Effective Management with Flexibility to Access Authorized Data from Cloud based IoT

A Project Report Submitted to
Jawaharlal Nehru Technological University,Kakinada
in the partial fulfillment for the award of the Degree of

BACHELOR OF TECHNOLOGY
IN
COMPUTER SCIENCE AND ENGINEERING

Submitted by

EEMANI MAHESWARI	18491A0569
GUDDAPATHALA VINEELA	18491A0564
RAAVI MOUNIKA	18491A0586
ARLA MANOGNA	18491A0561

Under the Noble Guidance of

P. Adilakshmi, M. Tech.

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

QIS COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

(An ISO 9001-2008 Certified & NBA Accredited Institution)

(Affiliated to Jawaharlal Nehru Technological University, Kakinada)

VENGAMUKKAPALEM, ONGOLE -523272, A.P

2018-2022

QIS COLLEGE OF ENGINEERING AND TECHNOLOGY

(AUTONOMOUS)

(An ISO 9001-2008 Certified & NBA Accredited Institution)

(Affiliated to Jawaharlal Nehru Technological University, Kakinada)

VENGAMUKKAPALEM, ONGOLE -523272, A.P

DEPARTMENTOF COMPUTER SCIENCE & ENGINEERING

BONAFIDE CERTIFICATE

*This is to certify that the project entitled "*Effective Management with Flexibility to Access Authorized Data from Cloud based*" is a bonafide* work of

EEMANI MAHESWARI	18491A0569
GUDDAPATHALA VINEELA	18491A0564
RAAVI MOUNIKA	18491A0586
ARLA MANOGNA	18491A0561

in the partial fulfillment of the requirement for the award of the degree of Bachelor of Technology in **COMPUTER SCIENCE & ENGINEERING** *and for the academic year* ***2021-2022****. This work is done under my supervision and guidance.*

Signature of the Guide
P. Adilakshmi, M. Tech
Assistant Professor
Dept. of CSE.

Signature of the Head of the Department
Dr. Y. Narasimha Rao, M. Tech, PhD
Professor and HOD
Dept. of CSE.

Signature of the External Examiner

ACKNOWLEDGMENT

Task successful" makes everyone happy. But the happiness will be gold without glitter if we didn't state the persons who have supported us to make it a success.

We would like to place on record the deep sense of gratitude to the Honorable Secretary & Correspondent **Sri. N. SURYA KALYAN CHAKRAVARTHY GARU, QIS Group of Institutions, Ongole** for providing necessary facilities tocarry the project work.

We express our gratitude to the Honorable chairman **Sri. N. NAGESWARA RAO GARU, QIS Group of Institutions, Ongole** for his valuable suggestions and advices in the B.Tech. Course.

We express our gratitude to **Dr. Y. V. HANUMANTHA RAO,** Principal of **QIS Collegeof Engineering & Technology, Ongole** for his valuable suggestions andadvices in the B. Tech course.

We express our gratitude to the **Head of the Department** of **CSE**, **Dr. Y. NARASIMHA RAO GARU, M. Tech, Ph.D., QIS College of Engineering & Technology, Ongole** for his constant supervision, guidance and co-operation throughout the project.

We would like to express our thankfulness to our project guide **P. ADILAKSHMI, M. Tech, Assistant professor**, **QIS College of Engineering & Technology, Ongole** for her constant motivation and valuable help throughout the project work.

Finally, we would like to thank our Parents, Family and friends for their co- operation to complete this project.

TEAM MEMBERS

EEMANI MAHESWARI	18491A0569
GUDDAPATHALA VINEELA	18491A0564
RAAVI MOUNIKA	18491A0586
ARLA MANOGNA	18491A0561

DECLARATION

We hereby declare that the project work entitled "**Effective Management with Flexibility to Access Authorized Data from Cloud based IoT"** done under the guidance of **P. ADILAKSHMI, M. Tech** is being submitted to the "COMPUTER SCIENCE AND ENGINEERING", QIS College of Engineering & Technology, Ongole is of our own and has not been submitted to any other University or Educational institution for any degree.

TEAM MEMBERS

EEMANI MAHESWARI	18491A0569
GUDDAPATHALA VINEELA	18491A0564
RAAVI MOUNIKA	18491A0586
ARLA MANOGNA	18491A0561

ABSTRACT

Cloud-based Internet of Things (IoT) management services are a promising means of ingesting data from globally dispersed devices. In this setting, it is important to regulate access to data managed by potentially untrusted cloud servers. Attribute-based encryption (ABE) is a highly effective tool for access control. However, applying ABE to IoT environments shows limitationsin the following three aspects: First, the demands for storage resources increase in proportion to the complexity of the access control policies. Second, the computation cost of ABE is onerous for resource-limited devices. Lastly, ABE alone is intractable to prevent illegal key-sharing which leads to unauthorized access to data. In this paper, we propose an efficient and secure cloud based IoT data management scheme using ABE. First, we remove the storage-side dependency on the complexity of the access control policies. Second, a substantial part of computationally intensive operations is securely outsourced to the cloud servers. Lastly, unauthorized access to data via illegal key-sharing is strictly forbidden. Our security analysis and experimental results show the security and practicability of the proposed scheme.

TABLE OF CONTENT

INTRODUCTION

1. INTRODUCTION

1.1 Objectives:

As billions of devices connect to the world, Internet of Things (IoT) management services outsource the IoT data to cloud services such as Amazon AWS IoT [1] or Google Cloud IoT Core [2]. In the cloud-based IoT management systems, IoT devices typically belong to different trust domains with complex, asymmetric trust relationships. Thus, it is challenging to regulate access to the outsourced IoT data of a specific security domain. Attribute-based encryption (ABE) is a promising solution because it provides secure and fine-grained access control over encrypted data according to access policies associated with it [3]. Specifically, in ciphertext-policy ABE (CP-ABE), an encryptor can specify an access policy for the ciphertext with a set of descriptive attributes [4]. A decryptor can recover the plaintext if and only if the access privilege in his secret key satisfies the access policy embedded in the ciphertext. In order to adopt CP-ABE to the cloud-based IoT management in practice, there are several issues to resolve. First, the ciphertext size grows in linear to the number of attributes [5], [6], [7]. This can be highly problematic in IoT systems because IoT applications and services need a multitude of attributes [8]. Although some CP-ABE schemes which support constant-size ciphertexts exist [9], [10], [11], [12], this functionality alone cannot suffice to deploy CPABE to IoT systems.

Second, CP-ABE loads heavy computational costs on a decryptor (rather than an encryptor), which could be battery-powered mobile devices like laptops [13], [14], [15]. Recent studies on outsourced decryption of ciphertexts showed how to allow an untrusted cloud server to partially decrypt ciphertexts on behalf of users. However, the increasing volume of ciphertexts could not be solved. To cope with the aforementioned issues, one may consider merging the existing approaches such as outsourceable decryption [5] and constant-size cipher text [11] ABE schemes for each purpose. Although this may seem plausible, it cannot resolve the problems because of a key blinding technique in the outsourceable decryption algorithm [5]. Specifically, it enables a user to blind his secret key using a (secret) blinding factor, say z, in the sense that the cloud can perform partial decryption using the blinded key and then return the plaintext masked by z, Thus, the user simply unmasks it using z. Applying this technique to constant size cipher text [11], however, results in a plaintext masked not only by z but also other elements that are needed for final decryption but unknown to the user. Therefore, the user can by no means recover the plaintext correctly. Li et al. [16] proposed a method to simultaneously achieve short ciphertext and outsourceable decryption functionalities. However, their scheme significantly limits the access control capabilities because a user can decrypt a cipher text if and only if the attributes assigned to him is the same as the attributes in the access policy.

Third, secret keys could be easily abused by illegal key sharing among unauthorized users. In this case, authorized but malicious users may collude with unauthorized users to illegally share their secret keys. Then, the unauthorized users would freely access IoT data in the cloud. Previous studies tackled the key leakage problem by proposing traceable ABE. Unfortunately, most of the traceable ABE schemes focus only on detecting original key owners. That is, once secret keys are shared by dishonest users, the shared key holders still freely enjoy unauthorized access to encrypted data in the cloud. Furthermore, traceability alone is of no use to preventing key leakage due to several practical key recovery attacks such as side channel analysis1. Similarly, key revocation cannot be a possible solution for the shared (or leaked) key problem because the shared (or leaked) key holders are still able to use the key to access and recover data until it is revoked. Overall, designing a secure and efficient access control scheme that resolves the above problems is a pivotal issue in cloud-based IoT management systems. In this paper, we propose an efficient and secure cloud based IoT data management scheme using our novel CPABE construction.

The proposed scheme features efficient storage and bandwidth management, and capability of tracing traitors if they illegally share their secret keys. The cloud server can perform a significant part of the computational overhead related to decryption via a user-specific transformation key. This key is crucial in forbidding unauthorized access by a shared (or leaked) key holder because the proposed access mechanism of the IoT data works as follows: the cloud (1) authenticates the identity of the key holder, (2) partially decrypts the ciphertext using the key holder's transformation key, and (3) returns the partially decrypted result. Note that the transformation key is closely associated with the key holder, who can recover the plaintext from the partially decrypted cipher text if and only if he is the original owner of the attribute key. Therefore, the other users who illegally hold the shared (or leaked) keys are unable to obtain the plaintext.

In summary, the proposed scheme is resilient to the forensically intractable key abuse attacks. This property holds even if the attributes in the shared (or leaked) key satisfy the access policy attached to the encrypted IoT data. Thus, users holding a shared (or leaked) key cannot recover the IoT data. Correct decryption is available only for the original key owner. Lastly, this resiliency method is built on top of our CP-ABE scheme which supports constant-size ciphertext, outsourceable decryption, and key traceability.

1.2 Existing System:

To cope with the aforementioned issues, one may consider merging the existing approaches such as outsourceable decryption [5] and constant-size ciphertext [11] ABE schemes for each purpose. Although this may seem plausible, it cannot resolve the problems because of a key blinding technique in the outsourceable decryption algorithm [5]. Specifically, it enables a user to blind his secret key using a (secret) blinding factor, say z, in the sense that the cloud can perform partial decryption using the blinded key and then return the plaintext masked by z, Thus, the user simply unmasks it using z. Applying this technique to constantsize ciphertext [11], however, results in a plaintext masked not only by z but also other elements that are needed for final decryption but unknown to the user.

1.3 Drawbacks of Existing System

Therefore, The user can by no means recover the plaintext correctly. Li et al. [16] proposed a method to simultaneously achieve short ciphertext and outsourceable decryption functionalities. However, their scheme significantly limits the access control capabilities because a user can decrypt a ciphertext if and only if the attributes assigned to him is the same as the attributes in the access policy.

LIERATURE SURVEY

2. LITERATURE SURVEY

2.1 Toward Secure Data Computation and Outsource for Multi-User Cloud-Based IoT

Abstract: Cloud computing has promoted the success of Internet of Things (IoT) with offering abundant storage and computation resources where the data from IoT sensors can be remotely outsourced to the cloud servers, whereas storing, exchanging and processing data collected through IoT sensors via centralised or decentralised cloud servers make cloud-based IoT systems prone to internal or external attacks. To protect IoT data against potential malicious users and adversaries, some cryptographic schemes have been applied to ensure confidentiality and integrity of IoT data. It is however a challenging task to perform any arithmetical computations once data items are encrypted. Fully-homomorphic encryption which is based on lattices can, in principle, provide a solution, but it is unfortunately inefficient in computation and hence cannot be applied to IoT. Fully-homomorphic encryption is feasible when we allow an involvement of semi-trusted server. However, it is challenging to provide such a system in the situation of distributed environments for shared IoT data. We solve this problem and provide a fully-homomorphic encryption scheme for cloud-based IoT applications. We introduce a new method with the aid of semi-trusted server who can help in the computation of the homomorphic multiplications without gaining any useful information of the encrypted data.

2.2 Security and Privacy in IoT-Cloud-Based e-Health Systems - A Comprehensive Review

Abstract: When the Internet and other interconnected networks are used in a health system, it is referred to as "e-Health." In this paper, we examined research studies from 2017–2020 to explore the utilization of intelligent techniques in health and its evolution over time, particularly the integration of Internet of Things (IoT) devices and cloud computing. E-Health is defined as "the ability to seek, find, understand and appraise health information derived from electronic sources and acquired knowledge to properly solve or treat health problems. As a repository for health information as well as e-Health analysis, the Internet has the potential to protect consumers from harm and empower them to participate fully in informed health-related decision-making. Most importantly, high levels of e-Health integration mitigate the risk of encountering unreliable information on the Internet. Various research perspectives related to security and privacy within IoT-cloud-based e-Health systems are examined, with an emphasis on the opportunities, benefits and challenges of the implementation such systems. The combination of IoT-based e-Health systems integrated with intelligent systems such as cloud computing that provide smart objectives and applications is a promising future trend.

2.3 IoT Privacy and Security: Challenges and Solutions

Abstract: Privacy and security are among the significant challenges of the Internet of Things (IoT). Improper device updates, lack of efficient and robust security protocols, user unawareness, and famous active device monitoring are among the challenges that IoT is facing. In this work, we are exploring the background of IoT systems and security measures, and identifying (a) different security and privacy issues, (b) approaches used to secure the components of IoT-based environments and systems, (c) existing security solutions, and (d) the best privacy models necessary and suitable for different layers of IoT driven applications. In this work, we proposed a new IoT layered model: generic and stretched with the privacy and security components and layers identification. The proposed cloud/edge supported IoT system is implemented and evaluated. The lower layer represented by the IoT nodes generated from the Amazon Web Service (AWS) as Virtual Machines. The middle layer (edge) implemented as a Raspberry Pi 4 hardware kit with support of the Greengrass Edge Environment in AWS. We used the cloud-enabled IoT environment in AWS to implement the top layer (the cloud). The security protocols and critical management sessions were between each of these layers to ensure the privacy of the users' information. We implemented security certificates to allow data transfer between the layers of the proposed cloud/edge enabled IoT model. Not only is the proposed system model eliminating possible security vulnerabilities, but it also can be used along with the best security techniques to countermeasure the cybersecurity threats facing each one of the layers; cloud, edge, and IoT.

2.4 Enabling privacy and security in Cloud of Things: Architecture, applications, security & privacy challenges

Abstract: The Cloud of Things (IoT) that refers to the integration of the Cloud Computing (CC) and the Internet of Things (IoT), has dramatically changed the way treatments are done in the ubiquitous computing world. This integration has become imperative because the important amount of data generated by IoT devices needs the CC as a storage and processing infrastructure. Unfortunately, security issues in CoT remain more critical since users and IoT devices continue to share computing as well as networking resources remotely. Moreover, preserving data privacy in such an environment is also a critical concern. Therefore, the CoT is continuously growing up security and privacy issues. This paper focused on security and privacy considerations by analyzing some potential challenges and risks that need to be resolved. To achieve that, the CoT architecture and existing applications have been investigated. Furthermore, a number of security as well as privacy concerns and issues as well as open challenges, are discussed in this work.

2.5 SYSTEM REQUIREMENTS:

HARDWARE REQUIREMENTS:

• Processor	: I3.
• Hard Disk	: 40 GB.
• Ram	: 2 GB.

SOFTWARE REQUIREMENTS:

• Operating system	: Windows.
• Coding Language	: JAVA/J2EE
• Data Base	: MYSQL
• IDE	: Netbeans8.1

2.6 Technology Description:

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

- Simple
- Architecture neutral
- Object oriented
- Portable
- Distributed
- High performance
- Interpreted
- Multithreaded
- Robust
- Dynamic
- Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java byte codes* —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer. Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

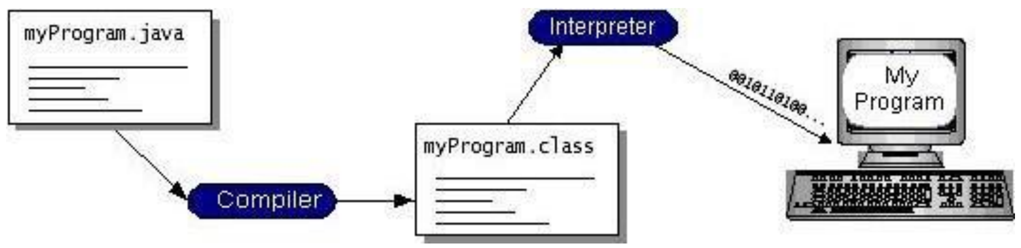
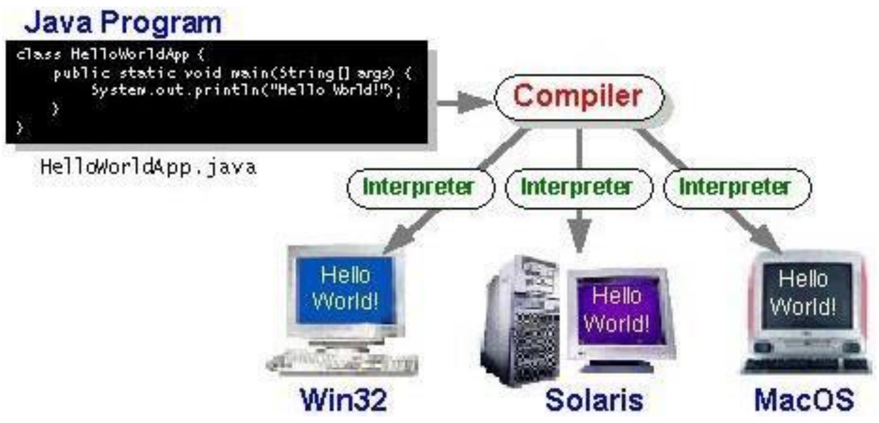


Fig:1 The process of Compilation and Execution

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



```
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

Fig:2 Java Compiler

The Java Platform

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

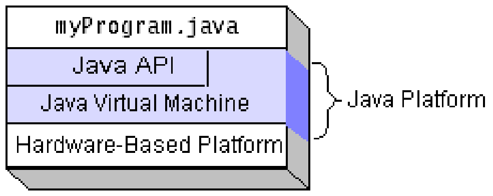
The Java platform has two components:

- The *Java Virtual Machine* (Java VM)
- The *Java Application Programming Interface* (Java API)

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

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

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



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

What Can Java Technology Do?

The most common types of programs written in the Java programming language are *applets* and *applications*. If you've surfed the Web, you're probably already familiar with applets. An applet is a program that adheres to certain conventions that allow it to run within a Java-enabled browser.

However, the Java programming language is not just for writing cute, entertaining applets for the Web. The general-purpose, high-level Java programming language is also a powerful software platform. Using the generous API, you can write many types of programs.

An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*. A servlet can almost be thought of as an applet that runs on the server side. Java Servlets are a popular choice for building interactive web applications, replacing the use of CGI scripts. Servlets are similar to applets in that they are runtime extensions of applications. Instead of working in browsers, though, servlets run within Java Web servers, configuring or tailoring the server.

How does the API support all these kinds of programs? It does so with packages of software components that provides a wide range of functionality. Every full implementation of the Java platform gives you the following features:

- **The essentials**: Objects, strings, threads, numbers, input and output, data structures, system properties, date and time, and so on.

- **Applets**: The set of conventions used by applets.
- **Networking**: URLs, TCP (Transmission Control Protocol), UDP (User Data gram Protocol) sockets, and IP (Internet Protocol) addresses.
- **Internationalization**: Help for writing programs that can be localized for users worldwide. Programs can automatically adapt to specific locales and be displayed in the appropriate language.
- **Security**: Both low level and high level, including electronic signatures, public and private key management, access control, and certificates.
- **Software components**: Known as JavaBeans^{TM}, can plug into existing component architectures.
- **Object serialization**: Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC^{TM})**: Provides uniform access to a wide range of relational databases.

The Java platform also has APIs for 2D and 3D graphics, accessibility, servers, collaboration, telephony, speech, animation, and more. The following figure depicts what is included in the Java 2 SDK.

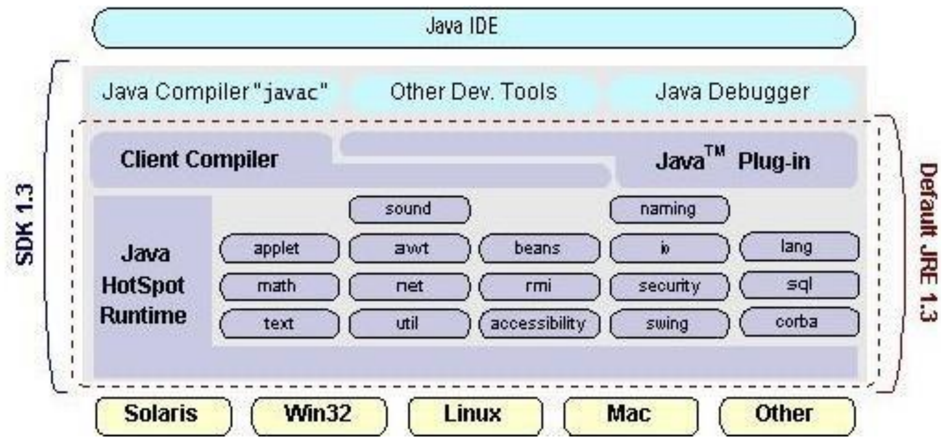
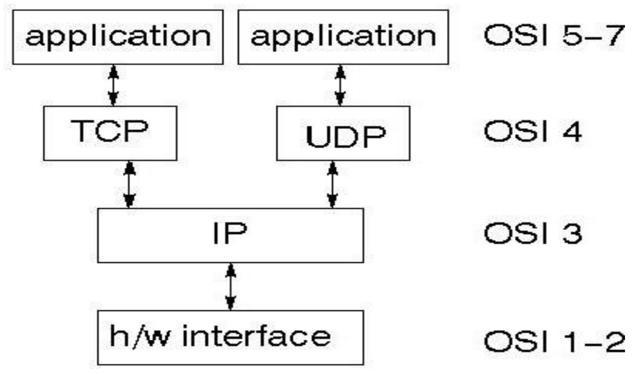


Fig:3 Java IDE

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one:



TCP is a connection-oriented protocol; UDP (User Datagram Protocol) is a connectionless protocol.

IP datagram's

The IP layer provides a connectionless and unreliable delivery system. It considers each datagram independently of the others. Any association between datagram must be supplied by the higher layers. The IP layer supplies a checksum that includes its own header. The header includes the source and destination addresses. The IP layer handles routing through an Internet. It is also responsible for breaking up large datagram into smaller ones for transmission and reassembling them at the other end.

UDP

UDP is also connectionless and unreliable. What it adds to IP is a checksum for the contents of the datagram and port numbers. These are used to give a client/server model - see later.

TCP

TCP supplies logic to give a reliable connection-oriented protocol above IP. It provides a virtual circuit that two processes can use to communicate.

Internet addresses

In order to use a service, you must be able to find it. The Internet uses an address scheme for machines so that they can be located. The address is a 32 bit integer which gives the IP address. This encodes a network ID and more addressing. The network ID falls into various classes according to the size of the network address.

Network address

Class A uses 8 bits for the network address with 24 bits left over for other addressing. Class B uses 16 bit network addressing. Class C uses 24 bit network addressing and class D uses all 32.

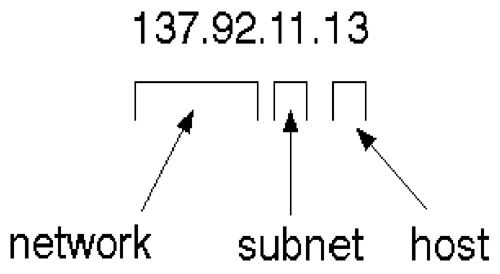
Subnet address

Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

Host address

8 bits are finally used for host addresses within our subnet. This places a limit of 256 machines that can be on the subnet.

Total address



The 32 bit address is usually written as 4 integers separated by dots.

Port addresses

A service exists on a host, and is identified by its port. This is a 16 bit number. To send a message to a server, you send it to the port for that service of the host that it is running on. This is not location transparency! Certain of these ports are "well known".

Sockets

A socket is a data structure maintained by the system to handle network connections. A socket is created using the call socket. It returns an integer that is like a file descriptor. In fact, under Windows, this handle can be used with Read File and Write File functions.

```
#include <sys/types.h>
#include <sys/socket.h>
int socket(int family, int type, int protocol);
```

Here "family" will be AF_INET for IP communications, protocol will be zero, and type will depend on whether TCP or UDP is used. Two processes wishing to communicate over a network create a socket each. These are similar to two ends of a pipe - but the actual pipe does not yet exist.

What is a Java Web Application?

A Java web application generates interactive web pages containing various types of markup language (HTML, XML, and so on) and dynamic content. It is typically comprised of web components such as JavaServer Pages (JSP), servlets and JavaBeans to modify and temporarily store data, interact with databases and web services, and render content in response to client requests.

Because many of the tasks involved in web application development can be repetitive or require a surplus of boilerplate code, web frameworks can be applied to alleviate the overhead associated with common activities. For example, many frameworks, such as JavaServer Faces, provide libraries for templating pages and session management, and often promote code reuse.

What is Java EE?

Java EE (Enterprise Edition) is a widely used platform containing a set of coordinated technologies that significantly reduce the cost and complexity of developing, deploying, and managing multi-tier, server-centric applications. Java EE builds upon the Java SE platform and provides a set of APIs (application programming interfaces) for developing and running portable, robust, scalable, reliable and secure server-side applications.

Some of the fundamental components of Java EE include:

- Enterprise JavaBeans (EJB): a managed, server-side component architecture used to encapsulate the business logic of an application. EJB technology enables rapid and simplified development of distributed, transactional, secure and portable applications based on Java technology.
- Java Persistence API (JPA): a framework that allows developers to manage data using object-relational mapping (ORM) in applications built on the Java Platform.

JavaScript and Ajax Development

JavaScript is an object-oriented scripting language primarily used in client-side interfaces for web applications. Ajax (Asynchronous JavaScript and XML) is a Web 2.0 technique that allows changes to occur in a web page without the need to perform a page refresh. JavaScript toolkits can be leveraged to implement Ajax-enabled components and functionality in web pages.

Web Server and Client

Web Server is a software that can process the client request and send the response back to the client. For example, Apache is one of the most widely used web server. Web Server runs on some physical machine and listens to client request on specific port.

A web client is a software that helps in communicating with the server. Some of the most widely used web clients are Firefox, Google Chrome, Safari etc. When we request something from server (through URL), web client takes care of creating a request and sending it to server and then parsing the server response and present it to the user.

HTML and HTTP

Web Server and Web Client are two separate softwares, so there should be some common language for communication. HTML is the common language between server and client and stands for **H**yper**T**ext **M**arkup **L**anguage.

Web server and client needs a common communication protocol, HTTP (**H**yper**T**ext **T**ransfer **P**rotocol) is the communication protocol between server and client. HTTP runs on top of TCP/IP communication protocol.

Some of the important parts of HTTP Request are:

- **HTTP Method** – action to be performed, usually GET, POST, PUT etc.
- **URL** – Page to access
- **Form Parameters** – similar to arguments in a java method, for example user,password details from login page.

Sample HTTP Request:

1GET /FirstServletProject/jsps/hello.jsp HTTP/1.1

2Host: localhost:8080

3Cache-Control: no-cache

Some of the important parts of HTTP Response are:

- **Status Code** – an integer to indicate whether the request was success or not. Some of the well known status codes are 200 for success, 404 for Not Found and 403 for Access Forbidden.
- **Content Type** – text, html, image, pdf etc. Also known as MIME type
- **Content** – actual data that is rendered by client and shown to user.

MIME Type or Content Type: If you see above sample HTTP response header, it contains tag “Content-Type”. It’s also called MIME type and server sends it to client to let them know the kind of data it’s sending. It helps client in rendering the data for user. Some of the mostly used mime types are text/html, text/xml, application/xml etc.

Understanding URL

URL is acronym of Universal Resource Locator and it’s used to locate the server and resource. Every resource on the web has it’s own unique address. Let’s see parts of URL with an example.

http://localhost:8080/FirstServletProject/jsps/hello.jsp

http:// – This is the first part of URL and provides the communication protocol to be used in server-client communication.

localhost – The unique address of the server, most of the times it’s the hostname of the server that maps to unique IP address. Sometimes multiple hostnames point to same IP addresses and web server virtual host takes care of sending request to the particular server instance.

8080 – This is the port on which server is listening, it’s optional and if we don’t provide it in URL then request goes to the default port of the protocol. Port numbers 0 to 1023 are reserved ports for well known services, for example 80 for HTTP, 443 for HTTPS, 21 for FTP etc.

FirstServletProject/jsps/hello.jsp – Resource requested from server. It can be static html, pdf, JSP, servlets, PHP etc.

MySQL:

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation.
The MySQL Web site (http://www.mysql.com/) provides the latest information about MySQL software.

- **MySQL is a database management system.**
  A database is a structured collection of data. It may be anything from a simple shopping list to a picture gallery or the vast amounts of information in a corporate network. To add, access, and process data stored in a computer database, you need a database management system such as MySQL Server. Since computers are very good at handling large amounts of data, database management systems play a central role in computing, as standalone utilities, or as parts of other applications.

- **MySQL databases are relational.**
  A relational database stores data in separate tables rather than putting all the data in one big storeroom. The database structures are organized into physical files optimized for speed. The logical model, with objects such as databases, tables, views, rows, and columns, offers a flexible programming environment. You set up rules governing the relationships between different data fields, such as one-to-one, one-to-many, unique, required or optional, and “pointers” between different tables. The database enforces these rules, so that with a well-designed database, your application never sees inconsistent, duplicate, orphan, out-of-date, or missing data.

The SQL part of “MySQL” stands for “Structured Query Language”. SQL is the most common standardized language used to access databases. Depending on your programming environment, you might enter SQL directly (for example, to generate reports), embed SQL statements into code written in another language, or use a language-specific API that hides the SQL syntax.

SQL is defined by the ANSI/ISO SQL Standard. The SQL standard has been evolving since 1986 and several versions exist. In this manual, “SQL-92” refers to the standard released in 1992, “SQL:1999” refers to the standard released in 1999, and “SQL:2003” refers to the current version of the standard. We use the phrase “the SQL standard” to mean the current version of the SQL Standard at any time.

- **MySQL software is Open Source.**
  Open Source means that it is possible for anyone to use and modify the software. Anybody can download the MySQL software from the Internet and use it without paying anything. If you wish, you may study the source code and change it to suit your needs. The MySQL software uses the GPL (GNU General Public License), http://www.fsf.org/licenses/, to define what you may and may not do with the software in different situations. If you feel uncomfortable with the GPL or need to embed MySQL code into a commercial application, you can buy a commercially licensed version from us. See the MySQL Licensing Overview for more information (http://www.mysql.com/company/legal/licensing/).

- **The MySQL Database Server is very fast, reliable, scalable, and easy to use.**
  If that is what you are looking for, you should give it a try. MySQL Server can run comfortably on a desktop or laptop, alongside your other applications, web servers, and so on, requiring little or no attention. If you dedicate an entire machine to MySQL, you can adjust the settings to take advantage of all the memory, CPU power, and I/O capacity available. MySQL can also scale up to clusters of machines, networked together.
  You can find a performance comparison of MySQL Server with other database managers on our benchmark page.

MySQL Server was originally developed to handle large databases much faster than existing solutions and has been successfully used in highly demanding production environments for several years. Although under constant development, MySQL Server today offers a rich and useful set of functions. Its connectivity, speed, and security make MySQL Server highly suited for accessing databases on the Internet.

- **MySQL Server works in client/server or embedded systems.**

  The MySQL Database Software is a client/server system that consists of a multi-threaded SQL server that supports different backends, several different client programs and libraries, administrative tools, and a wide range of application programming interfaces (APIs).

  We also provide MySQL Server as an embedded multi-threaded library that you can link into your application to get a smaller, faster, easier-to-manage standalone product.

- **A large amount of contributed MySQL software is available.**

  MySQL Server has a practical set of features developed in close cooperation with our users. It is very likely that your favorite application or language supports the MySQL Database Server.

The official way to pronounce “MySQL” is “My Ess Que Ell” (not “my sequel”), but we do not mind if you pronounce it as “my sequel” or in some other localized way.

PROPOSED SYSTEM

3. PROPOSED SYSTEM

We propose an efficient and secure cloudbased IoT data management scheme using our novel CPABE construction. The proposed scheme features efficient storage and bandwidth management, and capability of tracing traitors if they illegally share their secret keys. The cloud server can perform a significant part of the computational overhead related to decryption via a user-specific transformation key. This key is crucial in forbidding unauthorized access by a shared (or leaked) key holder because the proposed access mechanism of the IoT data works as follows: the cloud (1) authenticates the identity of the key holder, (2) partially decrypts the ciphertext using the key holder's transformation key, and (3) returns the partially decrypted result. Note that the transformation key is closely associated with the key holder, who can recover the plaintext from the partially decrypted ciphertext if and only if he is the original owner of the attribute key. Therefore, the other users who illegally hold the shared (or leaked) keys are unable to obtain the plaintext.

System Architecture:

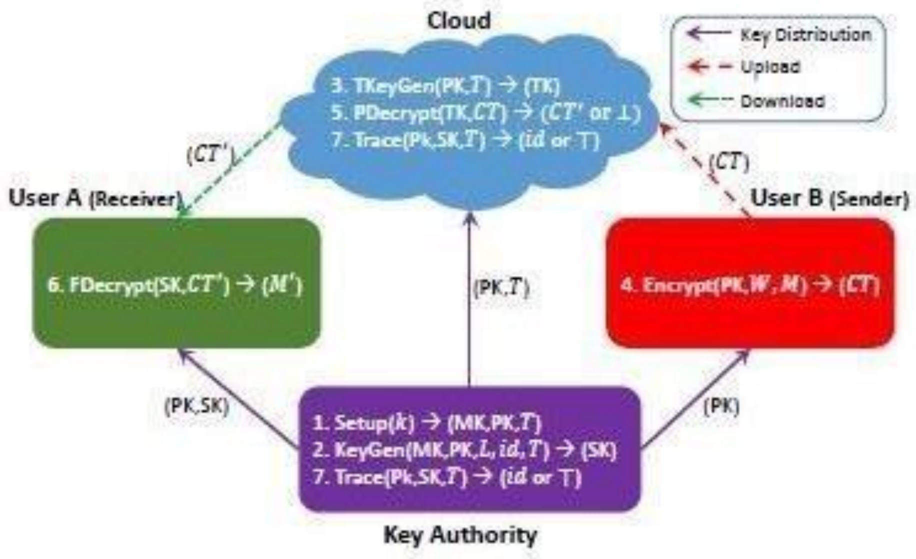

Cloud
3. TKeyGen(PK,T) → (TK)
5. PDecrypt(TK,CT) → (CT' or ⊥)
7. Trace(Pk,SK,T) → (id or ⊤)
Key Distribution
Upload
Download
(CT')
(CT)
User A (Receiver)
User B (Sender)
6. FDecrypt(SK,CT') → (M')
(PK,T)
4. Encrypt(PK,W, M) → (CT)
(PK,SK)
(PK)
1. Setup(k) → (MK,PK,T)
2. KeyGen(MK,PK,L, id, T) → (SK)
7. Trace(Pk,SK,T) → (id or ⊤)
Key Authority

Fig:4 System Architecture

3.1 SYSTEM STUDY

3.1.1 FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ♦ ECONOMICAL FEASIBILITY
- ♦ TECHNICAL FEASIBILITY
- ♦ SOCIAL FEASIBILITY

3.1.2 ECONOMICAL FEASIBILITY

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

3.1.3 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.1.4 SOCIAL FEASIBILITY

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

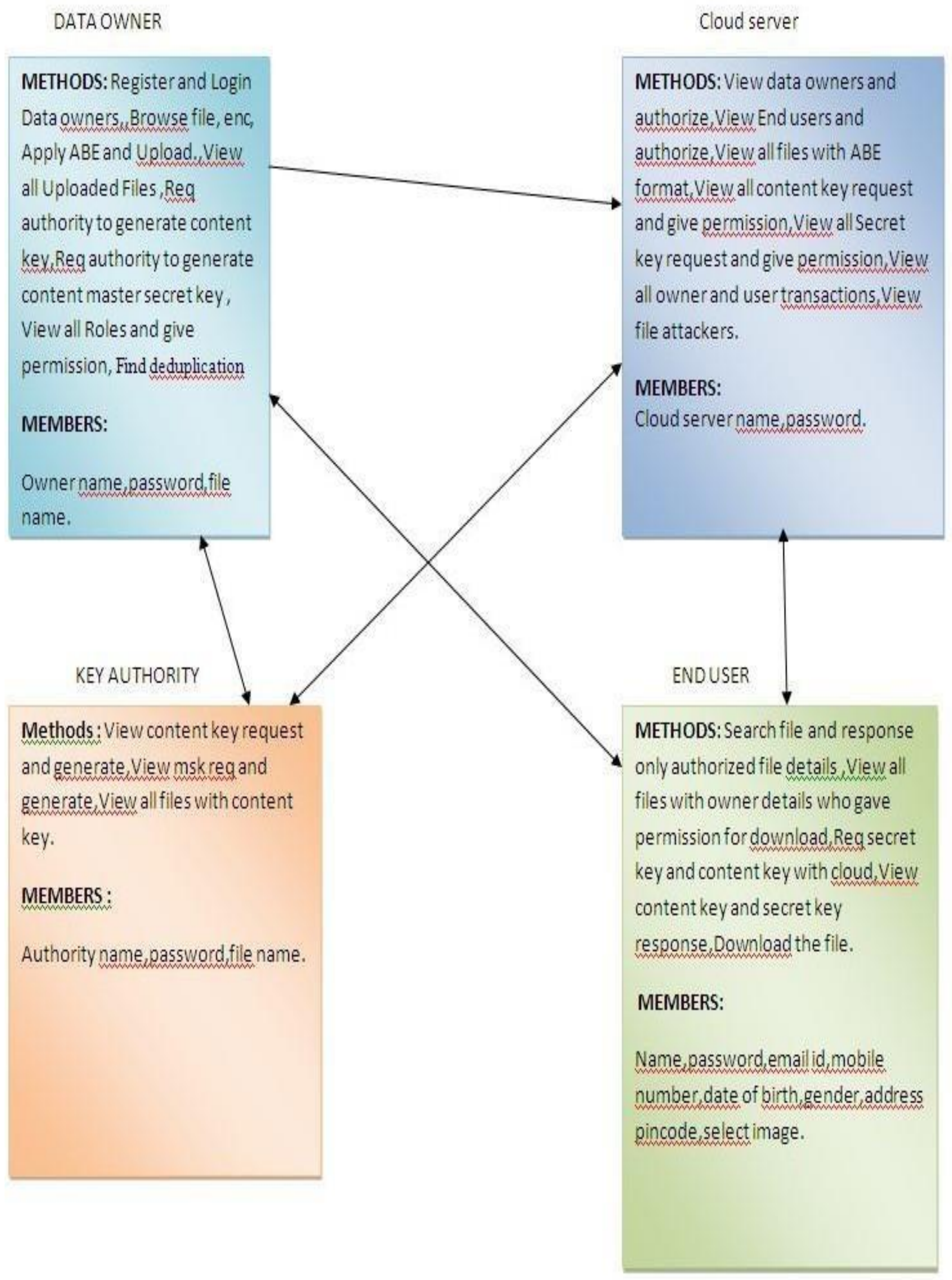
3.2 SYSTEM DESIGN

CLASS DIAGRAM:

A Unified Modeling Langue (UML) class diagram is a sort of static structure diagram for software engineering which describes the structure of a system by showing system classes, attributes, operations (or methods) and class relationships. This defines which class contains knowledge.

The list of classes used in project are:

(1) Data Owner
(2) Cloud Server
(3) Login
(4) Register
(5) Trapdoor Generator
(6) End user


DATA OWNER
METHODS: Register and Login Data owners,,Browse file, enc, Apply ABE and Upload.,View all Uploaded Files ,Req authority to generate content key,Req authority to generate content master secret key , View all Roles and give permission, Find deduplication
MEMBERS:
Owner name,password,file name.
Cloud server
METHODS: View data owners and authorize,View End users and authorize,View all files with ABE format,View all content key request and give permission,View all Secret key request and give permission,View all owner and user transactions,View file attackers.
MEMBERS:
Cloud server name,password.
KEY AUTHORITY
Methods : View content key request and generate,View msk req and generate,View all files with content key.
MEMBERS :
Authority name,password,file name.
END USER
METHODS: Search file and response only authorized file details ,View all files with owner details who gave permission for download,Req secret key and content key with cloud,View content key and secret key response,Download the file.
MEMBERS:
Name,password,email id,mobile number,date of birth,gender,address pincode,select image.

DATA FLOW DIAGRAM

1. The bubble diagram is also called DFD. It is a simple graphical method that can constitute a system in respect of the system's input data, specific processing and generation of the system's output data.

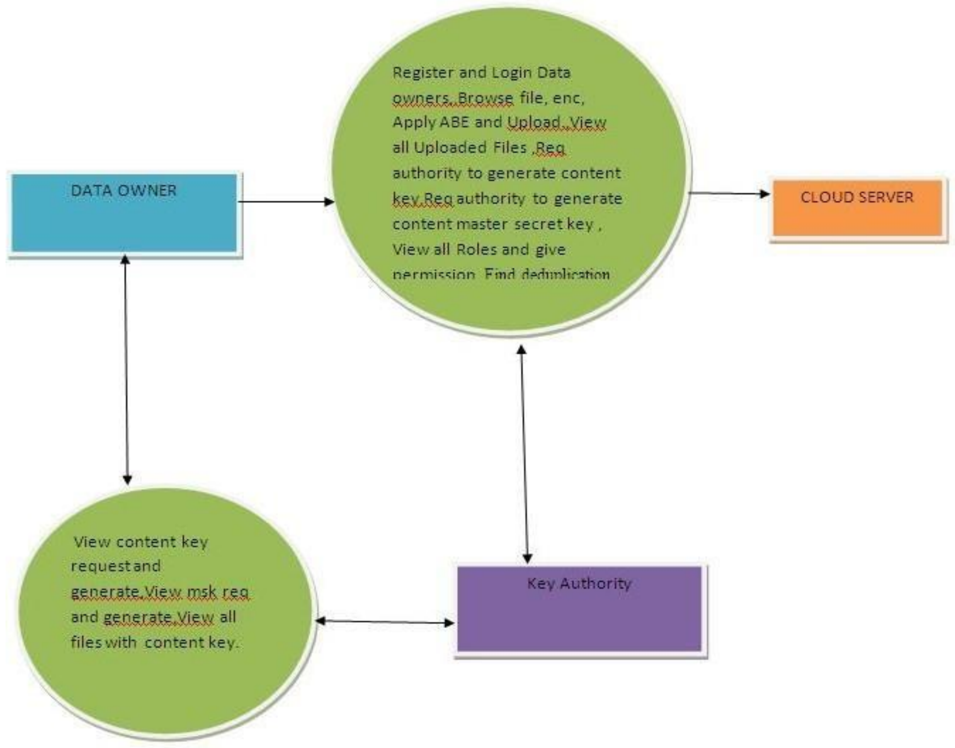
2. One of the most important modeling tools is the DFD (Data Flow Diagram). The machine components are modeled. Such components are the system process, the process data, an external entity that communicates with the system and the flow of information.

3. DFD shows how the data moves through the system and how a number of transformations modify the system. This is an information processing and transformations method used as data travels from input to output. It's an information management method

The bubble diagram is also referred as DFD. A DFD can be used in any abstraction level to describe a device. DFD can be distributed into levels reflecting an enhanced flow of knowledge and functional detail.

Data Flow Diagram

Level-0

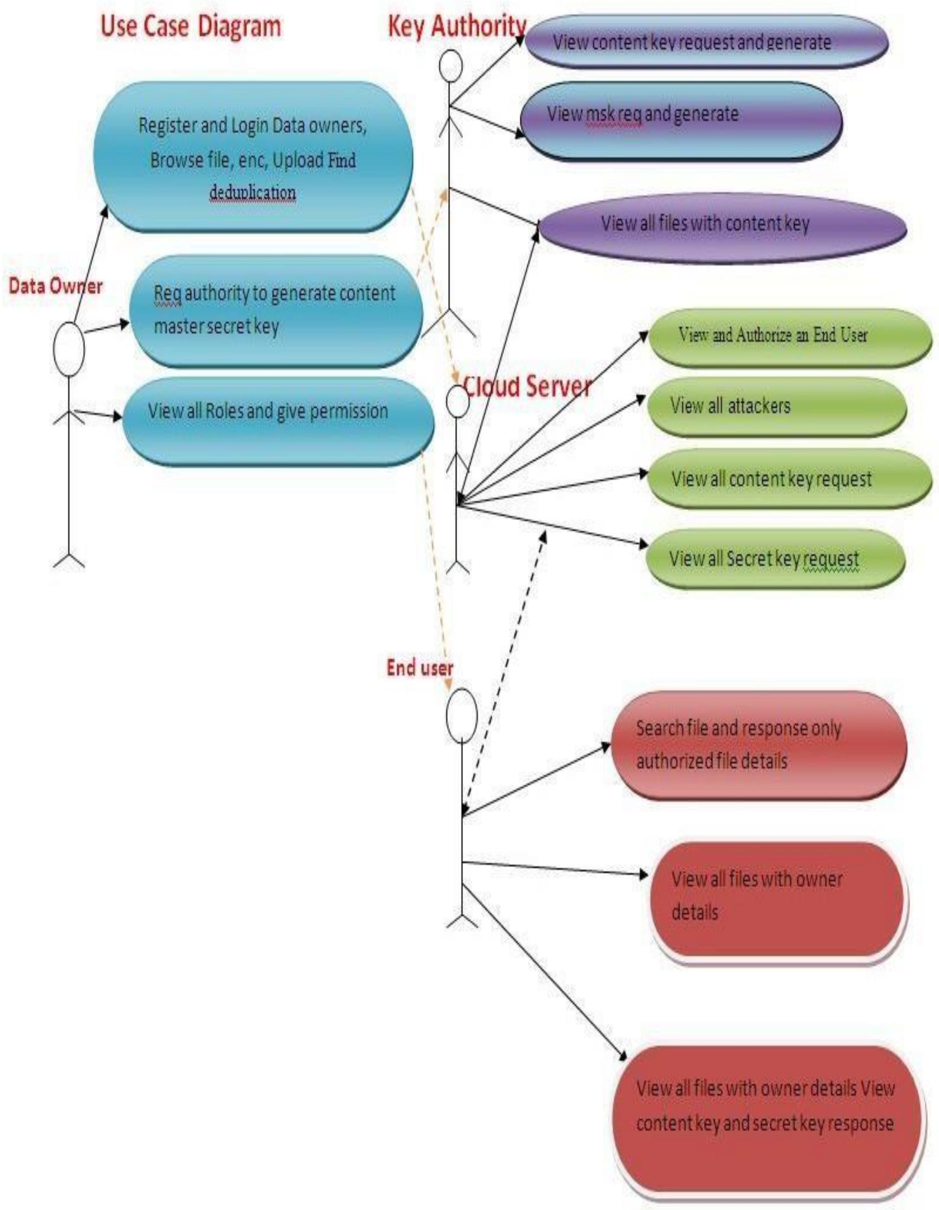

DATA OWNER
Register and Login Data owners, Browse file, enc, Apply ABE and Upload, View all Uploaded Files ,Req authority to generate content key,Req authority to generate content master secret key , View all Roles and give permission, Find deduplication
CLOUD SERVER
View content key request and generate,View msk req and generate,View all files with content key.
Key Authority

3.3 UML DIAGRAMS:

USE CASE DIAGRAM

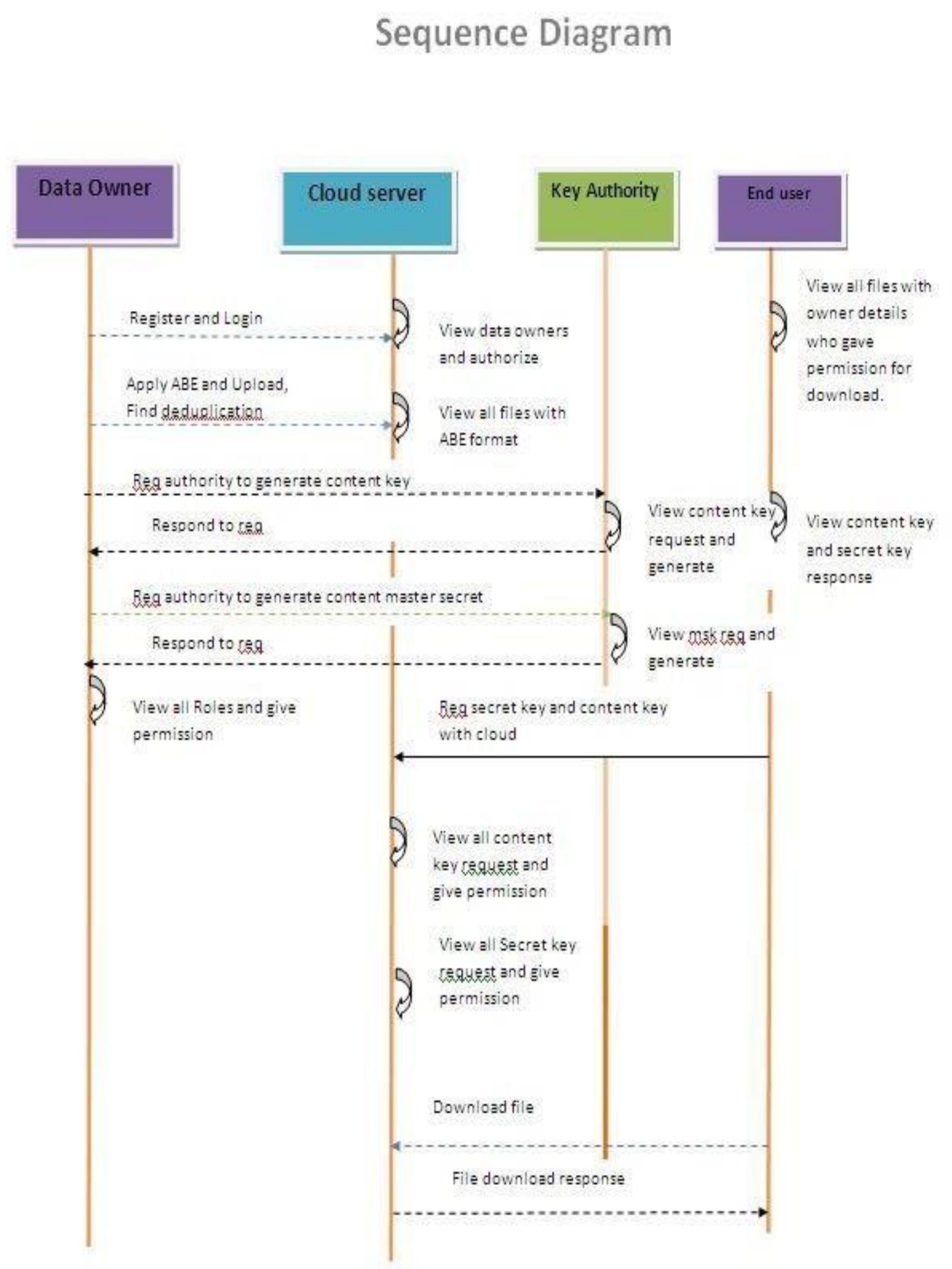
It represents the behavioural aspects of the system. The list of actors used in this system is:

(1) Cloud Server
(2) Data Owner
(3) End user


Use Case Diagram
Key Authority
View content key request and generate
View msk req and generate
Register and Login Data owners, Browse file, enc, Upload Find deduplication
View all files with content key
Data Owner
Req authority to generate content master secret key
View and Authorize an End User
View all Roles and give permission
Cloud Server
View all attackers
View all content key request
View all Secret key request
End user
Search file and response only authorized file details
View all files with owner details
View all files with owner details View content key and secret key response

SEQUENCE DIAGRAM

A sequence diagram in UML is a sort of interaction diagram showing how and in what sequencer processes work. A Message Sequence Map construction. Sequence diagrams also are referred to as event maps, event scenarios, and time charts.


Sequence Diagram
Data Owner
Cloud server
Key Authority
End user
Register and Login
View data owners and authorize
Apply ABE and Upload, Find deduplication
View all files with ABE format
View all files with owner details who gave permission for download.
Req authority to generate content key
Respond to req
View content key request and generate
View content key and secret key response
Req authority to generate content master secret
Respond to req
View msk req and generate
View all Roles and give permission
Req secret key and content key with cloud
View all content key request and give permission
View all Secret key request and give permission
Download file
File download response

3.4 IMPLEMENTATION

DATA OWNER:

In this module, initially the data owner has to register to the cloud server and get authorized. After the authorization from cloud data owner will encrypt and add file to the cloud server where in after the addition of file data owner requests the content key and the master secret key to the authority for the file he uploaded and finds Find deduplication ,only after the keys generated the file is uploaded to the cloud server. After the uploading of the file the data owner will have to provide download and the search permission for individual file for the users to perform search and download.

CLOUD SERVER

The cloud server manages a cloud to provide data storage service. Data owners encrypt their data files and store them in the cloud for sharing with cloud End users. To access the shared data files users will request the permission of content key and the MSK master secret key. And the cloud will provide the permission .and also views all the transactions and attackers related to the files.

KEY AUTHORITY

Key Authority generates the content key and the secret key requested by the end user. KeyAuthority can view all files with the content key and master secret key generated with the corresponding data owner details of the particular file.

END USER

User has to register and login for accessing the files in the cloud. User is authorized by the cloud to verify the registration. User has to request for the MSK master secret key and content key to download the file. Users can only download and search the file if the data owner of the particular file has provided the permissions.

SAMPLE CODE

4. SAMPLE CODE

```
package distributeddeduplication;
import javax.swing.JFrame;
public class Owner extends javax.swing.JFrame {
   public Owner() {
      initComponents();
      jLabel6.setVisible(false);
   }
private void initComponents() {
      jLabel1  =  new  javax.swing.JLabel();
      jLabel2  =  new  javax.swing.JLabel();
      jLabel3 = new javax.swing.JLabel();
      jTextField1 = new javax.swing.JTextField();
      jPasswordField1 = new javax.swing.JPasswordField();
      jButton1 = new javax.swing.JButton();
      jButton2 = new javax.swing.JButton();
      jLabel4 = new javax.swing.JLabel();
      jLabel5 = new javax.swing.JLabel();
      jLabel6 = new javax.swing.JLabel();
      jLabel7 = new javax.swing.JLabel();
      jLabel8 = new javax.swing.JLabel();
      jLabel9 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setMaximumSize(new   java.awt.Dimension(600,   380));
      setPreferredSize(new java.awt.Dimension(600, 380));
      getContentPane().setLayout(null);
      jLabel1.setFont(new java.awt.Font("Perpetua Titling MT", 1, 18));
      jLabel1.setText("Owner      Login");
      getContentPane().add(jLabel1);
      jLabel1.setBounds(180, 40, 180, 40);
      jLabel2.setFont(new java.awt.Font("Perpetua Titling MT", 1, 12));
```

```
jLabel2.setText("Username");
getContentPane().add(jLabel2);
jLabel2.setBounds(70, 110, 100, 40);

jLabel3.setFont(new java.awt.Font("Perpetua Titling MT", 1, 12));
jLabel3.setText("Password");
getContentPane().add(jLabel3);
jLabel3.setBounds(70, 170, 90, 40);
getContentPane().add(jTextField1);
jTextField1.setBounds(180, 110, 180, 30);
getContentPane().add(jPasswordField1);
jPasswordField1.setBounds(180, 170, 180, 30);
jButton1.setFont(new java.awt.Font("Times New Roman", 1, 14));
jButton1.setText("Login");
jButton1.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton1ActionPerformed(evt);
   }
});
getContentPane().add(jButton1);
jButton1.setBounds(130, 240, 90, 40);
jButton2.setFont(new java.awt.Font("Times New Roman", 1, 14));
jButton2.setText("Reset");
jButton2.addActionListener(new java.awt.event.ActionListener() {
   public void actionPerformed(java.awt.event.ActionEvent evt) {
      jButton2ActionPerformed(evt);
   }
});
getContentPane().add(jButton2);
jButton2.setBounds(270, 240, 90, 40);
jLabel4.setFont(new java.awt.Font("Times New Roman", 1, 12));
jLabel4.setText("New User");
getContentPane().add(jLabel4);
jLabel4.setBounds(180, 310, 60, 20);
```

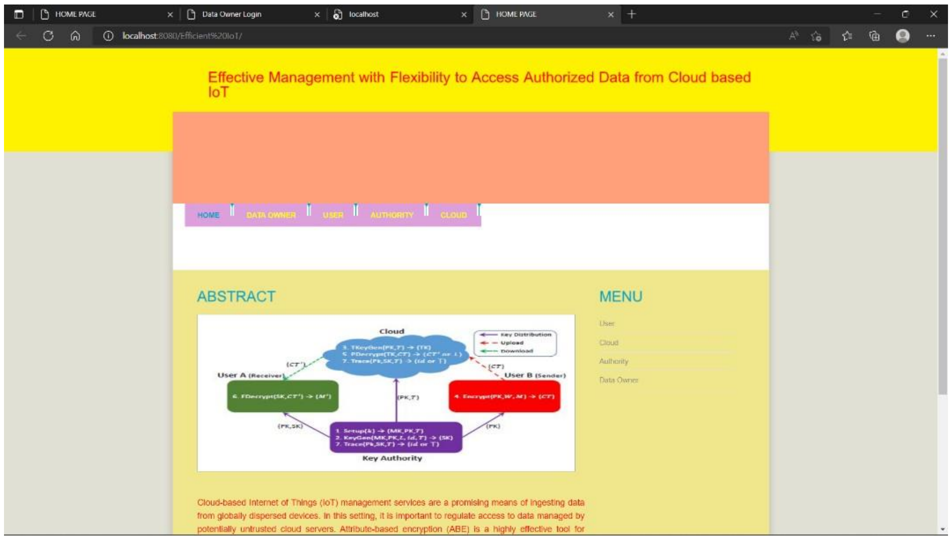
```
    jLabel5.setFont(new java.awt.Font("Times New Roman", 3, 12));
    jLabel5.setText("Click Here");
    jLabel5.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jLabel5MouseClicked(evt);
        }
    });
    getContentPane().add(jLabel5);
    jLabel5.setBounds(240, 310, 60, 30);
    jLabel6.setForeground(new java.awt.Color(255, 51, 102));
    jLabel6.setText("jLabel6");
    getContentPane().add(jLabel6);
    jLabel6.setBounds(150, 80, 390, 20);
    jLabel7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/owner.png")));
    getContentPane().add(jLabel7);
    jLabel7.setBounds(390, 110, 130, 150);
    jLabel8.setFont(new java.awt.Font("Times New Roman", 1, 14));
    jLabel8.setForeground(new java.awt.Color(255, 0, 102));
    jLabel8.setText("Exit");
    jLabel8.addMouseListener(new java.awt.event.MouseAdapter() {
        public void mouseClicked(java.awt.event.MouseEvent evt) {
            jLabel8MouseClicked(evt);
        }
    });
    getContentPane().add(jLabel8);
    jLabel8.setBounds(470, 30, 24, 17);
    jLabel9.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/backgound.jpg")));
    jLabel9.setText("jLabel9");
    getContentPane().add(jLabel9);
    jLabel9.setBounds(0, 0, 580, 350);
    setSize(new java.awt.Dimension(590, 383));
    setLocationRelativeTo(null);
}
private void jLabel5MouseClicked(java.awt.event.MouseEvent evt) {
    dispose();
    new Reg().setVisible(true);
}
public void setTitle() {
    new Owner().setTitle("Owner Login");
}
private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
    jTextField1.setText("");
    jPasswordField1.setText("");
    jLabel6.setText("");
}
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
            if (jTextField1.getText() == null || jTextField1.getText().trim().equals("")) {
        jLabel6.setText("Please Enter    Username");
        jLabel6.setVisible(true);
    }        else        if        (jPasswordField1.getText()        ==        null        ||
jPasswordField1.getText().trim().equals("")) {
             jLabel6.setText("Please Enter    Password");
        jLabel6.setVisible(true);
    } else {
        RegValidation rv = new RegValidation();
        String            status            =            rv.userValidate(jTextField1.getText(),
jPasswordField1.getText(), 1);
        if (status.equals("Success")) {
            dispose();
            new Ohome(jTextField1.getText()).setVisible(true);
        } else {
            jLabel6.setText("Please Enter Correct Username and Password");
            jLabel6.setVisible(true);
        }
    }
}
```

```
   private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
            dispose();
      new Main().setVisible(true);
   }
   public static void main(String args[]) {
      try {
         for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
            if ("Nimbus".equals(info.getName())) {
               javax.swing.UIManager.setLookAndFeel(info.getClassName());
               break;
            }
         }
      } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Owner.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
      } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Owner.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
      } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Owner.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {

java.util.logging.Logger.getLogger(Owner.class.getName()).log(java.util.logging.Lev
el.SEVERE, null, ex);
      }
      java.awt.EventQueue.invokeLater(new Runnable() {
         public void run() {
            new Owner().setVisible(true);
            new Owner().setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
         }
      });
   }
   private javax.swing.JButton jButton1;
   private javax.swing.JButton jButton2;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JLabel jLabel3;
   private javax.swing.JLabel jLabel4;
   private javax.swing.JLabel jLabel5;
   private javax.swing.JLabel jLabel6;
   private javax.swing.JLabel jLabel7;
   private javax.swing.JLabel jLabel8;
   private javax.swing.JLabel jLabel9;
   private javax.swing.JPasswordField jPasswordField1;
   private javax.swing.JTextField jTextField1;
}
package distributeddeduplication;
import javax.swing.JFrame;
public class Main extends javax.swing.JFrame {
   public Main() {
      initComponents();
   }
BEGIN:initComponents
   private void initComponents()  {
      jPanel1 = new javax.swing.JPanel();
      jLabel1 = new javax.swing.JLabel();
      jLabel2 = new javax.swing.JLabel();
      jPanel2 = new javax.swing.JPanel();
      jLabel3 = new javax.swing.JLabel();
      jLabel4 = new javax.swing.JLabel();
      jLabel5 = new javax.swing.JLabel();
      jLabel6 = new javax.swing.JLabel();
      jLabel7 = new javax.swing.JLabel();
      jLabel8 = new javax.swing.JLabel();

setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
      setMaximumSize(new java.awt.Dimension(750, 450));
      setPreferredSize(new java.awt.Dimension(750, 450));
      getContentPane().setLayout(null);
      jPanel1.setBackground(new java.awt.Color(255, 255, 255));
      jPanel1.setLayout(null);
      jLabel1.setFont(new java.awt.Font("Perpetua Titling MT", 1, 24));
      jLabel1.setForeground(new java.awt.Color(255, 0, 51));
      jLabel1.setText("Normalization of Duplicate Records");
      jPanel1.add(jLabel1);
      jLabel1.setBounds(60, 10, 610, 60);
      jLabel2.setFont(new java.awt.Font("Perpetua Titling MT", 1, 24));
      jLabel2.setForeground(new java.awt.Color(255, 102, 0));
      jLabel2.setText("from Multiple Sources");
      jPanel1.add(jLabel2);
      jLabel2.setBounds(160, 60, 420, 60);
      getContentPane().add(jPanel1);
      jPanel1.setBounds(10, 20, 710, 120);
      jPanel2.setBackground(new java.awt.Color(255, 255, 255));
      jPanel2.setLayout(null);
      jLabel3.setFont(new java.awt.Font("Perpetua Titling MT", 1, 14));
      jLabel3.setText("Owner");
      jPanel2.add(jLabel3);
      jLabel3.setBounds(80, 30, 80, 30);
      jLabel4.setFont(new java.awt.Font("Perpetua Titling MT", 1, 14));
      jLabel4.setText("User");
      jPanel2.add(jLabel4);
      jLabel4.setBounds(270, 30, 70, 30);
      jLabel5.setFont(new java.awt.Font("Perpetua Titling MT", 1, 12));
      jLabel5.setText("Cloud");
      jPanel2.add(jLabel5);
      jLabel5.setBounds(450, 30, 60, 30);
      jLabel6.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/user.png")));
      jLabel6.addMouseListener(new java.awt.event.MouseAdapter() {
          public void mouseClicked(java.awt.event.MouseEvent evt) {
              jLabel6MouseClicked(evt);
          }
      });
      jPanel2.add(jLabel6);
      jLabel6.setBounds(220, 60, 160, 150);
      jLabel7.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/owner.png")));
      jLabel7.addMouseListener(new java.awt.event.MouseAdapter() {
          public void mouseClicked(java.awt.event.MouseEvent evt) {
              jLabel7MouseClicked(evt);
          }
      });
      jPanel2.add(jLabel7);
      jLabel7.setBounds(40, 60, 160, 150);
      jLabel8.setIcon(new
javax.swing.ImageIcon(getClass().getResource("/images/cloud.png")));
      jLabel8.addMouseListener(new java.awt.event.MouseAdapter() {
          public void mouseClicked(java.awt.event.MouseEvent evt) {
              jLabel8MouseClicked(evt);
          }
      });
      jPanel2.add(jLabel8);
      jLabel8.setBounds(410, 60, 160, 150);
      getContentPane().add(jPanel2);
      jPanel2.setBounds(70, 170, 590, 210);
      setSize(new java.awt.Dimension(746, 438));
      setLocationRelativeTo(null);
  }
  private void jLabel7MouseClicked(java.awt.event.MouseEvent evt) {
      dispose();
      new Owner().setVisible(true);
  }
  private void jLabel6MouseClicked(java.awt.event.MouseEvent evt) {
      dispose();
      new User().setVisible(true);
  }
  private void jLabel8MouseClicked(java.awt.event.MouseEvent evt) {
      dispose();
      new Cloud().setVisible(true);
  }
  public static void main(String args[]) {
      try {
          for (javax.swing.UIManager.LookAndFeelInfo info :
javax.swing.UIManager.getInstalledLookAndFeels()) {
              if ("Nimbus".equals(info.getName())) {
                  javax.swing.UIManager.setLookAndFeel(info.getClassName());
                  break;
              }
          }
      } catch (ClassNotFoundException ex) {

java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
      } catch (InstantiationException ex) {

java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
      } catch (IllegalAccessException ex) {

java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
      } catch (javax.swing.UnsupportedLookAndFeelException ex) {
```

```
  java.util.logging.Logger.getLogger(Main.class.getName()).log(java.util.logging.Level
.SEVERE, null, ex);
        }
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new Main().setVisible(true);
                new  Main().setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            }
        });
    }
    private  javax.swing.JLabel  jLabel1;
    private  javax.swing.JLabel  jLabel2;
    private  javax.swing.JLabel  jLabel3;
    private  javax.swing.JLabel  jLabel4;
    private  javax.swing.JLabel  jLabel5;
    private  javax.swing.JLabel  jLabel6;
    private  javax.swing.JLabel  jLabel7;
    private  javax.swing.JLabel  jLabel8;
    private  javax.swing.JPanel  jPanel1;
    private javax.swing.JPanel jPanel2;
}
```

SCREEN SHOTS

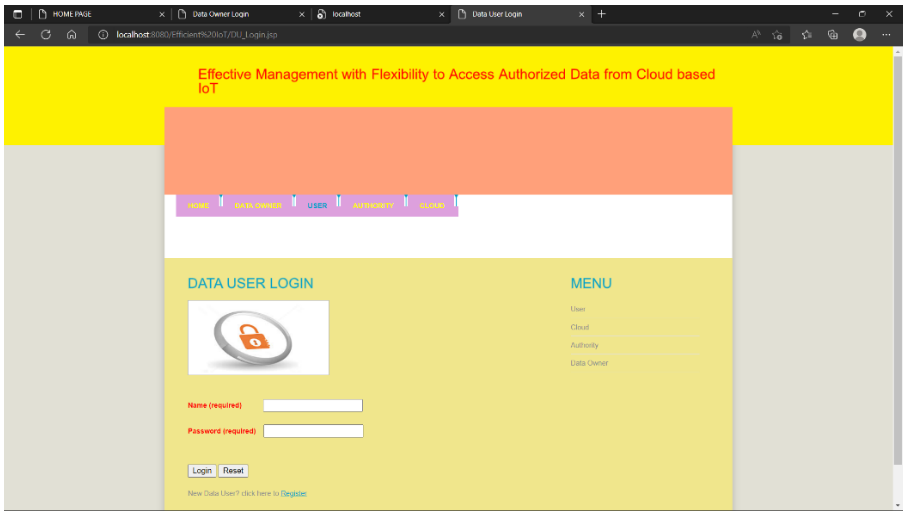
5. SCREEN SHOTS



Effective Management with Flexibility to Access Authorized Data from Cloud based IoT

ABSTRACT

Cloud-based Internet of Things (IoT) management services are a promising means of ingesting data from globally dispersed devices. In this setting, it is important to regulate access to data managed by potentially untrusted cloud servers. Attribute-based encryption (ABE) is a highly effective tool for



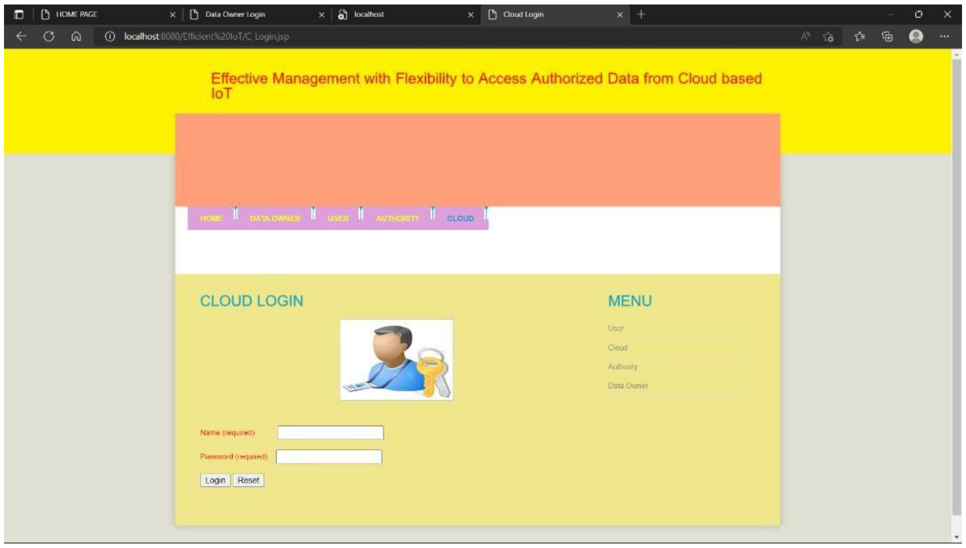
Effective Management with Flexibility to Access Authorized Data from Cloud based IoT

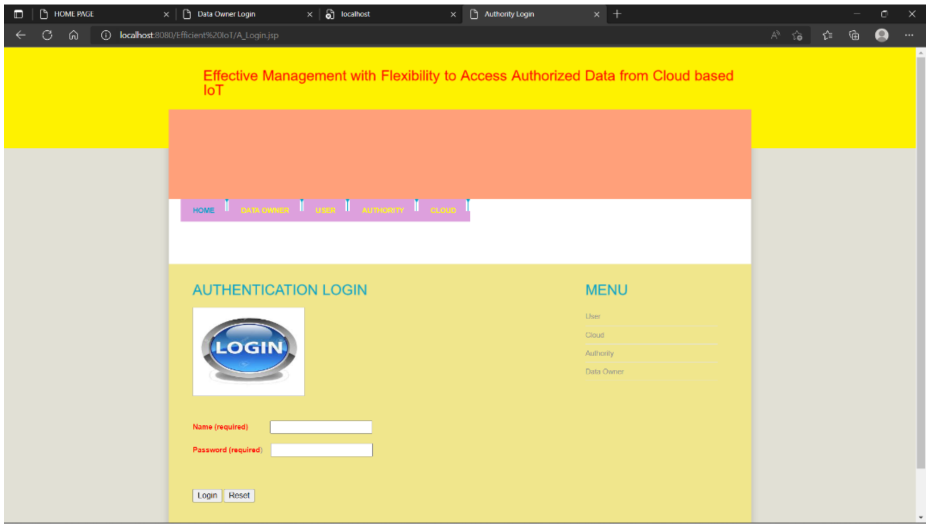
DATA USER LOGIN

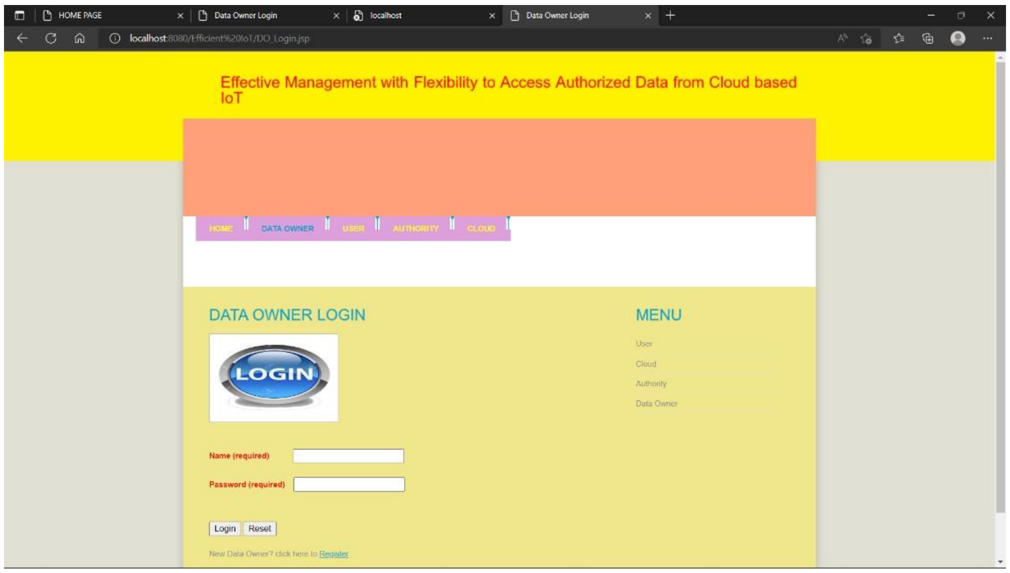
Name (required)

Password (required)

New Data User? click here to Register


HOME PAGE
Data Owner Login
localhost
Cloud Login
localhost:8080/Efficient%20IoT/C_Login.jsp
Effective Management with Flexibility to Access Authorized Data from Cloud based IoT
HOME
DATA OWNER
USER
AUTHORITY
CLOUD
CLOUD LOGIN
MENU
User
Cloud
Authority
Data Owner
Name (required)
Password (required)
Login
Reset


HOME PAGE
Data Owner Login
localhost
Authority Login
localhost:8080/Efficient%20IoT/A_Login.jsp
Effective Management with Flexibility to Access Authorized Data from Cloud based IoT
HOME
DATA OWNER
USER
AUTHORITY
CLOUD
AUTHENTICATION LOGIN
LOGIN
MENU
User
Cloud
Authority
Data Owner
Name (required)
Password (required)
Login
Reset



Effective Management with Flexibility to Access Authorized Data from Cloud based IoT

DATA OWNER LOGIN

Name (required)

Password (required)

New Data Owner? click here to Register

TESTING METHODS

1. SYSTEM TESTING

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

TESTING METHODOLOGIES

6.1 Unit testing

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

6.2 Integration testing

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

The following are the types of Integration Testing:

1. Top Down Integration

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

2. Bottom-up Integration

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.

- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

6.3 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.

6.4 System Testing

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

6.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

6.6 Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box
.you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.7 Output Testing

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

6.8 Validation Checking

Validation checks are performed on the following fields.

Text Field:

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

Numeric Field:

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested.

A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

CONCLUSION

CONCLUSION

We propose a novel CP-ABE scheme which is deployable to practical cloud-based IoT management systems. The proposed scheme is efficient in terms of communication cost because the ciphertext size generated by IoT devices is constant, irrespective of the number of attributes. Additionally, the proposed scheme allows battery-powered user devices to offload a significant amount of decryption operations to the cloud. Finally, the proposed scheme supports key traceability and prevents illegally shared key holders from accessing the outsourced data, while the correct decryption can be achieved only by the original key owner. Thus, the proposed scheme is resilient to forensically intractable key abuse attacks, which are practical in IoT systems.

REFERENCES

REFERENCES

[1] “AWS IoT.” AWS. https://aws.amazon.com/ko/iot. (accessed Dec. 17, 2019)

[2] “Cloud IoT Core.” Google Cloud. https://cloud.google.com/iotcore. (accessed Dec. 17, 2019)

[3] A. Sahai, B. Waters, “Fuzzy identity-based encryption,” Advances in Cryptology-EUROCRYPT, 2005, pp. 457–473.

[4] J. Bethencourt, A. Sahai, B. Waters, “Ciphertext-policy attributebased encryption,” In IEEE symposium on security and privacy (SP’07), 2007, pp. 321–334.

[5] M. Green, S. Hohenberger, B. Waters, “Outsourcing the decryption of ABE ciphertexts,” In USENIX Security Symposium, Vol. 2011, No. 3, 2001.

[6] J. Lai, R. H. Deng, C. Guan, J. Weng, “Attribute-based encryption with verifiable outsourced decryption,” IEEE Transactions on Information Forensics and Security, 8(8), 2013, pp. 1343–1354.

[7] S. Lin, R. Zhang, H. Ma, M. Wang, “Revisiting attribute-based encryption with verifiable outsourced decryption,” IEEE Transactions on Information Forensics and Security, 10(10), 2015, pp. 2119–2130.

[8] S. Soursos, I. P. Zarko, P. Zwickl, I. Gojmerac, G. Bianchi, G. ˇ Carrozzo, “Towards the cross-domain interoperability of IoT platforms,” In European Conference on Networks and Communications (EuCNC), 2016, pp. 398–402.

[9] C. Chen, Z. Zhang, D. Feng, “Efficient ciphertext policy attributebased encryption with constant-size ciphertext and constant computation-cost,” In Provable Security, 2011, pp. 84–101.

[10] C. Hahn, H. Kwon, J. Hur, “Efficient attribute-based secure data sharing with hidden policies and traceability in mobile health networks,” Mobile Information Systems, 2016.

[11] Z. Zhou, D. Huang, Z. Wang, “Efficient privacy-preserving ciphertext-policy attribute based-encryption and broadcast encryption,” IEEE Transactions on Computers, 64(1), 2015, pp. 126–138.

[12] Z. Zhou, D. Huang, “On efficient ciphertext-policy attribute based encryption and broadcast encryption,” In Proceedings of the 17th ACM conference on Computer and communications security, 2010, pp. 753–755.

[13] Shang, W., Bannis, A., Liang, T., Wang, Z., Yu, Y., Afanasyev, A., and Zhang, L., “Named data networking of things,” In IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), 2016, pp. 117–128.

[14] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M., “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future generation computer systems, 29(7), 2013, pp. 1645–1660.

[15] Ghose, A., Biswas, P., Bhaumik, C., Sharma, M., Pal, A., and Jha, A., “Road condition monitoring and alert application: Using invehicle smartphone as internet-connected sensor,” In IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2012, pp. 489–491.

[16] Li, J., Sha, F., Zhang, Y., Huang, X., and Shen, J., “Verifiable outsourced decryption of attribute-based encryption with constant ciphertext length,” Security and Communication Networks, 2017.

[17] Y. Jiang, W. Susilo, Y. Mu, F. Guo, “Ciphertext-policy attributebased encryption against key-delegation abuse in fog computing,” Future Generation Computer Systems,78, 2017, pp. 720–729.

[18] Y. B. Saied, A. Olivereau, D. Zeghlache, M. Laurent, “Lightweight collaborative key establishment scheme for the Internet of Things,” Computer Networks, 64, 2014, pp. 273–295.

[19] M. J. Hinek, S. Jiang, R. Safavi-Naini, S. F. Shahandashti, “Attribute-based encryption with key cloning protection,” International Journal of Applied Cryptography, 2(3), 2012, pp. 250–270.

[20] S. Yu, K. Ren, W. Lou, J. Li, “Defending against key abuse attacks in KP-ABE enabled broadcast systems,” In International Conference on Security and Privacy in Communication Systems, 2009, pp. 311–329.

[21] J. Li, Q. Huang, X. Chen, S. S. Chow, D. S. Wong, D. Xie, “Multi-authority ciphertext-policy attribute-based encryption with accountability,” In Proceedings of the 6th ACM Symposium on Information, Computer and Communications Security, 2011, pp. 386–390.

[22] Z. Liu, Z. Cao, D. S. Wong, “Blackbox traceable CP-ABE: how to catch people leaking their keys by selling decryption devices on ebay,” In Proceedings of the ACM SIGSAC conference on Computer & communications security, 2013, pp. 475–486.

[23] J. Ning, Z. Cao, X. Dong, L. Wei, X. Lin, “Large universe ciphertext-policy attribute-based encryption with white-box traceability,” In European Symposium on Research in Computer Security, 2014, pp. 55–72.

[24] G. Yu, Z. Cao, G. Zeng, W. Han, “Accountable ciphertext-policy attribute-based encryption scheme supporting public verifiability and nonrepudiation,” In International Conference on Provable Security, 2016, pp. 3–18.

[25] J. Ning, Z. Cao, X. Dong, J. Gong, J. Chen, “Traceable CP-ABE with short ciphertexts: how to catch people selling decryption devices on eBay efficiently,” In European Symposium on Research in Computer Security, 2016, pp. 551–569.

JOURNAL OF CRITICAL REVIEWS

ISSN- 2394-5125 VOL 9, ISSUE 04, 2022

Effective Management with Flexibility to Access Authorized Data from Cloud based IoT

P Adi lakshmi,[1] E Maheswari[2], G Vineela[3], R Mounika[4], A Manogna[5]

[1]Asst. Professor, Department of Computer Science and engineering

[2,3,4,5]Student, Department of Computer Science and engineering

[1,2,3,4,5]QIS College of Engineering and Technology, Ongole-523001

Abstract-Data confidentiality and access control in the cloud may be achieved via attribute-based encryption. Attribute privacy is exposed when obvious characteristics are added to the ciphertext to help users locate accessible data in large datasets. The anonymous key-policy attribute-based encryption (AKP-ABE) is extended in this research to provide fine-grained data retrieval while maintaining attribute privacy to create an efficient attribute- based access control with authorised search scheme (EACAS). Data users may build search rules based on access policies and generate the trapdoor using the secret key supplied by data owners in EACAS, which allows for retrieval of data that is relevant to them. As a result, a virtual attribute with no semantic value is used in data encryption and trapdoor creation to enable the cloud to do an attribute-based search on the outsourced encrypted data. By setting the search rules, data users are able to search for interesting data based on protected characteristics while data owners may accomplish fine-grained access control on outsourced data. Finally, we show that EACAS is more efficient in terms of compute and storage overheads than currently available methods.

Keywords— Access control, authorized search, cloud storage, data sharing, key-policy attribute-basedencryption.

I. INTRODUCTION

Internet of Things (IoT) management services such as Amazon AWS IoT [1] or Google Cloud IoT Core [2] outsource the IoT data to cloud services. IoT devices in cloud-based management systems usually belong to distinct trust domains with complicated, asymmetrical trust relationships, such as the internet of things. Because of this, it is difficult to control access to outsourced IoT data from a single security domain. As a potential approach, attribute- based encryption (ABE) allows safe and fine-grained access control over encrypted data in accordance with the rules connected to it [3]. An encryptor may establish an access policy for the ciphertext using a collection of descriptive characteristics in ciphertext-policy ABE (CP-ABE) [4]. Using a decryption key, an attacker can only decipher the plaintext if the access privileges granted by his secret key match those in the ciphertext.

There are a number of concerns that need to be resolved before CP-ABE may be used in cloud-based IoT administration. To begin, the ciphertext develops in size in a manner that is proportional to the number of characteristics [5, 6, 7]. In IoT systems, this might be a major issue because of the many qualities required by IoT applications and services [8]. IoT systems can't be protected by CPABE schemes that only allow constant-size ciphertexts, even if they exist [9, 10, 11, 12].

A second advantage of CP-ABE is that it places high computing demands on a decryptor (rather than an encryptor), which may be battery-operated mobile devices like laptops. An untrusted cloud server may decode portion of ciphertexts on behalf of users, according to recent research in this area. Because of this, ciphertext volume could not be solved. It may be possible to combine current technologies such as outsourceable decryption [5] and constant-size cypher text ABE schemes for each function in order to deal with the aforementioned concerns A key blinding mechanism in the outsourced decryption algorithm [5] prevents it from solving the challenge. A user's private key may be obscured using a (secret) blinding factor, say z, such that the cloud can partially decode the data using the obscured key and return the plaintext obscured by z, specifically. As a result, z is all that is needed to uncover it. But when used with constant-size encryption text [11], this approach masks not just the plaintext but also additional parts that are required for decryption, but which the user does not know about. As a result, the user will never be able to accurately retrieve the plaintext. To accomplish both small ciphertext and outsourceable decryption functionality, Li et al. [16] devised a technique. Nevertheless, their technique severely restricts the ability to manage access, since only users whose characteristics match those in the access policy may decode a cypher text. As a last point, secret keys are vulnerable to being misused by unauthorised users who share their private keys. Secret keys may be unlawfully shared between authorised and unauthorised users in this situation. It would then be possible for anybody to access IoT data on the cloud. Traceable ABE was proposed as a solution to the leakage issue in previous investigations. Sadly, the majority of traceable ABE schemes simply look for the original owners of the keys. That example, if dishonest people exchange secret keys, the shared key holders still have access to encrypted cloud data. Due to realistic key recovery techniques such as side channel analysis, traceability alone is not enough to prevent key leakage. When it comes to the shared (or leaked) key issue, key revocation is not a viable solution, since the key holders will still be able access and retrieve data until it is revoked. In cloud-based IoT management systems, establishing a secure and efficient access control mechanism that addresses the aforementioned issues is critical. Using our new CPABE structure, we provide an effective and secure cloud-based IoT data management system in this study. The suggested system has the potential to track traitors who unlawfully distribute their secret keys, aswell as efficient storage and bandwidth management. A user-specific transformation key enables the cloud server to handle a considerable portion of the computational cost associated with decryption. In order to prevent unwanted access by a shared (or leaked) key holder, the cloud authenticates the identity of the key holder, partly decrypts the ciphertext using the key holder's transformation key, and finally provides the partially decrypted result. As a reminder, the transformation key is intimately linked to the original owner of the attribute key, who may decode the partly encrypted cypher text and retrieve plaintext only if he is the original owner. The plaintext can't be accessed byanybody else using the shared (or leaked) keys. Against summarise, the proposed system is impervious to forensically intractable key abuse assaults.. If the properties of the shared (or leaked) key are in line with the access policy associated to the encrypted IoT data, then this property holds. A shared (or leaked) key prevents users from accessing the IoT data. Only the original key holder can decrypt the message correctly. Our CP-ABE scheme, whichallows for outsourced decryption and key traceability as well as constant-size ciphertext, is the foundation for this resiliency method.

II. RELATEDWORKS

ABE schemes with outsourceable decryption [5] and constant-size ciphertext [11] may be combined to address the aforementioned concerns. A key blinding mechanism in the outsourced decryption algorithm [5]prevents it from solving the challenge. Users may blind their own secret key by applying a (secret) blinding factor known as z, which allows the cloud to do partial decryption using this blinded key, and then return plaintext masked by z. As a result, z is all that is needed to uncover it. Constant-size ciphertext, on the other hand, is disguised notonly by z but also by additional components that are required for decryption but unknown to the user when using this approach [11]. As a result, the user will never be able to accurately retrieve the plaintext. To accomplish both small ciphertext and outsourceable decryption functionality, Li et al. [16] devised a technique.

Nevertheless, their technique severely restricts the ability to manage access, since only users whose characteristics match those in the access policy may decode ciphertexts.Data from IoT sensors can be outsourced to the clouds via cloud servers and stored, exchanged, and processed via centralised or decentralised servers in the cloud, which makes these IoT systems vulnerable to both internal and external attacks. Cloud computing has contributed to the success of the Internet of Things by providing abundant storage and computation resources. Some cryptographic algorithms have been used to safeguard IoT data from possible harmful users and adversaries in order to maintain its confidentiality and integrity. Even if the data is encrypted, it is difficult to do any arithmetical operations. In theory, lattice-based fully-homomorphic encryption may offer a solution, but since it is computationally intensive, it cannot be deployed to the Internet of Things. When a semi-trusted server is involved, full-homomorphic encryption is possible. A distributed system for sharing IoT data, on the other hand, is difficult to implement. A fully-homomorphic encryption technique for cloud-based IoT applications has been developed to address this issue. A semi-trusted server may be used to assist in the calculation of homomorphic multiplications without getting any helpful information from the encrypted input.The term "e-Health" refers to the usage of the Internet and other linked networks in a healthcare system. It was our goal in this study to look at studies from 2017– 2020 to see how the integration of Internet of Things (IoT) devices and cloud computing has changed the way intelligent technologies are used in health care. Health information produced from electronic sources and gained knowledge, as characterised by the term "e-health," may be used to diagnose, treat, and prevent disease. The Internethas the ability to safeguard consumers from damage and allow them to fully engage in informed health-related decision-making as a storehouse for health information and e-Health analysis. High e-Health integration levels reduce the danger of encountering false material on the Internet. IoT-cloud-based eHealth systems are evaluatedfrom a variety of viewpoints, with a focus on the prospects, advantages, and obstacles of their deployment. Combining the Internet of Things and cloud computing with eHealth systems that are based on smart goals and applications is an exciting new trend. The Internet of Things has several privacy and security concerns (IoT). The

Internet of Things (IoT) has a number of issues, including a lack of effective and strong security mechanisms, user ignorance, and the well-known active device monitoring. By investigating the history of Internet of Things (IoT) systems and security measures, we are able to better understand: (a) different security and privacy issues; (b) approaches used to secure IoT-based environments and systems; (c) existing security solutions; and (d) the best privacy models necessary and suitable for different layers of IoT-driven application. IoT layered model: general and extended privacy and security components and layers identification were presented in this study. The suggested cloud/edge backed IoT system is put into action and assessed.

Amazon Web Service (AWS) Virtual Machines constitute the bottom layer of the IoT nodes. With the Greengrass Edge Environment on AWS, a Raspberry Pi 4 hardware kit was used to implement the intermediatelayer (edge). AWS's cloud-enabled IoT ecosystem was utilised to build the top layer of our solution (the cloud). Users' information was protected by the security protocols and crucial management sessions that were placed between each of these levels. In order to facilitate data movement across the levels of the proposed cloud/edge enabled IoT paradigm, we introduced security certificates. With the best security methodologies, the suggested system model may be utilised in conjunction with the proposed system model to countermeasure cybersecurity risks at each tier; cloud, edge, and IoT. As a result of Cloud Computing (CC) and Internet of Things (IoT) integration, treatment procedures have been radically altered in the ubiquitous computing era. As the volume of data created by IoT devices grows, so does the demand for a storage and processing infrastructure like the CC. Despite the fact that users and IoT devices continue to share computing and networking resources remotely, security challenges in CoT remain increasingly significant.

Furthermore, maintaining data privacy in such a setting is a top priority. As a result, the CoT is constantly expanding its focus on security and privacy. In this article, we examined some of the issues around data security and privacy and how they may be addressed. The CoT architecture and current applications have been examined in order to attain this goal. In addition, this study addresses a variety of security and privacy concerns and difficulties, as well as unresolved obstacles.

III. PROPOSED SYSTEM ARCHITECTURE

Using our new CPABE structure, we offer a cloud-based IoT data management system that is both efficient and safe. The suggested system has the potential to track traitors who unlawfully distribute their secret keys, as well as efficient storage and bandwidth management. A user-specific transformation key enables the cloud server to handle a considerable portion of the computational cost associated with decryption.

In order to prevent unwanted access by a shared (or leaked) key holder, the cloud authenticates the identity of the key holder, partly decrypts the ciphertext using the key holder's transformation key, and finally provides the partially decrypted result. You should be aware that the attribute key is intimately linked to the key holder, who can only decipher the plaintext if he is also the original owner of the attribute key. The plaintext can't be accessed by anybody else using the shared (or leaked) keys.To begin, the owner of the data must first sign up for an account on the cloud server and get authorization. Data owner will encrypt and upload file to cloud server after receiving consent from cloud data owner, and data owner will then request content key and master secret for file he uploaded and discovers Only when the keys have been produced can the file be sent to the cloud server for deduplication. After uploading a file, the data owner must grant the user access to search and download the file, as well as authorization to share the material. The cloud server is in charge of running and maintaining a cloud for the purpose of storing data. Data owners encrypt their data files and put them in the cloud for cloud End users to access and collaborate on. A content key and a master secret key are required for users to access shared data files. And the cloud will provide access to the data, as well as monitor all transactions and attacks associated to them.. The end user requests a content key anda secret key from Key Authority. The content key and master secret key produced with the associated data owner information of a certain file may be seen by KeyAuthority on all files. To see files stored in the cloud, the user must first create an account and log in.

The cloud has granted the user permission to validate the user's registration. To download the file, the user must request the MSK master secret key and the content key. As long as the data owner allows it, users may download and search their files.

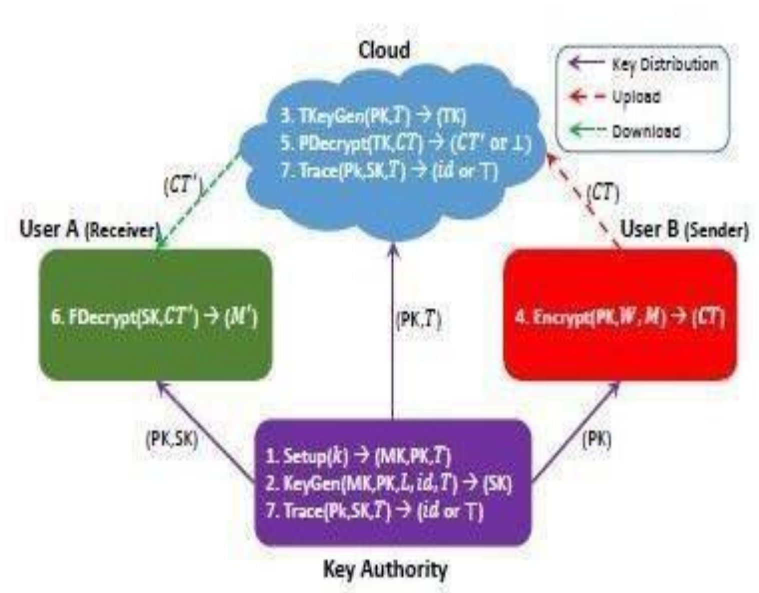
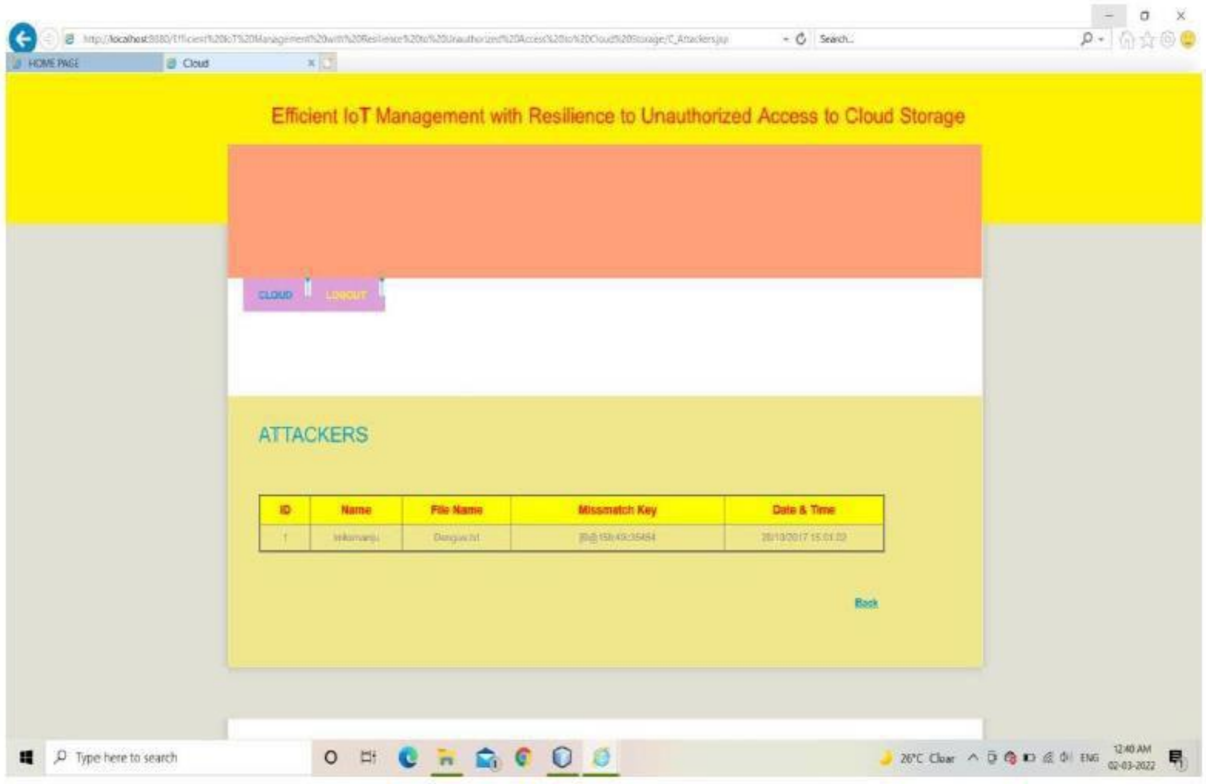

Cloud
Key Distribution
Upload
Download
3. TKeyGen(PK,T) → (TK)
5. PDecrypt(TK,CT) → (CT' or ⊥)
7. Trace(Pk,SK,T) → (id or T)
(CT')
(CT)
User A (Receiver)
User B (Sender)
6. FDecrypt(SK,CT') → (M')
(PK,T)
4. Encrypt(PK,W, M) → (CT)
(PK,SK)
(PK)
1. Setup(k) → (MK,PK,T)
2. KeyGen(MK,PK,L,id,T) → (SK)
7. Trace(Pk,SK,T) → (id or T)
Key Authority

Fig:1 System Architecture

IV. RESULTS AND DISCUSSION

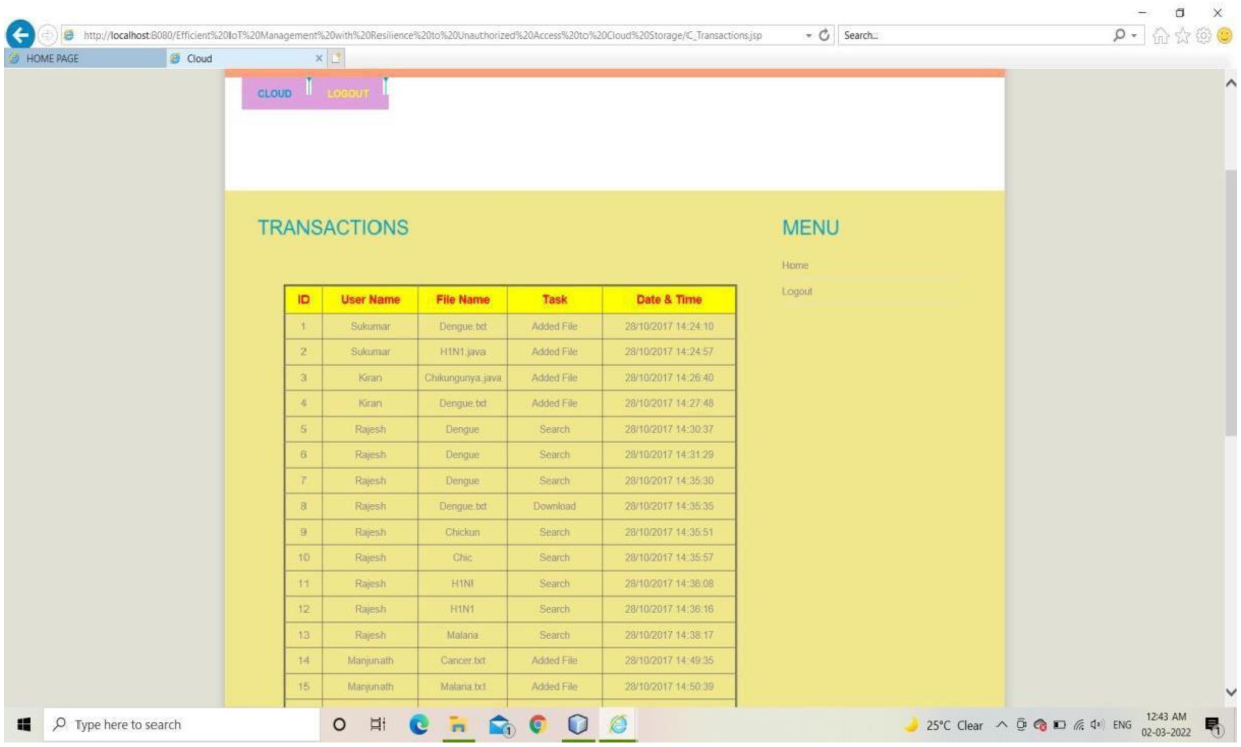
The results obtained are shown from Fig. 2 to Fig.5.


HOME PAGE
Cloud
Efficient IoT Management with Resilience to Unauthorized Access to Cloud Storage
CLOUD
LOGOUT
Back

ATTACKERS

ID	Name	File Name	Missmatch Key	Date & Time
1	[illegible]	[illegible]	[B@15b49:35464	20/10/2017 15:01:22

Fig.2 Attackers



TRANSACTIONS

ID	User Name	File Name	Task	Date & Time
1	Sukumar	Dengue.txt	Added File	28/10/2017 14:24:10
2	Sukumar	H1N1.java	Added File	28/10/2017 14:24:57
3	Kiran	Chikungunya.java	Added File	28/10/2017 14:26:40
4	Kiran	Dengue.txt	Added File	28/10/2017 14:27:48
5	Rajesh	Dengue	Search	28/10/2017 14:30:37
6	Rajesh	Dengue	Search	28/10/2017 14:31:29
7	Rajesh	Dengue	Search	28/10/2017 14:35:30
8	Rajesh	Dengue.txt	Download	28/10/2017 14:35:35
9	Rajesh	Chickun	Search	28/10/2017 14:35:51
10	Rajesh	Chic	Search	28/10/2017 14:35:57
11	Rajesh	H1NI	Search	28/10/2017 14:36:08
12	Rajesh	H1N1	Search	28/10/2017 14:36:16
13	Rajesh	Malaria	Search	28/10/2017 14:38:17
14	Manjunath	Cancer.txt	Added File	28/10/2017 14:49:35
15	Manjunath	Malaria.txt	Added File	28/10/2017 14:50:39

Fig.3 Transactions

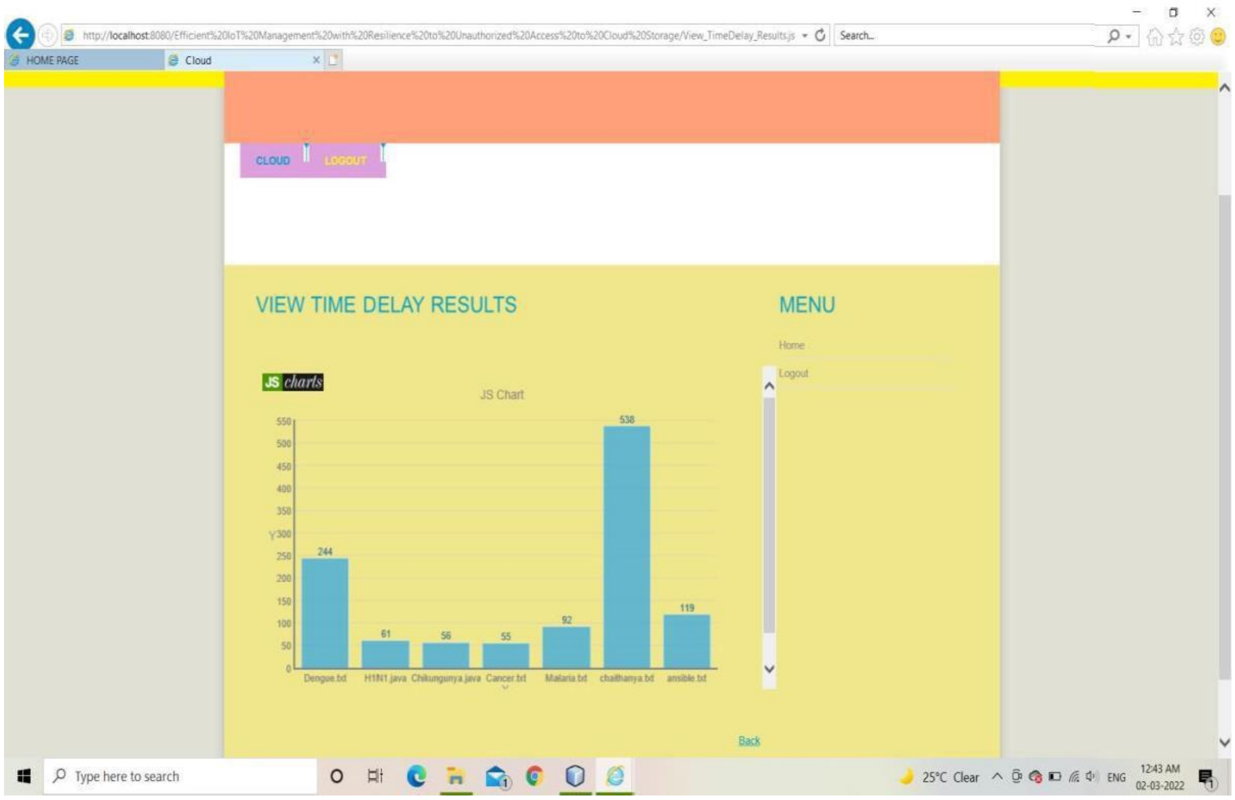


Fig.4 View Time Delay results

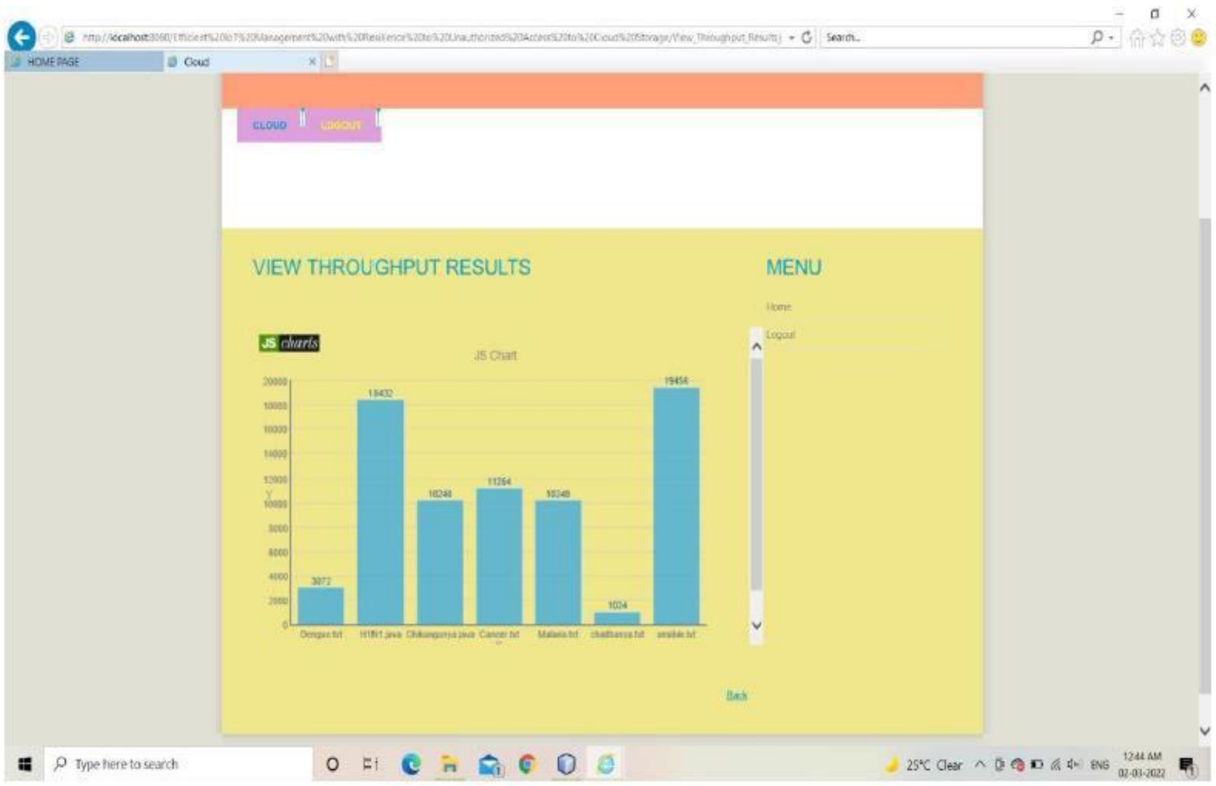


Fig.5 View Throughput results

V. FUTURE SCOPE AND CONCLUSION

Cloud-based IoT management systems may benefit from a revolutionary CP-ABE method. IoT devices create a consistent size ciphertext regardless of the amount of characteristics, making the proposed technique efficient in terms of transmission costs. Battery-powered user devices may transfer a considerable portion of decryption efforts to the cloud under the suggested system. Key traceability ensures that only the original owner of a given key may decode an encrypted file, preventing unlawfully shared key holders from accessing the material that has been outsourced. In IoT systems, forensically intractable key abuse assaults are common, and the suggested approach can withstand them.

REFERENCES

[1] "AWS IoT." AWS. https://aws.amazon.com/ko/iot. (accessed Dec. 17, 2019)
[2] "Cloud IoT Core." Google Cloud. https://cloud.google.com/iotcore. (accessedDec. 17, 2019)
[3] A. Sahai, B. Waters, "Fuzzy identity-based encryption," Advances in Cryptology-EUROCRYPT, 2005, pp. 457–473.
[4] J. Bethencourt, A. Sahai, B. Waters, "Ciphertext-policy attributebasedencryption," In IEEE symposium on security and privacy (SP'07), 2007, pp. 321–334.
[5] M. Green, S. Hohenberger, B. Waters, "Outsourcing the decryption of ABEciphertexts," In USENIX Security Symposium, Vol. 2011, No. 3, 2001.
[6] J. Lai, R. H. Deng, C. Guan, J. Weng, "Attribute-based encryption with verifiableoutsourced decryption," IEEE Transactions on Information Forensics and Security,8(8), 2013, pp. 1343–1354.
[7] S. Lin, R. Zhang, H. Ma, M. Wang, "Revisiting attribute-based encryption withverifiable outsourced decryption," IEEE Transactions on Information Forensics andSecurity, 10(10), 2015, pp. 2119–2130.

[8] S. Soursos, I. P. Zarko, P. Zwickl, I. Gojmerac, G. Bianchi, G. ˇ Carrozzo,“Towards the cross-domain interoperability of IoT platforms,” In EuropeanConference on Networks and Communications (EuCNC), 2016, pp. 398–402.
[9] C. Chen, Z. Zhang, D. Feng, “Efficient ciphertext policy attributebased encryptionwith constant-size ciphertext and constant computation-cost,” In Provable Security,2011, pp. 84–101.
[10] C. Hahn, H. Kwon, J. Hur, “Efficient attribute-based secure data sharing withhidden policies and traceability in mobile health networks,” Mobile InformationSystems, 2016.
[11] Z. Zhou, D. Huang, Z. Wang, “Efficient privacy-preserving ciphertext-policyattribute based-encryption and broadcast encryption,” IEEE Transactions onComputers, 64(1), 2015, pp. 126–138.
[12] Z. Zhou, D. Huang, “On efficient ciphertext-policy attribute based encryptionand broadcast encryption,” In Proceedings of the 17th ACM conference on Computerand communications security, 2010, pp. 753–755.
[13] Shang, W., Bannis, A., Liang, T., Wang, Z., Yu, Y., Afanasyev, A., and Zhang,L., “Named data networking of things,” In IEEE First International Conference onInternet-of-Things Design and Implementation (IoTDI), 2016, pp. 117–128.
[14] Gubbi, J., Buyya, R., Marusic, S., and Palaniswami, M., “Internet of Things(IoT): A vision, architectural elements, and future directions,” Future generationcomputer systems, 29(7), 2013, pp. 1645–1660.
[15] Ghose, A., Biswas, P., Bhaumik, C., Sharma, M., Pal, A., and Jha, A., “Roadcondition monitoring and alert application: Using invehicle smartphone as internet -connected sensor,” In IEEE International Conference on Pervasive Computing andCommunications Workshops (PERCOM Workshops), 2012, pp. 489–491.
[16] Li, J., Sha, F., Zhang, Y., Huang, X., and Shen, J., “Verifiable outsourceddecryption of attribute-basedencryption with constant ciphertext length,” Securityand Communication Networks, 2017.
[17] Y. Jiang, W. Susilo, Y. Mu, F. Guo, “Ciphertext-policy attributebased encryptionagainst key-delegation abuse in fog computing,” Future Generation ComputerSystems,78, 2017, pp. 720–729.
[18] Y. B. Saied, A. Olivereau, D. Zeghlache, M. Laurent, “Lightweight collaborativekey establishment scheme for the Internet of Things,” Computer Networks, 64, 2014,pp. 273–295.
[19] M. J. Hinek, S. Jiang, R. Safavi-Naini, S. F. Shahandashti, “Attribute-basedencryption with key cloning protection,” International Journal of AppliedCryptography, 2(3), 2012, pp. 250–270.
[20] S. Yu, K. Ren, W. Lou, J. Li, “Defending against key abuse attacks in KP –ABEenabled broadcast systems,” In International Conference on Security and Privacy inCommunication Systems, 2009, pp. 311–329.
[21] J. Li, Q. Huang, X. Chen, S. S. Chow, D. S. Wong, D. Xie, “Multi-authorityciphertext-policy attribute-based encryption with accountability,” In Proceedings ofthe 6th ACM Symposium on Information, Computer and Communications Security,2011, pp. 386–390.
[22] Z. Liu, Z. Cao, D. S. Wong, “Blackbox traceable CP-ABE: how to catch peopleleaking their keys by selling decryption devices on ebay,” In Proceedings of the ACMSIGSAC conference on Computer & communications security, 2013, pp. 475–486.
[23] J. Ning, Z. Cao, X. Dong, L. Wei, X. Lin, “Large universe ciphertext –policyattribute-based encryption with white-box traceability,” In European Symposium onResearch in Computer Security, 2014, pp. 55–72.
[24] G. Yu, Z. Cao, G. Zeng, W. Han, “Accountable ciphertext-policy attribute-basedencryption scheme supporting public verifiability and nonrepudiation,” InInternational Conference on Provable Security, 2016, pp. 3–18.
[25] J. Ning, Z. Cao, X. Dong, J. Gong, J. Chen, “Traceable CP -ABE with shortciphertexts: how to catch people selling decryption devices on eBay efficiently,” InEuropean Symposium on Research in Computer Security, 2016, pp. 551–569.