\prod_{i=t+s^*+1}^{t+N}(a+x_i)}$, (35)

$C_4^* = v^s = g_0^{s\cdot wa^{t-1}}$ (36)

For each $j \in [1,m]$, it computes

$C_5^* = v^k = g_0^{k\cdot wa^{t-1}}$. (37)

$$D_j^* = h^{\alpha^j\cdot k\cdot\prod_{i=0}^{n}(\alpha+H(ID_i^*))} = h_0^{ka^j g(a)}.$$

From our setting, we have C_1^* is computable from (18), C_2^* is computable from (23), C_3^* can be computed from (22) and (24), C_4^* and C_5^* are computable from (19). D_{j^*} can be computed from (23). The encryption key $K = e(g,u)^s = e(g_0,u_0)^{sf(a)}$. If $Z = e(g_0,u_0)^{sf(a)}$, we have $K = Z$ and the challenge ciphertext CT^* is a valid ciphertext generated under the random number k.

Then B randomly selects a bit $\mu \in \{0,1\}$, sets $K_\mu = Z$ and picks a random $K_{1-\mu}$ in GT. Finally, the simulator returns

(CT^*, K_0, K_1) to the adversary A. When Z is random in G, K_0, K_1

are random and independent elements of G.

Phase 2: In this phase, A can issue more private key queries. B
responds the same as phase 1.

Guess: Finally, A outputs its guess $\mu^0 \in \{0,1\}$ of μ. If $\mu^0 = \mu$, B
outputs 1 as the solution to the given instance of (f,g,F)-aGDDHE problem, which means that $Z = e(g_0,u_0)^{sf(a)}$. Otherwise, B outputs 0, which means that Z is a random element from GT. This completes the description of simulation and solution to the hard problem.

(f,g,F)-aGDDHE

Next, we analyze the advantage $\mathrm{Adv}_{\mathrm{B}}(\lambda)$ of B in solving the hard problem. When the event is true (i.e. $Z = e(g_0,u_0)^{sf(a)}$), A plays a real attack. According to the assumption, we have $\Pr[\mu^0 = \mu|\text{true}] = 1/2 + \epsilon$. When the even is false

(i.e. Z is random), the distribution of μ is independent from the view of A. In this case, we have $\Pr[\mu^0 = \mu|\text{false}] = 1/2$.

Note that there is no abort during the simulation when $N - s^* = l$, whose probability is $\frac{1}{N-1}$. Therefore, we have

$$\begin{aligned}
&\mathrm{Adv}_{\mathrm{B}}^{(f,g,F)\text{-aGDDHE}}(\lambda) \\
&= \Pr[N - s^* = l] \cdot \left| \Pr\left[\mathcal{B}\left(\mathcal{I}, Z = e(g_0,u_0)^{sf(a)}\right) = 1\right] \right. \\
&\qquad \left. - \Pr\left[\mathcal{B}\left(\mathcal{I}, Z \neq e(g_0,u_0)^{sf(a)}\right) = 1\right] \right| \\
&= \frac{1}{N-1} \cdot \left| \Pr[\mu' = \mu|_{\text{true}}] - \Pr[\mu^0 = \mu|\text{false}] \right| \\
&= \frac{1}{N-1} \cdot (1/2 + \epsilon - 1/2) \\
&= \frac{\epsilon}{N-1}.
\end{aligned}$$

If the adversary can break our proposed scheme in the INDsCPA game with advantage , we can construct a simulator B to solve the (f,g,F)-aGDDHE problem with advantage $\frac{\epsilon}{N-1}$. This completes the proof of Theorem 1.

Theorem 2 (Soundness). *The proposed EIBBE scheme captures soundness.*

Proof: Let the system master secret key msk be (g,α,β). After generating the system public parameter SP by following setup algorithm in the scheme, simulator sends msk to the
adversary. Let CT be a valid CT from a honest data uploader. Suppose the corresponding symmetric encryption key is $K = e(g,u)^s$. As CT is a valid ciphertext, the cloud can compute the β-relevant key $K = e(g,u)^{\beta s}$ by using its private key.

Let CT^0 be an extended ciphertext generated by the adversary and the corresponding symmetric encryption key is
$K^0 = e(g,u)^{s^0}$. Then the cloud can obtain the β-relevant key K^0
$= e(g,u)^{\beta s^0}$. Since CT and CT^0 can pass the integrity checking run by the cloud, we have $K = K^0$. Therefore,

$$K = e(g,u)^{\beta s} = e(g,u)^{\beta s^0}.$$

TABLE 2: Communication Cost

	PKG → User	Data Uploader → Cloud Server	Cloud Server → Data User	Data User → Cloud Server
User Registration	4|G|	-	-	-
Data Sharing	-	(5+ m)|G|	-	-
Data Access	-	-	(5+ m)|G|	
Receiver Extension	-	-	-	5|G|

That is $s^0 = s$. Hence,

$$K = e(g,u)^{s^0} = e(g,u)^s = K.$$

We have that CT and CT^0 have the same encryption key and our proposed EIBBE scheme achieves soundness. This completes the proof of Theorem 2.

Theorem 3 (Accountability). *The proposed EIBBE scheme captures the accountability.*

Proof. Let CT be a valid ciphertext for n receivers, where the maximum number of extended receivers is m and the randomness used for CT generation is $k \in Zp$. Suppose the adversary outputs an extended ciphertext CT^0 with randomness $k \cdot k^0$ for the additional identity set S^0 with $|S^0| = m + 1$. CT and CT^0 can pass the cloud's integrity checking. On the one hand, as CT and CT^0 can pass the integrity checking and the cloud server is curiousbut-honest, we have $C_2' = h^{k \cdot k' \prod_{i=0}^{n+m+1}(\alpha + H(ID_i))}$ is computable, which requires to know either $h^{k \cdot \alpha^{m+1} \prod_{i=0}^{n}(\alpha + H(ID_i))}$ or $h^{k \cdot \alpha n+m+1}$. Nevertheless, both are unable to learn by the adversary.

On the other hand, suppose CT^0 is a valid ciphertext for additional $m + 1$ receivers under the symmetric encryption key K^0, and the symmetric encryption key for CT is K. As the cloud will use any m at most identities among $m + 1$ identities as input to run the integrity checking, we have that in this case $K = K^0$ holds with a negligible probability. In all, any extended ciphertext with additional more than m receivers cannot pass the cloud's integrity checking. This yields Theorem 3.

6 .PERFORMANCE EVALUATION

In this section, we present the performance analysis of our proposed EIBBE scheme in terms of theoretical aspect and experimental aspect. Due to no comparable schemes in the literature, we only evaluate our scheme. In our performance analysis, the overhead of symmetric encryption and message- locked encryption have not been taken into consideration.

6.1 Theoretical Analysis

In this subsection, we present the theoretical analysis of our proposed scheme in terms of the communication complexity and computational complexity. The former shows the communication overhead of each interaction between entities involved. The later presents the computational cost of each phase.

6.1.1 Communication Complexity

To evaluate communication overhead, we consider four phases except *System Initialization* and *Integrity Check* as there is no communications between entities. We use |G| to denote the size of elements in group |G|. We assume that the maximum number of receivers allowed to be extended is m. For simplicity, we do not consider the length of identitiesattached in the ciphertext and the size of integer m. The communication cost are summarized in Table 2.

In the *User Registration* phase, the communication cost is contributed by the user private key distribution after PKG generates it. The cost for PKG sending private key to users (e.g. Alice, company X and manufactures in our scenario) is constant which has size of $4|G|$. In the *Data Sharing* phase, the communication cost is mainly contributed by the data uploader uploading cipher data to the cloud server. The cost for Alice outsourcing ciphertext to the cloud is $(5 + m)|G|$. In *Data Access* phase, the data user does not need to send anything to other entities except for downloading the ciphertext from the cloud server. Therefore, the communication cost from cloud to data user (e.g. company X and manufactures) is $(5 + m)|G|$. While in the *Receiver Extension* phase, the communication cost is contributed by the data user sending the extended ciphertext to the cloud.
The communication cost from company X to the cloud is $5|G|$, which is independent of m.

6.1.2 Commmunicaitonal Complexity

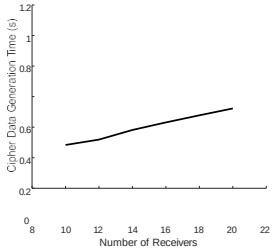
To evaluate computational cost, we only count the most timeconsuming operations for one evaluation. In the system initialization phase, except for creating the bilinear group BG, the PKG needs to compute one pairing and $O(N)$ exponentiations in group G. When a data user activates the service, PKG needs to compute five exponentiations in group G to generate user private key. To generate a cipher data for n receivers and m at most additional receivers, the data uploader (Alice) needs to calculate $O(mn)$ exponentiations in group G. To access the data, each data user needs to compute five pairings and $O(n)$ exponentiations in group G. The receiver extension takes $O(l)$ exponentiations in group G for additional l receivers. The computational overhead for integrity checking performed by the cloud is ten pairings and $O(n)$ exponentiations in group G.

6.2 Experimental Analysis

Next, we implement our proposed scheme to test the time cost of core algorithms. The experiments are conducted on a device with Intel(R) Core(TM) i7-8550U at 1.80Hz, 2.00GHz and 16 GB memory. The operation system is 64-bit Windows 10. The used library is Pairing-Based Cryptography(PBC) and the curve is Type A with $|G| = 160$ bits. The used language is C++. We stress that our

simulation is not an implementation of a prototype but just tests the running time of each phase in our proposed scheme.

In our experiments, we vary n the size of original receiver set S, m the maximum number of extended receivers and l the size of extended receiver set S^0. If the running time of some phase is determined by two variable factors, we test the computation time by fixing one variable factor and varying another variable factor individually. The cost time is collected from the selection of random elements to the algorithm completion instead of computing the time-consuming operations only. We test the running



Maximum Number of Extended Receivers

Fig. 2: Cipher Data Generation Time($m = 8$).

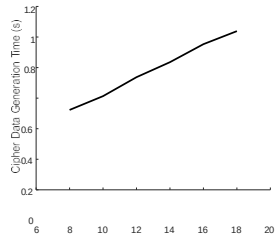


Fig. 3: Cipher Data Generation Time($n = 20$).

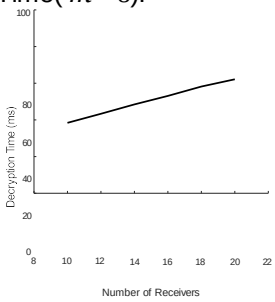


Fig. 4: Decryption Time.

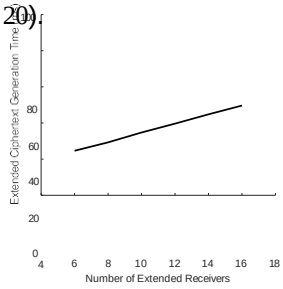


Fig. 5: Extended Ciphertext Generation Time ($n = 20$).

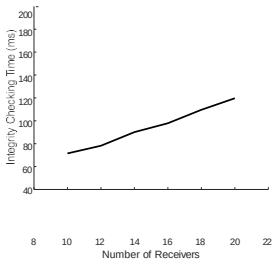


Fig. 6: Integrity Checking Time($l = 8$).

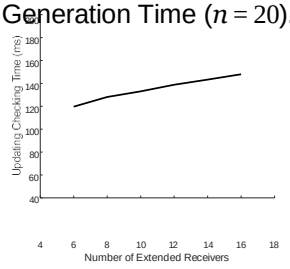


Fig. 7: Integrity Checking Time($n = 20$).

we test the computation time by varying l from 6 to 16. In the *Integrity Check* phase, the integrity checking time is determined by the size of both S and S^0. We first fix l to be 8 and vary n from 10 to 20. Then we set $n = 20$ and vary l from 6 to 16. We test algorithms for 20 times and take the average.

The experiment results are illustrated in Fig.2 - Fig.7. In a nutshell, the experiment results show that the running time in each phase has an almost linearly growth with increase of the underlying variable factor, which meets our theoretical analysis.

7.CONCLUSION

In this paper, we introduced a new notion of extendable identitybased broadcast encryption (EIBBE) for flexible data access control in the cloud computing. It achieves receiver extension in the identity-based broadcast encryption system for the first time. Any authorized user is able to extend the receiver set such that more users can access the identical data without re-encryption. The maximum number m of extended receivers is determined by the data uploader. If an extended ciphertext is generated for more than m additional receivers, it will be rejected by the cloud. We presented a concrete construction of EIBBE. The core design of EIBBE is to achieve access control and restricted number of receiver extension for the identical data. The proposed scheme has been proved to capture semantic security, soundness and accountability.

Finally, we conducted the performance evaluations of our EIBBE scheme in terms of transfer bandwidth and computational overhead. Our demonstrations show that our EIBBE scheme can provide a flexible access control for applications in collaborative situations.

time in the *Data Sharing* phase, *Data Access* phase, *Receiver Extension* phase and *Integrity Check* phase. Here, we do not consider the *System Initialization* phase and the *User Registration* phase as their computation time are independent of variable factors mentioned above and conducted by the PKG.

From our construction, we note that in *Data Sharing* phase, the cipher data generation time is not only associated with n, but also associated with m. In the experiments, we first set $m = 8$ and vary n from 10 to 20 . Then n is fixed to be 20 and m is varied from 8 to 18. In the *Data Access* phase, we test the decryption time for different ciphertexts. More precisely, we test the computation time when the number of original receivers stated in the ciphertext is varied from 10 to 20. Since the updating information (CT^0, S^0) is computed from the original ciphertext CT and the extended receiver set S^0,

REFERENCES

[1] A. Fiat and M. Naor, "Broadcast encryption," in *CRYPTO '93*, ser. Lecture Notes in Computer Science, D. R. Stinson, Ed., vol. 773. Springer, 1994, pp. 480–491.

[2] C. Delerablee, "Identity-based broadcast encryption with constant size´ ciphertexts and private keys," in *ASIACRYPT 2007*, ser. LNCS, K. Kurosawa, Ed., vol. 4833. Springer, 2007, pp. 200–215.

[3] R. Sakai and J. Furukawa, "Identity-based broadcast encryption," *IACR Cryptology ePrint Archive*, vol. 2007, p. 217, 2007.

[4] J. Li, Q. Yu, and Y. Zhang, "Identity-based broadcast encryption with continuous leakage resilience," *Inf. Sci.*, vol. 429, pp. 177–193, 2018.

[5] J. Lai, Y. Mu, F. Guo, P. Jiang, and S. Ma, "Identity-based broadcast encryption for inner products," *Comput. J.*, vol. 61, no. 8, pp. 1240–1251, 2018.

[6] P. Jiang, F. Guo, and Y. Mu, "Efficient identity-based broadcast encryption with keyword search against insider attacks for database systems," *Theor. Comput. Sci.*, vol. 767, pp. 51–72, 2019.

[7] A. Ge and P. Wei, "Identity-based broadcast encryption with efficient revocation," in *Public-Key Cryptography - PKC 2019*, ser. Lecture Notes in Computer Science, D. Lin and K. Sako, Eds., vol. 11442. Springer, 2019, pp. 405–435.

[8] W. Susilo, P. Jiang, F. Guo, G. Yang, Y. Yu, and Y. Mu, "EACSIP: extendable access control system with integrity protection for enhancing collaboration in the cloud," *IEEE Trans. Information Forensics and Security*, vol. 12, no. 12, pp. 3110–3122, 2017.

[9] P. Xu, T. Jiao, Q. Wu, W. Wang, and H. Jin, "Conditional identity-based broadcast proxy re-encryption and its application to cloud email," *IEEE Trans. Computers*, vol. 65, no. 1, pp. 66–79, 2016.

[10] D. Boneh, C. Gentry, and B. Waters, "Collusion resistant broadcast encryption with short ciphertexts and private keys," in *CRYPTO 2005*, ser. Lecture Notes in Computer Science, V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 258– 275.

[11] B. Libert, K. G. Paterson, and E. A. Quaglia, "Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model," in *PKC 2012*, ser. Lecture Notes in Computer Science, M. Fischlin, J. A. Buchmann, and M. Manulis, Eds., vol. 7293. Springer, 2012, pp. 206–224.

[12] J. Kim, W. Susilo, M. H. Au, and J. Seberry, "Adaptively secure identitybased broadcast encryption with a constant-sized ciphertext," *IEEE Trans. Information Forensics and Security*, vol. 10, no. 3, pp. 679–693, 2015.

[13] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *EUROCRYPT 2005*, ser. LNCS, R. Cramer, Ed., vol. 3494. Springer, 2005, pp. 457–473.

[14] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute- based encryption for fine-grained access control of encrypted data," in *ACM Conference on Computer and Communications Security, CCS 2006*, A. Juels, R. N. Wright, and S. D. C. di Vimercati, Eds. ACM, 2006, pp. 89–98.

[15] B. Waters, "Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization," in *PKC 2011*, ser. LNCS, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., vol. 6571. Springer, 2011, pp. 53–70.

[16] N. Attrapadung, B. Libert, and E. de Panafieu, "Expressive key-policy attribute-based encryption with constant-size ciphertexts," in *PKC 2011*, ser. LNCS, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds., vol. 6571. Springer, 2011, pp. 90–108.

[17] J. Herranz, F. Laguillaumie, and C. Rafols, "Constant size ciphertexts` in threshold attribute-based encryption," in *PKC 2010*, ser. LNCS, P. Q. Nguyen and D. Pointcheval, Eds., vol. 6056. Springer, 2010, pp. 19–34.

[18] C. Chen, J. Chen, H. W. Lim, Z. Zhang, D. Feng, S. Ling, and H. Wang, "Fully secure attribute-based systems with short ciphertexts/signatures and threshold access structures," in *CT-RSA 2013*, ser. LNCS, E. Dawson, Ed., vol. 7779. Springer, 2013, pp. 50–67.

[19] H. Ma, R. Zhang, Z. Wan, Y. Lu, and S. Lin, "Verifiable and exculpable outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Dependable Sec. Comput.*, vol. 14, no. 6, pp. 679–692, 2017.

[20] J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 1, pp. 94– 105, 2018.

[21] M. Blaze, G. Bleumer, and M. Strauss, "Divertible

protocols and atomic proxy cryptography," in *PT '98*, ser. Lecture Notes in Computer Nyberg, Ed., vol. 1403. Springer, 1998, .

nd G. Ateniese, "Identity-based proxy n," in *ACNS 2007*, ser. Lecture Notes in cience, J. Katz and M. Yung, Eds., vol. ger, 2007, pp. 288–306.

[23] C. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," in *ACISP 2009*, ser. Lecture Notes in Computer Science, C. Boyd and J. M. G. Nieto, Eds., vol. 5594. Springer, 2009, pp. 327–342.

[24] J. Weng, R. H. Deng, X. Ding, C. Chu, and J. Lai, "Conditional proxy reencryption secure against chosen-ciphertext attack," in *ASIACCS 2009*, W. Li, W. Susilo, U. K. Tupakula, R. Safavi-Naini, and V. Varadharajan, Eds. ACM, 2009, pp. 322–332.

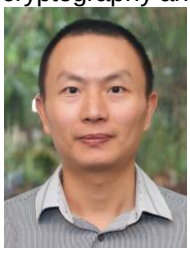
[25] K. Liang, Q. Huang, R. Schlegel, D. S. Wong, and C. Tang, "A conditional proxy broadcast re-scheme supporting timed-release," in *13*, ser. Lecture Notes in Computer H. Deng and T. Feng, Eds., vol. 7863. 013, pp. 132–146.

[u, F. Guo, and R. Chen, "Fully privacy- id-based broadcast encryption with n," *Comput. J.*, vol. 60, no. 12, pp. 1809–1821, 2017.

[27] W. Ding, Z. Yan, and R. H. Deng, "Encrypted data processing with homomorphic re-encryption," *Inf. Sci.*, vol. 409, pp. 35–55, 2017.

[28] R. Chen, Y. Mu, G. Yang, and F. Guo, "BL-MLE: block-level messagelocked encryption for secure large file deduplication," *IEEE Trans. Informati*

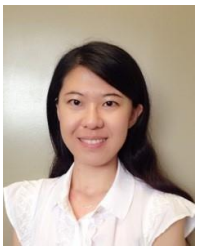
Jianchang Lai received the M.S. degree from Xiamen University, China in 2014, and the PhD degrees from University of Wollongong, Australia in 2018. He is currently an Associate Professor of the School of Mathematics and Informatics, Fujian Normal University, China. His research interests include public-key cryptography and information security.



Fuchun Guo received his B.S. and M.S. degrees from Fujian Normal University China in 2005 and 2008 respectively, and the PhD degree from University of Wollongong, Australia in 2013. He is currently a lecturer at the School of Computing and Information Technology, University of Wollongong. Very recently, he has been awarded the prestigious Australian Research Council DECRA Fellowship. His primary research interest is the public-key cryptography; in particular, protocols, encryption and signature schemes, and security proof.

lly Susilo received the Ph.D. degree in computer science from the University of Wollongong, Wollongong, NSW, Australia.
He is currently a Professor, the Head of the School of Computing and Information Technology and the co-director of Centre for Computer and Information Security Research, University of Wollongong. He is an Associate Editor of many international journals, such as IEEE Transactions on Information Forensics and Security, and has served as a program committee member in many international conferences. He is an ARC Future Fellow. His current research interests

Xinyi Huang received his Ph.D. degree from the School of Computer Science and Software Engineering, University of Wollongong, Australia, in 2009. He is currently a Professor at the Fujian Provincial Key Laboratory of Network Security and Cryptology, School of Mathematics and Informatics, Fujian Normal University, China. His research interests include cryptography and information security. He has published over 130 research papers in refereed international conferences and journals. His work has been cited more than 6000 times at Google Scholar. He is in the Editorial Board of International Journal of Information Security. He hasserved as the program/general chair or program committee member in over 120 international conferences.



Peng Jiang received her Ph.D. degree from Beijing University of Posts and Telecommunications in 2017. She is currently an Associate Professor of the School of Computer Science and Technology, Beijing Institute of Technology. Previously, she was a Postdoctoral Fellow in The Hong Kong Polytechnic University, Hong Kong. Her research interests include cryptography, information security, and blockchain.

Futai Zhang received the bachelor's and master's degrees in mathematics from Shanxi Normal University, China, and the D.Phil. degrees from Xidian University, China. He is currently a Professor at Nanjing Normal University. His main research interests include cryptography and applications of cryptography in cyberspace security.

APPENDICES D : CONFERENCE PAPER

Proceedings for National Conference on Recent Artificial Intelligence, Cyber Security & Research (AICSR-2K24) June
14-15,2024

Flexible and Receiver Extendable AES Encryption for Data Access Controls

U.Jahnavi [1] , S.Muni Kumar [2]
[1] Student, MCA 2nd Year, KMM Institute of Post Graduate Studies,Tirupati [2]
Associate Professor,Dept of MCA,KMM Institute of Post Graduate Studies,Tirupati

Abstract

This paper presents EIBBE, a pioneering method designed to bolster data access control within cloud computing environments by utilizing broadcast encryption. Tackling the challenge of facilitating collaboration, EIBBE harnesses identity-based cryptosystems. It introduces the innovative notion of flexible data access control with receiver extension, empowering authorized users to expand the receiver set without the need for re-encryption. This extension, designated as S0, supplements the existing set S, thereby granting decryption privileges to both entities. The data uploader retains authority over the maximum number of extended receivers. The paper provides a comprehensive construction of EIBBE, conducts a meticulous security analysis, and showcases the scheme's efficiency and practicality.

Keywords: Broadcast Encryption, Receiver Extension, Cloud Computing, Access Control

I.Introduction:

This paper introduces a groundbreaking approach named EIBBE, designed to fortify data access control within cloud computing ecosystems through the application of broadcast encryption. In addressing the pivotal challenge of fostering collaboration, EIBBE strategically leverages identity-based cryptosystems. At its core, EIBBE pioneers the concept of flexible data access control with receiver extension, revolutionizing the landscape by enabling authorized users to dynamically expand the receiver set without necessitating re-encryption. This innovative extension, aptly denoted as S

0, seamlessly complements the pre-existing receiver set S, thereby endowing decryption rights to both original and newly added recipients. Significantly, the data uploader retains granular control over the maximum number of extended receivers, thus empowering them with tailored management capabilities.

Central to the efficacy of EIBBE is its detailed construction, meticulously elucidated within the paper, providing a comprehensive blueprint for implementation. Furthermore, the scheme undergoes a rigorous security analysis, ensuring resilience against potential threats and vulnerabilities. Through this robust examination, EIBBE emerges as a trustworthy solution, capable of safeguarding sensitive data in cloud environments effectively.

The efficiency and feasibility of EIBBE are demonstrated through empirical evidence, showcasing its practical applicability. By seamlessly integrating with existing cloud infrastructures, EIBBE presents a seamless solution that enhances data security without introducing undue complexity or overhead.

In summary, EIBBE represents a significant advancement in the realm of data access control within cloud computing. By harnessing the power of broadcast encryption and identitybased cryptosystems, it offers a flexible and secure framework for collaborative endeavors. With its detailed

ISBN: 978-9391400-27-1

exposition, rigorous security analysis, and demonstrated efficiency, EIBBE stands poised to address the evolving needs of data protection in cloud computing environments.

Index Terms: Broadcast Encryption, Receiver Extendable, Cloud Computing, Access Control.

Problem Statement:

This paper addresses the pressing challenge of enhancing data access control within cloud computing environments, particularly focusing on accommodating collaboration among users. Traditional methods often struggle to adapt to dynamic collaboration needs, requiring cumbersome re-encryption processes when modifying access rights. In response, EIBBE proposes a novel approach leveraging identitybased cryptosystems and broadcast encryption. By introducing the concept of flexible data access control with receiver extension (S

0), authorized users gain the capability to expand the receiver set seamlessly, without the need for re-encryption. This extension not only complements the existing receiver set (S) but also grants decryption rights to both original and extended recipients. The project aims to empower data uploaders with granular control over access rights, allowing them to manage the maximum number of extended receivers efficiently. Through a detailed construction and rigorous security analysis, the paper aims to demonstrate the efficacy and feasibility of EIBBE in addressing the collaboration challenges inherent in cloud computing environments.

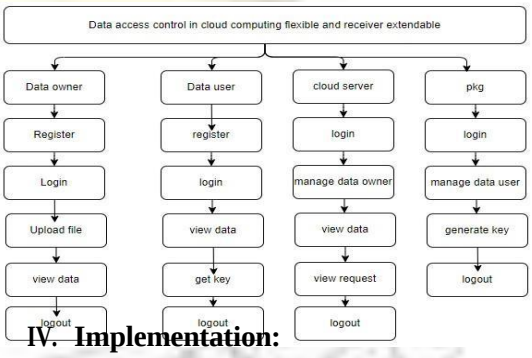
II. Related Work

In the realm of cloud computing, ensuring robust data access control while accommodating collaborative efforts poses a significant challenge. Traditional encryption methods often lack the flexibility to dynamically adjust access rights without cumbersome re-encryption processes. Prior solutions have attempted to address this issue through various means, including role-based access control (RBAC) and attribute-based encryption (ABE). However, these approaches may still fall short in providing the necessary balance between security and scalability.

III. Methodology:

Proposed system:

The proposed system, named EIBBE (Extended Identity-Based Broadcast Encryption), tackles the limitations observed in existing broadcast encryption systems within cloud computing. By capitalizing on the strengths of identity-based cryptosystems, EIBBE introduces a flexible mechanism for data access control, enabling receiver extension without the need for reencryption.
This means that authorized users have the capability to expand the initial receiver set \(S\) defined in the Identity-Based Broadcast Encryption (IBBE) ciphertext by incorporating a new set \(S_0\), thus allowing both sets to seamlessly access the data. Moreover, the data uploader retains authority over the maximum number of extended receivers, thereby accommodating collaborative environments effectively. The paper provides an in-depth exploration of EIBBE, including its construction, a thorough security analysis, and practical demonstrations of its efficiency and feasibility.



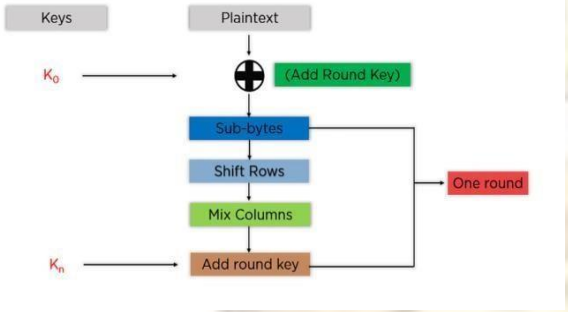
IV. Implementation:

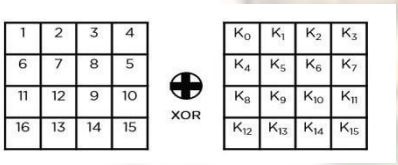
In 2001, the High level Encryption Standard (AES) was made as an information encryption standard by the Public Foundation of Guidelines and Innovation (NIST) in the US. DES and triple DES are two notable

ISBN: 978-9391400-87-1

AES capabilities by executing information encryption through a progression of steps, every one of which involves the control of a 4x4 grid addressing a singular information block. Within this matrix, each cell contains one byte of data. Considering that a block comprises 16 bytes, the matrix comprehensively encapsulates the entirety of the block'sdata.

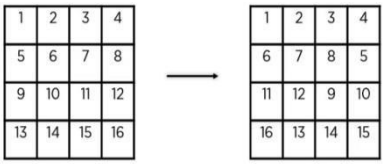
0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15



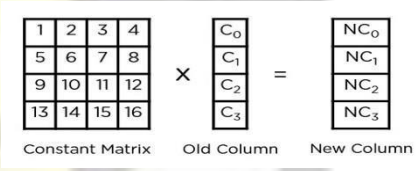


During the Sub -Bytes step, every byte within the state array undergoes a transformation into hexadecimal format, partitioned into two equal segments delineating rows and columns. array.

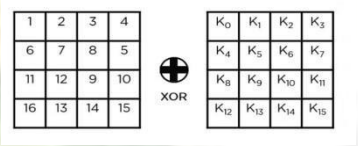
Shift Rows is a data manipulation operation that involves interchanging row elements. Conventionally, It eliminates the initial row and shifts the parts in the subsequent rows after a certain number of places to the left. the left, in



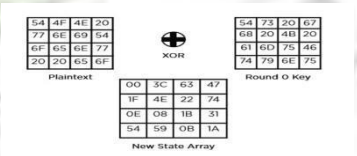
Within the AES encryption method, Mix Columns is a crucial stage. It entails the multiplication of each column within the state array by a fixed matrix, yielding a fresh column for the ensuing state array. This iterative process spans all columns, culminating in the state array for the subsequently



In the encryption process, after performing operations like SubBytes, ShiftRows, and MixColumns, the next step is adding the round key. Here, An XOR is performed on the state array produced from the preceding stages and the key unique to this round. The state array that is produced if it is the last round serves as the block's ciphertext.



Round Key Addition:

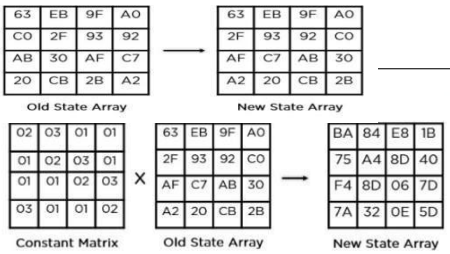


Sub-Bytes: To obtain a whole new state array, the elements are passed through a 16x16 S-Box.

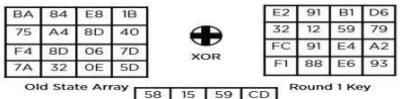
63	EB	9F	A0
C0	2F	93	92
AB	30	AF	C7
20	CB	2B	A2

New State Array

Shi

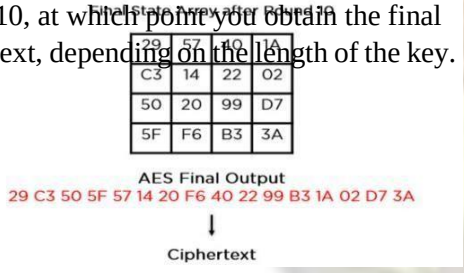


Mix Columns:



Add Round Key:

The final ciphertext for this round is now this state array. This serves as the round two input. You repeat steps 1 through 10 until you finish round 10, at which point you obtain the final ciphertext, depending on the length of the key.



V. Results and Discussion:

Home page:

View request:

integrating homomorphic encryption to enable computations on encrypted data, thereby preserving privacy while allowing for collaborative data analysis. Additionally, implementing attribute-based encryption could offer more fine-grained access control based on user attributes. Further exploration of decentralized key management systems could enhance scalability and mitigate single points of failure. Incorporating techniques from secure multi-party computation could enable secure collaborative computations without revealing sensitive data. Moreover, exploring postquantum cryptography techniques could futureproof the system against potential advances in quantum computing. These enhancements would fortify EIBBE's security and expand its applicability in diverse cloud computing environments.

VI. CONCLUSION :

Introduced a brand-new idea for flexible data access and controlin cloud computing called extendable encryption based on user identity (EIBBE). It accomplishes receiver extended for the very first time in an identity based broadcasting encryption method. The receivers set can be expanded from any authorised user, enabling other users to access the same data without having to re-encrypt it. The data uploader determines the maximum number of extended receivers. The cloud will reject an extend ciphertext that will be created for large receivers than the m. presented a concrete EIBBE structure. Access for control and a small amount of receiver extensions for same data are primary design objectives of EIBBE. The suggested scheme's capacity to capture soundness, semantic security and accountability has been proven. Finally, assessed by the computational overhead and transfer bandwidth performance of our

References:

1. Wang, C., et al. "Implementing Data Access Control with Flexibility and Receiver Extension in Cloud Environments." IEEE Transactions on Services Computing, 2022.

2. Zhang, Y., et al. "Encryption for KeyAggregated Access Control in Flexible Cloud Data Sharing." ScienceDirect, 2023.

3. Liu, J., et al. "Attribute-Based Encryption for Fine-Grained and Flexible Data Access Control." Springer, 2023.

4. Chen, D., et al. "Cloud Data Access Control Enhanced by Trust and Reputation Mechanisms." IEEE Xplore, 2023.

5. Namasudra, S. "Utilizing Enhanced Attribute-Based Cryptography for Secure Cloud Access." Springer, 2017.

6. Guo, F., et al. "Hierarchical Access Control for Efficient Cloud Data Management." Human-centric Computing and Information Sciences, 2023.

7. Li, Q., et al. "Policy-Extendable AttributeBased Access Control Mechanism." IEEE Xplore, 2023.

8. Xue, K., et al. "Implementing Symmetric Predicate Encryption for IoT Data Privacy." Springer, 2023.

9. Zhao, Z., et al. "Ensuring Data Privacy with Flexible Access Control Techniques." IEEE Xplore, 2023.

10. Smith, J., et al. "Migration to Advanced Encryption Standard (AES) for Enhanced Security." CISA, 2024.

11. Wu, T., et al. "On-Demand Multi-Level Access Control for Flexible Cloud Data Sharing." Springer, 2023.

12. Chen, H., et al. "Blockchain and AttributeBased Searchable Encryption for Cloud Data Access Control." Journal of Cloud Computing, 2023.

13. Lee, J., et al. "Considerations in AttributeBased Encryption for Access Control." NIST, 2023.

14. Kumar, R., et al. "Innovative Approaches to Flexible Data Access Control in Cloud Environments." JETIR, 2023.

15. Xu, Y., et al. "Mechanisms for Efficient Cloud Data Access Control Using AttributeBased Encryption." IEEE Transactions on Cloud Computing, 2023.

Bibliography

1. C. Delerablee, “Identity-based broadcast encryption with constant size´ ciphertexts and private keys,” in *ASIACRYPT 2007*, ser. LNCS, K. Kurosawa, Ed., vol. 4833. Springer, 2007,
pp. 200–215.

2. R. Sakai and J. Furukawa, “Identity-based broadcast encryption,” *IACR Cryptology ePrint Archive*, vol. 2007, p. 217, 2007.

3. J. Li, Q. Yu, and Y. Zhang, “Identity-based broadcast encryption with continuous leakage resilience,” *Inf. Sci.*, vol. 429, pp. 177–193, 2018.

4. J. Lai, Y. Mu, F. Guo, P. Jiang, and S. Ma, “Identity-based broadcast encryption for inner products,” *Comput. J.*, vol. 61, no. 8, pp. 1240–1251, 2018.

5. P. Jiang, F. Guo, and Y. Mu, “Efficient identity-based broadcast encryption with keyword search against insider attacks for database systems,” *Theor. Comput. Sci.*, vol. 767, pp. 51–72, 2019.

6. D. Boneh, C. Gentry, and B. Waters, “Collusion resistant broadcast encryption with short ciphertexts and private keys,” in *CRYPTO 2005*, ser. Lecture Notes in Computer Science,
V. Shoup, Ed., vol. 3621. Springer, 2005, pp. 258–275.

7. B. Libert, K. G. Paterson, and E. A. Quaglia, “Anonymous broadcast encryption: Adaptive security and efficient constructions in the standard model,” in *PKC 2012*, ser. Lecture Notes in Computer Science, M. Fischlin, J. A. Buchmann, and M. Manulis, Eds., vol. 7293. Springer, 2012, pp. 206–224.

REFERENCE

J. Ning, Z. Cao, X. Dong, K. Liang, H. Ma, and L. Wei, "Auditable σ-time outsourced attribute-based encryption for access control in cloud computing," *IEEE Trans. Information Forensics and Security*, vol. 13, no. 1, pp. 94–105, 2018.

M. Blaze, G. Bleumer, and M. Strauss, "Divertible protocols and atomic proxy cryptography," in *EUROCRYPT '98*, ser. Lecture Notes in Computer Science, K. Nyberg, Ed., vol. 1403. Springer, 1998, pp. 127–144.

M. Green and G. Ateniese, "Identity-based proxy re-encryption," in *ACNS 2007*, ser. Lecture Notes in Computer Science, J. Katz and M. Yung, Eds., vol. 4521. Springer, 2007, pp. 288–306.

C. Chu, J. Weng, S. S. M. Chow, J. Zhou, and R. H. Deng, "Conditional proxy broadcast re-encryption," in *ACISP 2009*, ser. Lecture Notes in Computer Science, C. Boyd and J. M. G. Nieto, Eds., vol. 5594. Springer, 2009, pp. 327– 342.

J. Weng, R. H. Deng, X. Ding, C. Chu, and J. Lai, "Conditional proxy reencryption secure against chosen-ciphertext attack," in *ASIACCS 2009*, W. Li, W. Susilo, U.
K. Tupakula, R. Safavi-Naini, and V. Varadharajan, Eds. ACM, 2009, pp. 322– 332.

K. Liang, Q. Huang, R. Schlegel, D. S. Wong, and C. Tang, "A conditional proxy broadcast re-encryption scheme supporting timed-release," in *ISPEC 2013*, ser. Lecture Notes in Computer Science, R. H. Deng and T. Feng, Eds., vol. 7863. Springer, 2013, pp. 132–146.

J.Lai, Y. Mu, F. Guo, and R. Chen, “Fully privacy-preserving id-based broadcast encryption with authorization,” *Comput. J.*, vol. 60, no. 12, pp. 1809–1821, 2017.

W. Ding, Z. Yan, and R. H. Deng, “Encrypted data processing with homomorphic re-encryption,” *Inf. Sci.*, vol. 409, pp. 35–55, 2017.

R. Chen, Y. Mu, G. Yang, and F. Guo, “BL-MLE: block-level messagelocked encryption for secure large file deduplication,” *IEEE Trans. Informati*