

MA-TEECM: MUTUAL ANONYMOUS AUTHENTICATION- BASED CREDENTIAL MIGRATION TECHNOLOGY FOR MOBILE TRUSTED EXECUTION ENVIRONMENTS

**A
Major Project Report**

Submitted to



Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

By

VENKATA SAI SWAPNA PALLAPOTHU - 20VE1A05H8

Under the Guidance

of

Mr.M.V.NAGESH

ASSISTANT PROFESSOR



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(Affiliated to JNTUH, Approved by A.I.C.T.E and Accredited by NAAC, New Delhi)

Bandlaguda, Beside InduAranya, Nagole,

Hyderabad-500068, Ranga Reddy Dist.

(2020-2024)



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that the Major Project Report on **“MA-TEECM: Mutual Anonymous Authentication-Based Credential Migration Technology for Mobile Trusted Execution Environments”** submitted by **VENKATA SAI SWAPNA PALLAPOTHU** bearing Hall ticket No. **20VE1A05H8** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **COMPUTER SCIENCE AND ENGINEERING** from Jawaharlal Nehru Technological University, Kukatpally, Hyderabad for the academic year 2023-2024 is a record of bonafide work carried out by her under our guidance and Supervision.

Project Coordinator

DR.U.M.FERNANDES DIMLO

Professor

Head of the Department

DR.U.M.FERNANDES DIMLO

Professor

Internal Guide

Mr.M.V.NAGESH

Assistant Professor

External Examiner



SREYAS INSTITUTE OF ENGINEERING AND TECHNOLOGY
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

I, **VENKATA SAI SWAPNA PALLAPOTHU** bearing Hallticket number: **20VE1A05H8** hereby declare that the Major Project titled **MA-TEECM: MUTUAL ANONYMOUS AUTHENTICATION-BASED CREDENTIAL MIGRATION TECHNOLOGY FOR MOBILE TRUSTED EXECUTION ENVIRONMENTS** done by me under the guidance of **Mr.M.V.NAGESH, ASSISTANT PROFESSOR** which is submitted in the partial fulfillment of the requirement for the award of the B.Tech degree in **Computer Science and Engineering** at **Sreyas Institute of Engineering and Technology** for Jawaharlal Nehru Technological University, Hyderabad is my original work.

VENKATA SAI SWAPNA PALLAPOTHU - 20VE1A05H8

ACKNOWLEDGEMENT

The successful completion of any task would be incomplete without mention of the people who made it possible through their guidance and encouragement crowns all the efforts with success.

I take this opportunity to acknowledge with thanks and deep sense of gratitude to **Mr.M.V.NAGESH, Assistant Professor, Department of Computer Science and Engineering** for his constant encouragement and valuable guidance during the Project work.

A Special vote of Thanks to **DR.U.M.FERNANDES DIMLO, Head of the Department** and **DR.U.M.FERNANDES DIMLO, Project Coordinator** who has been a source of Continuous motivation and support. He had taken time and effort to guide and correct me all through the span of this work.

I owe very much to the **Department Faculty, Principal** and the **Management** who made my term at **Sreyas Institute of Engineering and Technology** a stepping stone for my career. I treasure every moment I had spent in college.

Last but not the least, my heartiest gratitude to my parents and friends for their continuous encouragement and blessings. Without their support this work would not have been possible.

VENKATA SAI SWAPNA PALLAPOTHU - 20VE1A05H8

ABSTRACT

ARM TrustZone is the most widely used mobile trusted execution environment (TEE) technology today. Its hardware-enabled isolated execution environment provides reliable assurance of secure storage of credentials in mobile devices. However, the research on managing credentials stored in the TEE throughout the lifecycle of mobile devices has received little attention in recent years, and the credentials in TEE generally face usability problems caused by the mobile device lifecycle events. Aiming at the risk of information disclosure caused by the third-party service providers in the traditional credential migration scheme, this paper presents a mutual anonymous authentication-based credential migration framework for mobile trusted execution environments. First, we propose a peer-to-peer credential migration model between mobile terminals based on TrustZone and SGX, which solves the single point of failure caused by attacks on trusted third parties that act as credential transfer stations and managers in traditional solutions; Second, we propose an identity authentication protocol between TEEs based on mutual anonymous authentication, and a detailed authentication process is designed based on the universal mobile TEE model; Third, we build a formal verification model using High-Level Protocol Specification Language (HLPSL). Finally, the formal and informal security analysis indicate that the improved scheme meets the expected security requirements and is secure against several known attacks..

KEYWORDS: *Credential migration, trusted execution environments, mutual authentication.*

S.NO		TABLE OF CONTENTS	PAGE NO.
1		INTRODUCTION	1
	1.1	INTRODUCTION	1
	1.2	PROBLEM STATEMENT	2
	1.3	LITERATURE SURVEY	2
	1.3.1	ALGORITHM	4
	1.4	IMPLEMENTATION	4
2		SYSTEM FEASIBILITY	6
	2.1	INTRODUCTION	6
	2.2	EXISTING SYSTEM	6
	2.3	DISADVANTAGES OF EXISTING SYSTEM	7
	2.4	PROPOSED SYSTEM	7
	2.5	ADVANTAGES OF PROPOSED SYSTEM	7
3		SYSTEM ANALYSIS	9
	3.1	THE STUDY OF THE SYSTEM	9
	3.2	INPUT & OUTPUT REPRESENTATION	11
	3.3	INTRODUCTION TO JAVA	13
	3.4	SYSTEM REQUIREMENTS	29
	3.4.1	HARDWARE REQUIREMENTS	29
	3.4.2	SOFTWARE REQUIREMENTS	29
4		SYSTEM DESIGN	30
	4.1	SYSTEM ARCHITECTURE	30
	4.2	FLOW OF EVENTS	31
	4.3	UML DIAGRAMS	32
	4.3.1	CONSTRUCTION OF USECASE DIAGRAM	34
	4.3.2	SEQUENCE DIAGRAM	34
	4.3.3	CLASS DIAGRAM	35
	4.3.4	ACTIVITY DIAGRAM	36

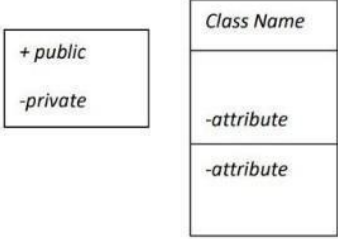

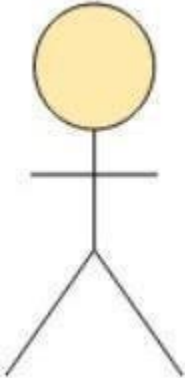
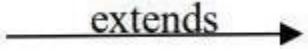
	4.3.5	STATE CHART DIAGRAM	37
	4.3.6	COLLABORATION DIAGRAM	38
	4.3.7	COMPONENT DIAGRAM	39
	4.3.8	DEPLOYMENT DIAGRAM	40
		TESTING AND VALIDATION	41
5	5.1	INTRODUCTION	41
	5.2	LEVELS OF TESTING	41
	5.3	TEST CASES	46
	5.4	EXECUTABLE CODE	48
6		SCREENSHOTS	68
	6.1	SCREENSHOTS OF MA-TEECM	68
7		RESULT	78
	7.1	RESULT OF MA-TEECM	78
8		CONCLUSION	79
9		FUTURE SCOPE	80
10		REFERENCES	81







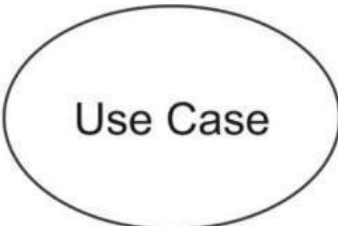
FIG. NO	LIST OF FIGURES	PAGE NO.
3.3.1	Working Process of Java	11
3.3.2	Java Virtual Machine	14
3.3.3	Default JRE	16
3.3.4	TCP/IP Stack	21
3.3.5	General J2ME Architecture	25
4.1.1	System Architecture	30
4.3.1.1	UseCase Diagram	34
4.3.2.1	Sequence Diagram	35
4.3.3.1	Class Diagram	36
4.3.4.1	Activity Diagram	36
4.3.5.1	State Chart Diagram	38
4.3.6.1	Collaboration Diagram	38
4.3.7.1	Component Diagram	39
4.3.8.1	Deployment Diagram	40
7.1.1	Output of MA-TEECM	70

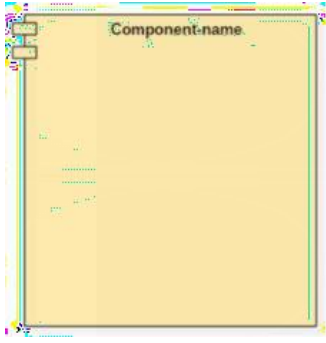
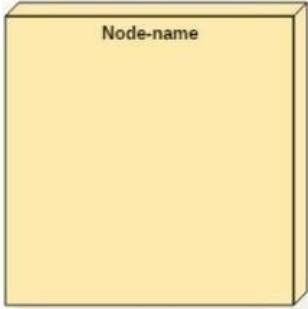
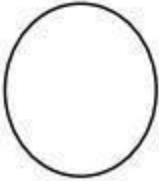


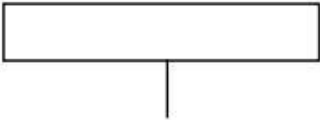
TABLE NO.	LIST OF TABLES	PAGE NO.
5.3.1	Test Case 1	46
5.3.2	Test Case 2	47
5.3.3	Test Case 3	47
5.3.4	Test Case 4	48


Screenshot. No	LIST OF SCREENSHOTS	Page. No
6.1.1	Home Page of MA-TEECM	68
6.1.2	Source Device Registration	68
6.1.3	Sending a skey by email	69
6.1.4	View the skey in sql	69
6.1.5	Login Group Manager	70
6.1.6	Activate Source Device request	70
6.1.7	Login Source Device using skey	71
6.1.8	Fill details and upload a file	71
6.1.9	View the uploaded file in sql	72
6.1.10	Registration of Target Device	72
6.1.11	Login Source Device using skey	73
6.1.12	View Target Device request	73
6.1.13	Login Target Device using skey	74
6.1.14	View the skey from the registrated mail in target device	74
6.1.15	Send request to Group Manager	75
6.1.16	Group Manager login	75
6.1.17	Activate Target Device Request	76
6.1.18	Delete data fromSource Device	76
6.1.19	Login Target Device	77
6.1.20	View data in Target Device	77

LIST OF SYMBOLS

S.NO	NAME OF SYMBOL	NOTATION	DESCRIPTION
1	CLASS		Represents a collection of similar entities grouped together.
2	ASSOCIATION		Associations represent static relationships between classes. Roles represent the way the two classes see each other.
3	ACTOR		It aggregates several classes into a single class.
4	RELATION (uses)	<i>Uses</i>	Used for additional process communication.
5	RELATION (extends)		Extends relationship is used when one use case is similar to another use case but does a bit more.

6	COMMUNICATION		Communication between various use cases.
7	STATE		State of the process
8	INITIAL STATE		Initial state of the object
9	FINAL STATE		Final state of the object
10	CONTROL FLOW		Represents various control flow between the states.
11	DECISION BOX		Represents decision making process from a constraint
12	USE CASE		Interact ion between the system and external environment.

13	COMPONENT		Represents physical modules which is a collection of components.
14	NODE		Represents physical modules which are a collection of components.
15	DATA PROCESS/ STATE		A circle in DFD represents a state or process which has been triggered due to some event or action.
16	EXTERNAL ENTITY		Represents external entities such as keyboard, sensors, etc
17	TRANSITION		Represents communication that occurs between processes.
18	OBJECT LIFELINE		Represents the vertical dimensions that the object communications.

19	MESSAGE		Represents the message exchanged.
----	---------	--	-----------------------------------

LIST OF ABBREVIATIONS

S.NO	ABBREVIATION	FULL FORM
1.	BYOD	Bring Your Own Device
2	TEE	Trusted Execution Environment
3.	TPM	Trusted Platform Module
4.	OCF	Open Certificate Platform
5.	SEE	Secure Execution Environment
6.	JVM	Java Virtual Machine
7.	API	Application Programming Interface
8.	GUI	Graphical User Interface
9.	TCP	Transmission Control Protocol
10.	UDP	User Datagram Protocol
11.	JSP	Java Server Pages
12.	JDBC	Java DataBase Connectivity
13.	ODBC	Open DataBase Connectivity
14.	J2ME	Java 2Micro Edition
15.	CLDC	Connected Limited Device Configuration
16.	CDC	Connected Device Configuration
17.	MIDP	Mobile Information Device Profile
18.	UML	Unified Modeling Language
19.	MAA	Mutual Anonymous Authentication
20.	SUT	Software Under Test

CHAPTER-1

INTRODUCTION

1.1 INTRODUCTION

Arm partners have shipped more than 232.4 billion Arm-based processor chips by mid-2022, which are widely used in mobile Internet devices such as mobile phones, tablet computers, and smartwatches. As mobile devices are more and more commonly used in business, finance, and information technology, the coexistence of sensitive data and normal data on mobile terminals is becoming very common. For example, Bring Your Own Device (BYOD) is a policy that allows employees to use their personal mobile devices to access office areas to process corporate data and login Intranet applications . Many enterprises accept it by creating secure containers on employees' personal mobile devices to ensure data security. However, because sensitive data, such as user credentials, are tightly coupled with mobile devices, when an user tries to migrate data to a new device due to a device's lifecycle events (such as terminal replacement or employee separation), the user usually needs to manually re-register credentials acquired in various scenarios to the new devices one by one, instead of migrating directly from the old terminal to the new. Credentials are the evidence that lets entities access privileged data and services, such as user keys, certificates, and other authentication information. As the device's usage time accumulates, a considerable amount of credentials will be stored in the trusted execution environment (TEE) of the mobile device, which poses several challenges to the credential management of the mobile terminal.

First, traditional user passwords are vulnerable to phishing and dictionary attacks, and key management software based on TEE is gradually gaining popularity to obtain more secure and convenient password management functions. For example, the Keystore system component has been introduced since Android 4.0, which makes the keys independent of the application or even the operating system. That is, the user can encrypt, decrypt and manage the key through the Keystore API without obtaining the key, which significantly improves the security of the keys. However, it also increases the cost for users to reconfigure keys. With the growth of the number of keys, it is no longer feasible to manually reconfigure keys on new terminals.

Second, with the rapid development and broad application of artificial intelligence technology, the machine learning process has been introduced in increasingly digital credentialing systems. For example, in all series of iPhone devices, the fingerprint and face print data stored in the TEE will be gradually strengthened over time, and if users cannot migrate this credential directly, it will take some time to relearn in the new terminal.

Finally, digital assets stored as credentials are gaining popularity, such as cryptocurrencies, NFT, and digital copyright certificates. Users urgently need a solution to automatically migrate their credential files to the new terminal when replacing devices. Therefore, it is necessary to migrate the credentials between devices considering device lifecycle events.

1.2 PROBLEM STATEMENT

Credentials are the evidence that lets entities access privileged data and services, such as user keys, certificates, and other authentication information. As the device's usage time accumulates, a considerable amount of credentials will be stored in the trusted execution environment of the device, which poses several challenges to the credential management of the mobile terminal. First, traditional user passwords are vulnerable to phishing and dictionary attacks, and key management software based on TEE is gradually gaining popularity to obtain more secure and convenient password management functions.

1.3 LITERATURE SURVEY

The TEE credential migration refers to transferring and reloading credential data between different TEEs. Credential migration services can save significant device re-initialization overhead and are critical for lifecycle events such as mobile device replacement. However, the standard TEE implementation today still cannot solve the problem of credential migration very well. The key migration issue first appeared in the research on the Trusted Platform Module (TPM), which is an essential part of TPM 1.2 and 2.0 specifications, and many researchers have proposed various methods to improve it [6]. However, research on key or credential migration for mobile TEE has not received sufficient attention. Based on a public resource known as the Open Certificate Platforms (OCP), Kari et al. proposed a trusted domain certificate migration protocol. They recommended encrypting and backing up the credentials on a trusted server with a password known only to the user and then completing the credential migration by entering the password again. The protocol framework does not require complex user interaction and authentication processes, however, all user credentials must be stored in the server in clear text, and the migration process becomes the process of reconfiguring the backup files in the server. Although a key known only by users protects the process, the architecture lacks a discussion on the identity authentication between the OCP and the two devices' TEE. There is a privacy breach due to the service provider's full access to user credentials and personal data. Arfaoui et al. propose a privacy-preserving scheme for migrating credentials between Global Platform TEEs, which requires dynamic interaction between

service providers and TEE managers. Although the authors mention that the service provider must authenticate the TEE, the migration protocol does not provide a specific identity certification procedure, and the necessity of mutual authentication between the service provider and the TEE is not covered.

Similarly, Literature and implement identity authentication management between credential migration devices through a trusted service provider. Carlton et al. demonstrated the necessity of mutual authentication in the credential migration service for the first time, and used formal tools to model their proposed mutual authentication protocol, proving the security of the protocol process. Tan and Song ,proposed a key migration protocol that supports mutual authentication between trusted roots, which achieves identity binding of both migration parties by adding device attributes in the authentication process between the source and target devices to the service provider. Nishimura et al. propose using a trusted third party to identify the owner of a personal device to prevent the sharing of authentication keys to malicious nodes. The literature mentioned above, however, all needs to assume that the third-party service provider is trusted.

- **TITLE :** Secure authentication key sharing between personal mobile devices based on owner identity

AUTHOR : H. Nishimura, Y. Omori, and T. Yamashita

ABSTRACT : The public key based Web authentication can be securely implemented using modern mobile devices with a hardware-assisted trusted environment such as the Trusted Execution Environment (TEE) as a secure storage of private keys. As a private key is strictly kept secret within the TEE and never leaves the device, there is a usability issue: the user must register the key separately on each device and Web site, which is burdensome for users who start using a new device. The aim of this research is to provide a solution with enhanced usability in key management by relaxing the restriction that the keys never leave the device and allowing the private keys to be shared among the devices while still maintaining an acceptable level of security. We introduce a third party that is responsible for supervising the key-sharing between devices in an authentication system. The third party performs the identification of the owner of each device to mitigate the risk of the keys being illegally shared to another person's device. Also, we propose a secure method for copying keys from the TEE of one device to that of another through a certificate-based mutually authenticated channel. We implemented the copying method in the ARM TrustZone-based TEE and showed that our approach is feasible on a commercially available smartphone.

● **TITLE :** Secure migration of WebAssembly-based mobile agents between secure enclaves

AUTHOR : V. A. B. Pop, S. Virtanen, P. Sainio,

ABSTRACT : Cryptography and security protocols are today commonly used to protect data at-rest and in-transit. In contrast, protecting data in-use has seen only limited adoption. Secure data transfer methods employed today rarely provide guarantees regarding the trustworthiness of the software and hardware at the communication endpoints. The field of study that addresses these issues is called Trusted or Confidential Computing and relies on the use of hardware-based techniques. These techniques aim to isolate critical data and its processing from the rest of the system. More specifically, it investigates the use of hardware isolated Secure Execution Environments (SEEs) where applications cannot be tampered with during operation. Over the past few decades, several implementations of SEEs have been introduced, each based on a different hardware architecture. However, lately, the trend is to move towards architecture-independent SEEs.

1.4 ALGORITHM

First, TAI verifies the availability of Agenti by checking the integrity of its process and issues a proxy signature certificate $\{\sigma, K\}$ for the verified Agenti, and then Agenti verifies the legitimacy of the certificate and uses it to secure subsequent communication. Silimar to literature [25], the specific process is as follows: Step 1: (x, V) . TAI randomly select the big primes p_1, q_1 , such that $q_1 | p_1 - 1$ and $g_1 \in \mathbb{Z}^*_{q_1}$ is a generator which order is q_1 . Generate the original signature private key $x \in \mathbb{R}\mathbb{Z}_{p_1-1}$, and the corresponding public key is $V = g^{x-1} \bmod p_1$. Where V, g_1 is disclosed to integrity-verified Agenti and potential signature verifiers. Step 2: (σ, K) . TAI generates random numbers $k \in \mathbb{R}\mathbb{Z}_{p_1-1}$, and calculates $K = g^{k-1} \bmod p_1$ and $\sigma = (x + kK) \bmod (p_1 - 1)$. Finally, send $\{\sigma, K\}$ to Agenti over a secure channel. Step 3: Proxy certificate verification. Agenti verify $g^{\sigma-1} \stackrel{?}{=} V K K \bmod p_1$. If holds, Agenti will become a legal proxy, otherwise, Agenti rejects the signature and terminates the protocol.

1.5 IMPLEMENTATION

MODULES:

Source device

when the source device's establishes a connection, the request process or even the key program itself may still use protocol vulnerabilities to transmit key request credentials to the receiver, causing the receiver to lose the ability to identify the connection to the sender.

Using this module source device will register with application and send request to group manager who will generate security key and send to source device which will be used to link source device information. Source device will authenticate with target device when migrating data after confirmation only data will be deleted from source device.

Target Device:

Identifying whether the target device belongs to the source device owner is critical in the credential migration 1) Ensure that the root of trust of the current device is secure, that is, satisfy the integrity, and grant it a ticket for end-to-end communication; 2) Verify that the terminal contains a root of trust before credential migration

Using this module target device will send authentication to source device to get key to login then send authentication request. To group manager to get other key with this key he can view data which is stored in source device.

Data Storage Server:

This module is used to store data from source device and target device and view information of every user.

Group Manager Server:

Specifically, a new group manager (GM) participant is introduced between the source device and the target device. GM is an enclave program responsible for verifying the integrity of the access devices, creating group signatures, and issuing group membership certificates for the source device and target device. With the assistance of the GM, a shared interaction channel is created for any legitimate devices.

Using this module group manager will login and view requests for key generation and key request from source and target devices and send keys through mail.

CHAPTER- 2

SYSTEM FEASIBILITY

2.1 INTRODUCTION

Identifying whether the target device belongs to the source device owner is critical in the credential migration. MA-TEECM is a wireless credential migration protocol designed for LANs where identification is replaced by the restrictive conditions in the process of LAN construction. Specifically, MA-TEECM splits the user identification task of the migrating terminal into the following two parts: 1) Ensure that the root of trust of the current device is secure, that is, satisfy the integrity, and grant it a ticket for end-to-end communication; 2) Verify that the terminal contains a root of trust before credential migration. Among them, the construction of the migration group is used to realize the first task, and mutual anonymous authentication is used to the second task. Furthermore, in practical applications, the user usually needs to shut down the device and reboot into engineering mode to initiate credential migration. Therefore, it is also possible to ensure the consistency of user identities through a specified operation procedure or further using a migration password known only by the current user.. Anonymity means that the identity information of the device will not be revealed during the authentication process. The TCG specification requires the root of trust to generate different session keys PK based on the public key of the device endorsement key EK, to ensure that the verifier cannot associate a specific root of trust with the session key. To determine whether the session key was generated from the same TA, the attacker needs to determine that T_1 , T_2 and $T'1$, $T'2$ were generated from the same E. According to the DDH assumption, this is impractical. According to the property of direct anonymous authentication, the challenger can only confirm that the verifier is from a valid trusted root, but cannot identify its real identity. Therefore, the MA-TEECM scheme satisfies anonymity

2.2 EXISTING SYSTEM

Based on a public resource known as the Open Certificate Platforms (OCP), Kari et al. proposed a trusted domain certificate migration protocol. They recommended encrypting and backing up the credentials on a trusted server with a password known only to the user and then completing the credential migration by entering the password again. The protocol framework does not require complex user interaction and authentication processes; however, all user credentials must be stored in the server in clear text, and the migration process becomes the process of reconfiguring the backup files in the server.

2.3 DISADVANTAGES OF EXISTING SYSTEM

- Although a key known only by users protects the process, the architecture lacks a discussion on the identity authentication between the OCP and the two devices' TEE.
- There is a privacy breach due to the service provider's full access to user credentials and personal data.

2.4 PROPOSED SYSTEM

Considering the target model, the attacker model, and the Global Platform TEE specification, this paper proposes a novel model MA-TEECM, for TEE credential migration based on mutual anonymous authentication. Specifically, a new group manager (GM) participant is introduced between the source TEE and the target TEE. GM is an enclave program running in Intel SGX, responsible for verifying the integrity of the access device's TEE, creating group signatures, and issuing group membership certificates for the source TEE and target TEE. With the assistance of the GM, a shared interaction channel is created for any legitimate TEE.

2.5 ADVANTAGES OF PROPOSED SYSTEM

We recommend that users implement credential migration between user devices in a peer-to-peer manner to prevent remote attackers from compromising key infrastructure. GM verifies the integrity of the TEE fingerprint of mobile devices and issues group member certificates to all nodes that pass the verification. We provide an online algorithm, named CEDC-O, based on Lyapunov optimization, to convert the long-term optimization problem. MA-TEECM enables mutual anonymous authentication, ensuring that users' identities remain private and secure during credential migration. Secure Credential Migration facilitates secure migration of credentials between mobile trusted execution environments, protecting sensitive information from unauthorized access. MA-TEECM ensures that trust is maintained throughout the credential migration process, guaranteeing the integrity of the transaction. By utilizing trusted execution environments and anonymous authentication, MA-TEECM provides an additional layer of security for mobile transactions. The technology enables efficient credential migration, reducing the overhead and latency associated with traditional authentication methods.

MA-TEECM enables seamless communication and credential migration between different mobile trusted execution environments, promoting interoperability. The technology allows users to

securely and privately migrate credentials between devices, enhancing the overall user experience. By minimizing the exposure of sensitive information, MA-TEECM reduces the risk of identity theft, fraud, and other security threats. MA-TEECM can help organizations comply with regulations and standards requiring robust security and privacy measures for mobile transactions. The technology is designed to support a large number of users and transactions, making it an ideal solution for large-scale mobile applications.

CHAPTER- 3

SYSTEM ANALYSIS

3.1 THE STUDY OF THE SYSTEM

- To conduct studies and analyses of an operational and technological nature, and
- To promote the exchange and development of methods and tools for operational analysis as applied to defense problems.

Logical design

The logical design of a system pertains to an abstract representation of the data flows, inputs and outputs of the system. This is often conducted via modeling, using an over-abstract (and sometimes graphical) model of the actual system. In the context of systems design are included. Logical design includes ER Diagrams i.e. Entity Relationship Diagrams

Physical design

The physical design relates to the actual input and output processes of the system. This is laid down in terms of how data is input into a system, how it is verified / authenticated, how it is processed, and how it is displayed as output. In Physical design, following requirements about the system are decided.

1. Input requirement,
2. Output requirements,
3. Storage requirements,
4. Processing Requirements,
5. System control and backup or recovery.

Put another way, the physical portion of systems design can generally be broken down into three sub-tasks:

1. User Interface Design
2. Data Design
3. Process Design

User Interface Design is concerned with how users add information to the system and with

how the system presents information back to them. Data Design is concerned with how the data is represented and stored within the system. Finally, Process Design is concerned with how data moves through the system, and with how and where it is validated, secured and/or transformed as it flows into, through and out of the system. At the end of the systems design phase, documentation describing the three sub-tasks is produced and made available for use in the next phase.

Physical design, in this context, does not refer to the tangible physical design of an information system. To use an analogy, a personal computer's physical design involves input via a keyboard, processing within the CPU, and output via a monitor, printer, etc. It would not concern the actual layout of the tangible hardware, which for a PC would be a monitor, CPU, motherboard, hard drive, modems, video/graphics cards, USB slots, etc. It involves a detailed design of a user and a product database structure processor and a control processor. The H/S personal specification is developed for the proposed system.

Functional Requirements

Mutual Anonymous Authentication enable secure and private authentication between devices and trusted execution environments. Credential Migration facilitate secure migration of credentials between mobile trusted execution environments. It ensure trust is maintained throughout the credential migration process. Support seamless communication and credential migration between different mobile trusted execution environments.

Non-Functional Requirements

Protect sensitive information from unauthorized access and ensure the integrity of credentials. Maintain user anonymity and confidentiality during authentication and credential migration. Minimize overhead and latency during credential migration. Scalability support a large number of users and transactions. Provide an intuitive and user-friendly experience for credential migration.

System Components

1. Mutual Anonymous Authentication Protocol: Enables secure and private authentication between devices and TEEs.
2. Credential Migration Manager: Facilitates secure credential migration between TEEs.
3. Trust Manager: Ensures trust is maintained throughout the credential migration process.

4. Interoperability Manager: Enables seamless communication and credential migration between different TEEs.

System Flow

User initiates credential migration request. MA-TEECM Server authenticates user device and TEE using mutual anonymous authentication protocol. Credential Migration Manager migrates credentials between TEEs. Trust Manager ensures trust is maintained throughout the process. Interoperability Manager enables seamless communication and credential migration. Migrated credentials are stored in the Credential Repository.

System Analysis

The MA-TEECM system is designed to facilitate secure and private credential migration between mobile trusted execution environments. The system requires mutual anonymous authentication between devices and trusted execution environments, ensuring that user identities remain private and secure. The credential migration process is facilitated by the Credential Migration Manager, which ensures that credentials are securely transferred between trusted execution environments. The Trust Manager ensures that trust is maintained throughout the credential migration process, guaranteeing the integrity of the transaction. The Interoperability Manager enables seamless communication and credential migration between different mobile trusted execution environments, promoting interoperability.

The system provides enhanced security and privacy, efficient and scalable credential migration, trust preservation throughout the process, and interoperability between different trusted execution environments. However, the system also faces challenges such as ensuring security and privacy in a decentralized environment, maintaining trust and interoperability between different trusted execution environments, scalability and efficiency in a large-scale deployment, user adoption and awareness of the technology, and integration with existing systems and infrastructure.

3.2 INPUT & OUTPUT REPRESENTATION

Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required,

controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

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

Objectives

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

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

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

Output Design

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

- a. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.
- b. Select methods for presenting information.
- c. Create document, report, or other formats that contain information produced by the system.

3.3 INTRODUCTION TO JAVA

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.

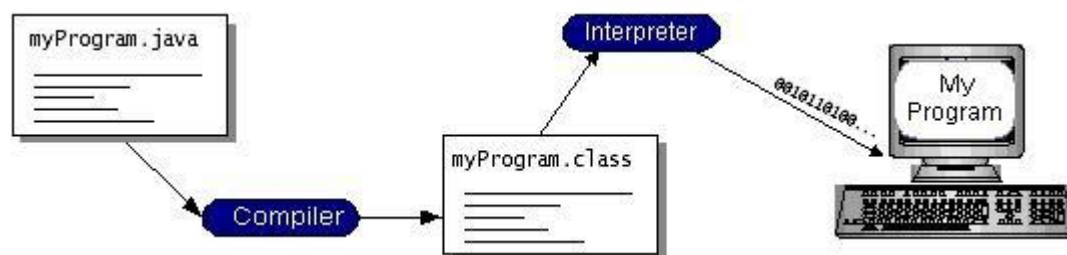


Figure 3.3.1: Working process of Java

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.

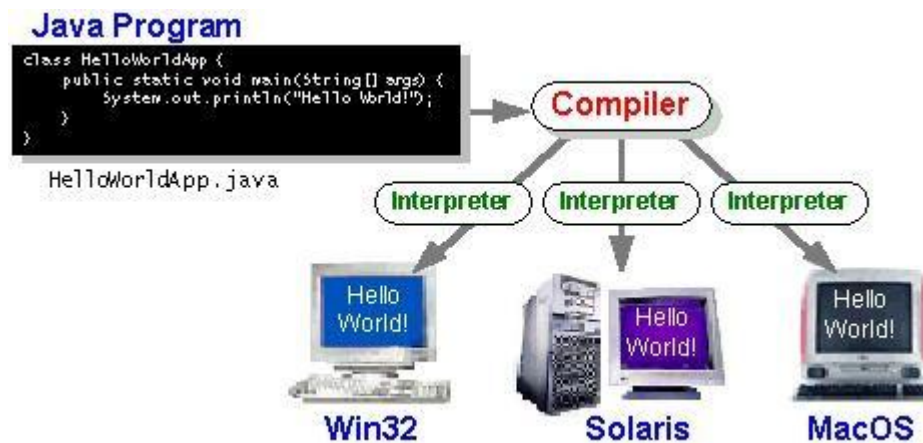


Figure 3.3.2: Java Virtual Machine

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.

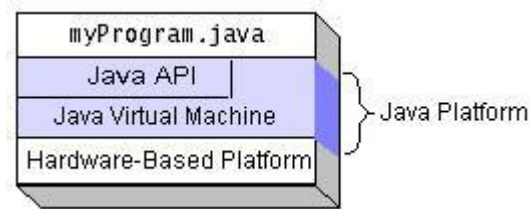
The Java platform has two components:

- 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. The Java platform is widely used in various domains, including:

1. Web Development: Java is used for building enterprise-level web applications, servlets, and JavaServer Pages (JSP).
2. Android App Development: Java is used for developing Android apps, as it's the primary language for Android development.
3. Desktop Applications: Java is used for building desktop applications, such as IDEs, media players, and games.
4. Enterprise Software: Java is used in enterprise software development for building complex systems, such as banking and financial applications.
5. Machine Learning and Data Science: Java is used in machine learning and data science for building predictive models, data analysis, and visualization.

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 Datagram 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™, can plug into existing component architectures.
- **Object serialization:** Allows lightweight persistence and communication via Remote Method Invocation (RMI).
- **Java Database Connectivity (JDBC™):** 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.

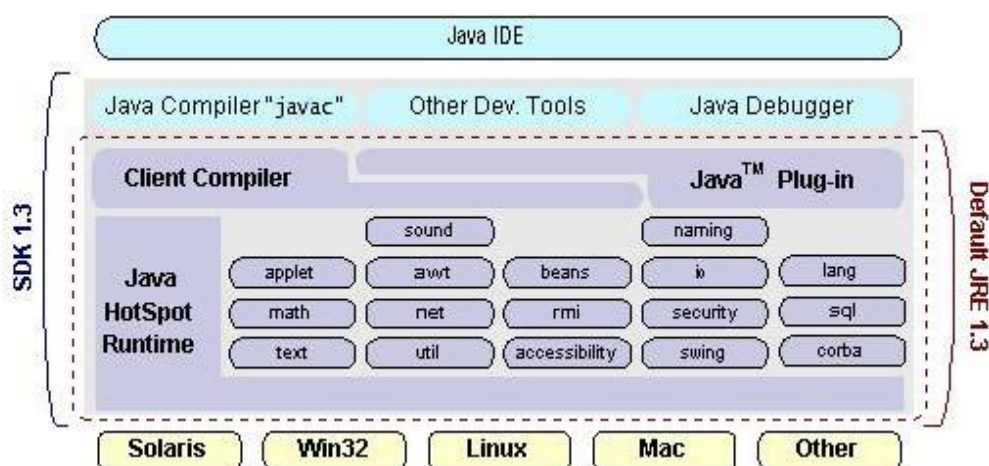


Figure 3.3.3: Default JRE

How Will Java Technology Change My Life?

We can't promise you fame, fortune, or even a job if you learn the Java programming language. Still, it is likely to make your programs better and requires less effort than other languages. We believe that Java technology will help you do the following:

- **Get started quickly:** Although the Java programming language is a powerful object-oriented language, it's easy to learn, especially for programmers already familiar with C or C++.
- **Write less code:** Comparisons of program metrics (class counts, method counts, and so on) suggest that a program written in the Java programming language can be four times smaller than the same program in C++.
- **Write better code:** The Java programming language encourages good coding practices, and its garbage collection helps you avoid memory leaks. Its object orientation, its JavaBeans component architecture, and its wide-ranging, easily extendible API let you reuse other people's tested code and introduce fewer bugs.
- **Develop programs more quickly:** Your development time may be as much as twice as fast versus writing the same program in C++. Why? You write fewer lines of code and it is a simpler programming language than C++.
- **Avoid platform dependencies with 100% Pure Java:** You can keep your program portable by avoiding the use of libraries written in other languages. The 100% Pure Java™ Product Certification Program has a repository of historical process manuals, white papers, brochures, and similar materials online.
- **Write once, run anywhere:** Because 100% Pure Java programs are compiled into machine-independent byte codes, they run consistently on any Java platform.
- **Distribute software more easily:** You can upgrade applets easily from a central server. Applets take advantage of the feature of allowing new classes to be loaded "on the fly," without recompiling the entire program.

ODBC

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

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an

ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

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

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

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

JDBC

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

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

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

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

Key Features of JDBC:

1. **Database Independence:** JDBC allows developers to write database-independent code, making it possible to switch between different databases without modifying the application.
2. **SQL Support:** JDBC supports SQL statements, including SELECT, INSERT, UPDATE, and DELETE, as well as stored procedures and functions.
3. **Connection Management:** JDBC manages connections to the database, including connection pooling and transaction management.
4. **Result Set Handling:** JDBC provides a ResultSet interface for retrieving and manipulating data from the database.
5. **Error Handling:** JDBC provides a robust error handling mechanism, allowing developers to handle database errors and exceptions.

JDBC Architecture:

1. **JDBC Driver:** A software component that implements the JDBC API and communicates with the database.
2. **JDBC API:** A set of interfaces and classes that provide a standard interface for interacting with the database.
3. **Database:** The underlying database management system, such as MySQL, Oracle, or PostgreSQL.

JDBC is widely used in Java applications for various purposes, including:

1. **Database Connectivity:** Connecting to a database and executing SQL statements.
2. **Data Retrieval:** Retrieving data from the database and manipulating it in the application.
3. **Data Insertion:** Inserting data into the database from the application.
4. **Data Update:** Updating existing data in the database from the application.
5. **Database Administration:** Performing database administration tasks, such as creating tables and indexes

JDBC Goals

Few software packages are designed without goals in mind. JDBC is one that, because of its many goals, drove the development of the API. These goals, in conjunction with early reviewer feedback, have finalized the JDBC class library into a solid framework for building database applications in Java.

The goals that were set for JDBC are important. They will give you some insight as to why certain classes and functionalities behave the way they do. The eight design goals for JDBC are as follows:

1. SQL Level API

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to “generate” JDBC code and to hide many of JDBC’s complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. JDBC must be implemental on top of common database interfaces

The JDBC SQL API must “sit” on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and vice versa.

4. Provide a Java interface that is consistent with the rest of the Java system

Because of Java’s acceptance in the user community thus far, the designers feel that they should not stray from the current design of the core Java system.

5. Keep it simple

This goal probably appears in all software design goal listings. JDBC is no exception. Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the API.

6. Use strong, static typing wherever possible

Strong typing allows for more error checking to be done at compile time; also, less error appear at runtime.

7. **Security**

To provide a secure way to access the database, including support for user authentication and encryption.

8. **Performance**

To optimize performance when interacting with the database, including support for batching and caching.

9. **Scalability**

To support large-scale applications and databases, including support for distributed transactions and multi-threading.

10. **Portability**

To enable developers to write portable code that can run on various platforms and databases.

11. **Simplification**

To simplify the process of interacting with databases, making it easier for developers to access and manipulate data.

Networking

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

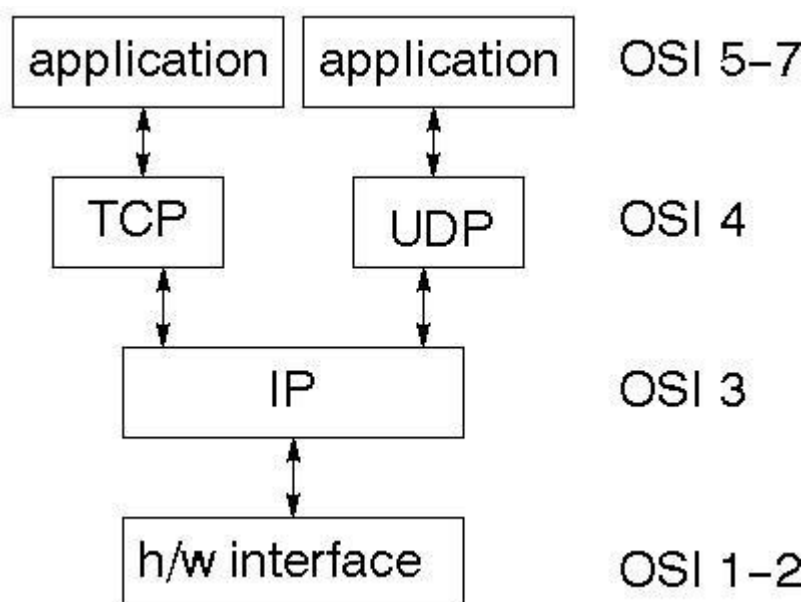


Figure 3.3.4: TCP/IP stack

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

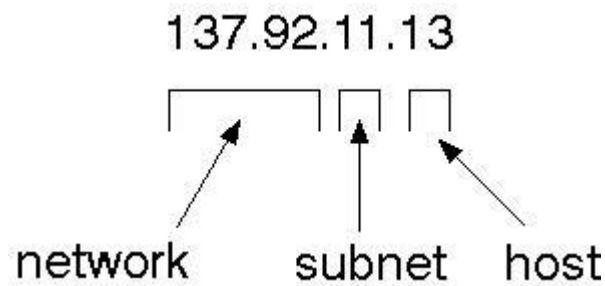
Internally, the UNIX network is divided into sub networks. Building 11 is currently on one sub network and uses 10-bit addressing, allowing 1024 different hosts.

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 total address in an IP datagram refers to the combination of the source IP address and the destination IP address. The total address is the combination of these two addresses, which uniquely identifies the connection between the sender and the recipient. It is used to route the packet through the internet and ensure that it reaches the intended destination.



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.

JFree Chart

JFreeChart is a free 100% Java chart library that makes it easy for developers to display professional quality charts in their applications. JFreeChart's extensive feature set includes:

A consistent and well-documented API, supporting a wide range of chart types;

A flexible design that is easy to extend, and targets both server-side and client-side applications;

Support for many output types, including Swing components, image files (including PNG and JPEG), and vector graphics file formats (including PDF, EPS and SVG);

1. Map Visualizations

Charts showing values that relate to geographical areas. Some examples include: (a) population density in each state of the United States, (b) income per capita for each country in Europe, (c) life expectancy in each country of the world. The tasks in this project include:

Sourcing freely redistributable vector outlines for the countries of the world, states/provinces in particular countries (USA in particular, but also other areas);

Creating an appropriate dataset interface (plus default implementation), a rendered, and integrating this with the existing XYPlot class in JFreeChart;

Testing, documenting, testing some more, documenting some more.

2. Time Series Chart Interactivity

Implement a new (to JFreeChart) feature for interactive time series charts --- to display a separate control that shows a small version of ALL the time series data, with a sliding "view" rectangle that allows you to select the subset of the time series data to display in the main chart.

3. Dashboards

There is currently a lot of interest in dashboard displays. Create a flexible dashboard mechanism that supports a subset of JFreeChart chart types (dials, pies, thermometers, bars, and lines/time series) that can be delivered easily via both Java Web Start and an applet.

4. Property Editors

The property editor mechanism in JFreeChart only handles a small subset of the properties that can be set for charts. Extend (or reimplement) this mechanism to provide greater end-user control over the appearance of the charts.

J2ME (Java 2 Micro edition):-

Sun Microsystems defines J2ME as "a highly optimized Java run-time environment targeting a wide range of consumer products, including pagers, cellular phones, screen-phones, digital set-top boxes and car navigation systems." Announced in June 1999 at the JavaOne Developer Conference, J2ME brings the cross-platform functionality of the Java language to smaller devices, allowing mobile wireless devices to share applications. With J2ME, Sun has adapted the Java platform for consumer products that incorporate or are based on small computing devices. J2ME (Java 2 Micro Edition) is a subset of the Java platform designed for mobile devices and embedded systems. It was introduced by Sun Microsystems (now owned by Oracle Corporation) to enable Java applications on resource-constrained devices like mobile phones, PDAs, and set-top boxes. J2ME is optimized for devices with limited memory and processing power. J2ME provides a subset of Java libraries, reducing the overall size of the runtime environment. It defines different configurations (e.g., CLDC, CDC) and profiles (e.g., MIDP, PBP) to cater to various device categories and use cases. It uses a customized JVM, called the KVM (K Virtual Machine), which is optimized for low-power devices. J2ME provides a sandboxed environment, restricting applications from accessing sensitive device resources without permission. It supports networking capabilities, enabling applications to communicate over the internet or with other devices. J2ME was widely used in the early 2000s for mobile applications, especially games, but has largely been replaced by newer technologies like Android and iOS. However, it still finds use in some niche areas, such as embedded systems and legacy devices.

1. General J2ME architecture

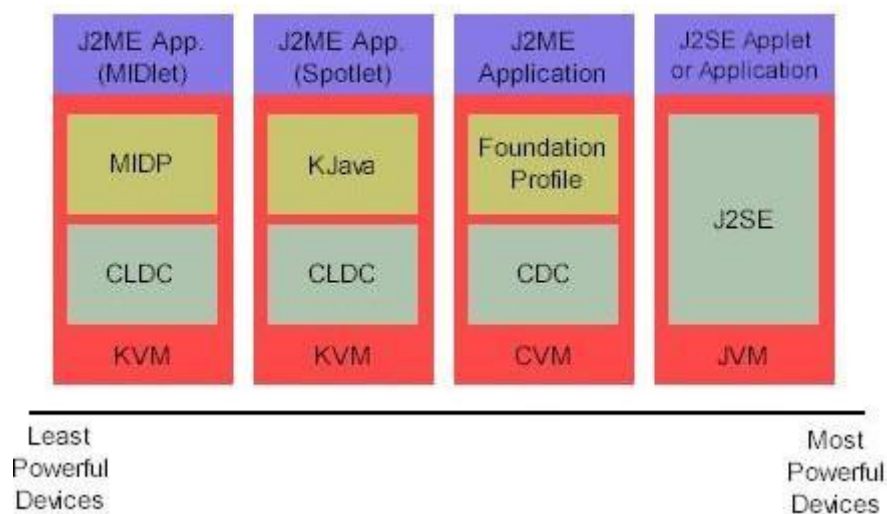


Figure 3.3.5: General J2ME architecture

J2ME uses configurations and profiles to customize the Java Runtime Environment (JRE). As a complete JRE, J2ME is comprised of a configuration, which determines the JVM used, and a profile, which defines the application by adding domain-specific classes. The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. We'll discuss configurations in detail in the The profile defines the application; specifically, it adds domain-specific classes to the J2ME configuration to define certain uses for devices. We'll cover profiles in depth in the The following graphic depicts the relationship between the different virtual machines, configurations, and profiles. It also draws a parallel with the J2SE API and its Java virtual machine. While the J2SE virtual machine is generally referred to as a JVM, the J2ME virtual machines, KVM and CVM, are subsets of JVM. Both KVM and CVM can be thought of as a kind of Java virtual machine -- it's just that they are shrunken versions of the J2SE JVM and are specific to J2ME.

2. Developing J2ME applications

Introduction In this section, we will go over some considerations you need to keep in mind when developing applications for smaller devices. We'll take a look at the way the compiler is invoked when using J2SE to compile J2ME applications. Finally, we'll explore packaging and deployment and the role preverification plays in this process.

3. Design considerations for small devices

Developing applications for small devices requires you to keep certain strategies in mind during the design phase. It is best to strategically design an application for a small device before you begin coding. Correcting the code because you failed to consider all of the "gotchas" before developing the application can be a painful process. Here are some design strategies to consider:

- * Keep it simple. Remove unnecessary features, possibly making those features a separate, secondary application.
- * Smaller is better. This consideration should be a "no brainer" for all developers. Smaller applications use less memory on the device and require shorter installation times. Consider packaging your Java applications as compressed Java Archive (jar) files.
- * Minimize run-time memory use. To minimize the amount of memory used at run time, use scalar types in place of object types. Also, do not depend on the garbage collector. You should manage the memory efficiently yourself by setting object references to null when you are finished with them. Another way to reduce run-time memory is to use lazy instantiation, only allocating objects on an as-needed basis. Other ways of reducing overall and peak memory use on small devices are to release resources quickly, reuse objects, and avoid exceptions.

4. Configurations overview

The configuration defines the basic run-time environment as a set of core classes and a specific JVM that run on specific types of devices. Currently, two configurations exist for J2ME, though others may be defined in the future:

* **Connected Limited Device Configuration (CLDC)** is used specifically with the KVM for 16-bit or 32-bit devices with limited amounts of memory. This is the configuration (and the virtual machine) used for developing small J2ME applications. Its size limitations make CLDC more interesting and challenging (from a development point of view) than CDC. CLDC is also the configuration that we will use for developing our drawing tool application. An example of a small wireless device running small applications is a Palm hand-held computer.

* **Connected Device Configuration (CDC)** is used with the C virtual machine (CVM) and is used for 32-bit architectures requiring more than 2 MB of memory. An example of such a device is a Net TV box.

5. J2ME profiles

What is a J2ME profile?

As we mentioned earlier in this tutorial, a profile defines the type of device supported. The Mobile Information Device Profile (MIDP), for example, defines classes for cellular phones. It adds domain-specific classes to the J2ME configuration to define uses for similar devices. Two profiles have been defined for J2ME and are built upon CLDC: KJava and MIDP. Both KJava and MIDP are associated with CLDC and smaller devices. Profiles are built on top of configurations. Because profiles are specific to the size of the device (amount of memory) on which an application runs, certain profiles are associated with certain configurations. A skeleton profile upon which you can create your own profile, the Foundation Profile, is available for CDC.

J2ME profiles are a crucial aspect of the Java 2 Micro Edition platform, which enables developers to create applications for resource-constrained devices like mobile phones, PDAs, and set-top boxes. These profiles define a subset of the Java platform's features and libraries, optimizing them for specific device categories and use cases. By using J2ME profiles, developers can create applications that are tailored to the unique characteristics of different devices, ensuring optimal performance, functionality, and user experience.

The Mobile Information Device Profile (MIDP) is one of the most widely used J2ME profiles, designed specifically for mobile phones and other resource-constrained devices. MIDP provides a compact and efficient Java runtime environment, including APIs for user interface, networking, and persistence. This profile is ideal for developing mobile applications that require

minimal resources and maximum efficiency.

Profile 1: KJava

KJava is Sun's proprietary profile and contains the KJava API. The KJava profile is built on top of the CLDC configuration. The KJava virtual machine, KVM, accepts the same byte codes and class file format as the classic J2SE virtual machine. KJava contains a Sun-specific API that runs on the Palm OS. The KJava API has a great deal in common with the J2SE Abstract Windowing Toolkit (AWT). However, because it is not a standard J2ME package, its main package is `com.sun.kjava`. We'll learn more about the KJava API later in this tutorial when we develop some sample applications.

Profile 2: MIDP

MIDP is geared toward mobile devices such as cellular phones and pagers. The MIDP, like KJava, is built upon CLDC and provides a standard run-time environment that allows new applications and services to be deployed dynamically on end user devices. MIDP is a common, industry-standard profile for mobile devices that is not dependent on a specific vendor. It is a complete and supported foundation for mobile application development. MIDP contains the following packages, the first three of which are core CLDC packages, plus three MIDP-specific packages.

- * `java.lang`
- * `java.io`
- * `java.util`
- * `javax.microedition.io`
- * `javax.microedition.lcdui`
- * `javax.microedition.midlet`
- * `javax.microedition.rms`

3.4 SYSTEM REQUIREMENTS

3.4.1 HARDWARE REQUIREMENTS

- System : Pentium IV 2.4 GHz.
- Hard Disk : 200 GB.
- Floppy Drive : 1.44 Mb.
- Monitor : 15 VGA Colour.
- Mouse : Logitech.
- Ram : 1 GB.

3.4.2 SOFTWARE REQUIREMENTS

- Operating system : Windows XP/7/10.
- Coding Language : Java
- Tool : Netbeans
- Database : MYSQL

CHAPTER - 4

SYSTEM DESIGN

4.1 SYSTEM ARCHITECTURE

Architecture Flow:

Below architecture diagram represents mainly flow of request from the users to database through servers. In this scenario overall system is designed in three tiers separately using three layers called presentation layer, business layer, data link layer. This project was developed using 3-tier architecture.

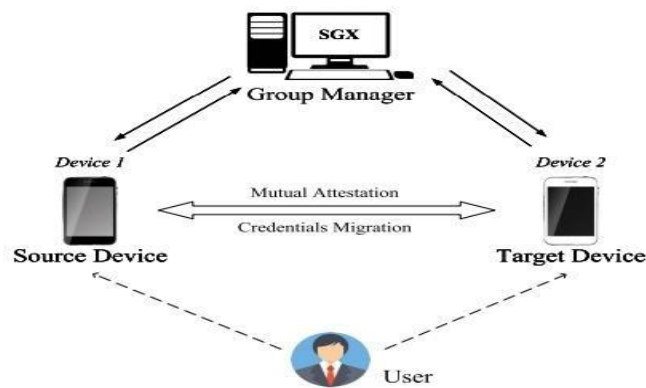


Figure 4.1.1: Architecture diagram

3-Tier Architecture:

The three-tier software architecture (a three layer architecture) emerged in the 1990s to overcome the limitations of the two-tier architecture. The third tier (middle tier server) is between the user interface (client) and the data management (server) components. This middle tier provides process management where business logic and rules are executed and can accommodate hundreds of users (as compared to only 100 users with the two tier architecture) by providing functions such as queuing, application execution, and database staging.

The three tier architecture is used when an effective distributed client/server design is needed that provides (when compared to the two tier) increased performance, flexibility, maintainability, reusability, and scalability, while hiding the complexity of distributed processing from the user. These characteristics have made three layer architectures a popular choice for Internet applications and net-centric information systems.

The system architecture consists of a user device, trusted execution environment, MA-TEECM server, and credential repository. The user device initiates the credential migration request, which is then authenticated by the MA-TEECM server using mutual anonymous authentication protocol. The Credential Migration Manager migrates the credentials between trusted execution environments, while the Trust Manager ensures trust is maintained throughout the process. The Interoperability Manager enables seamless communication and credential migration, and the migrated credentials are stored in the credential repository.

. Advantages of Three-Tier:

- Separates functionality from presentation.
- Clear separation – better understanding.
- Changes limited to well define components.
- Can be running on WWW.
- Effective network performance.

SYSTEM DESIGN

System Design Introduction:

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

4.2 FLOW OF EVENTS

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

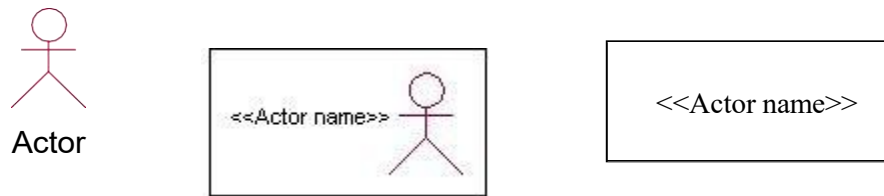
4.3 UML DIAGRAMS

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both main functions and secondary functions such as administration.
- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

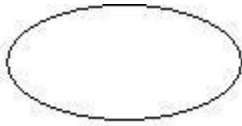
The actors identified in this system are:

- a. System Administrator
- b. Customer
- c. Customer Care

Identification of usecases:

Usecase: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behavior the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.
- Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What usecases will support and maintains the system?

4.3.1 CONSTRUCTION OF USECASE DIAGRAM

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



Figure 4.3.1.1: Use Case Diagram

4.3.2 SEQUENCE DIAGRAMS

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

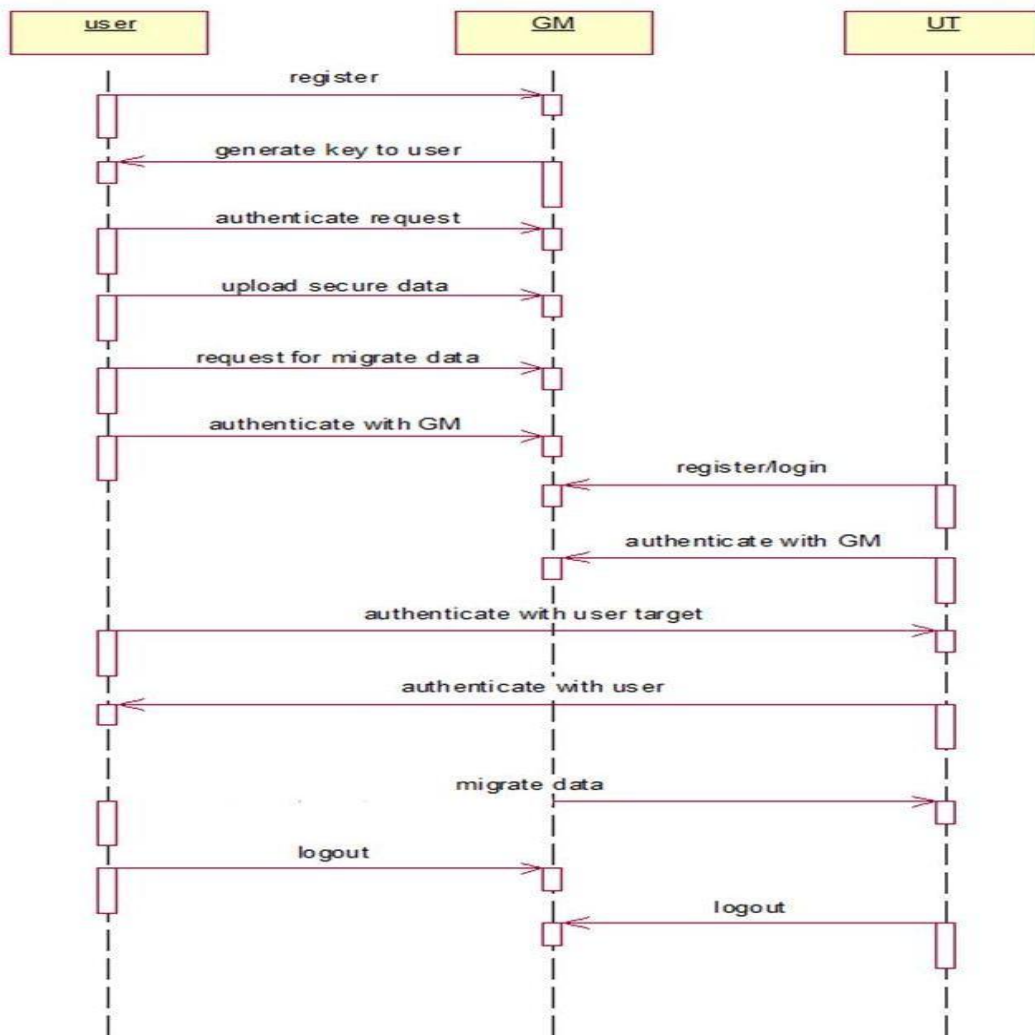


Figure 4.3.2.1: Sequence diagram

4.3.3. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

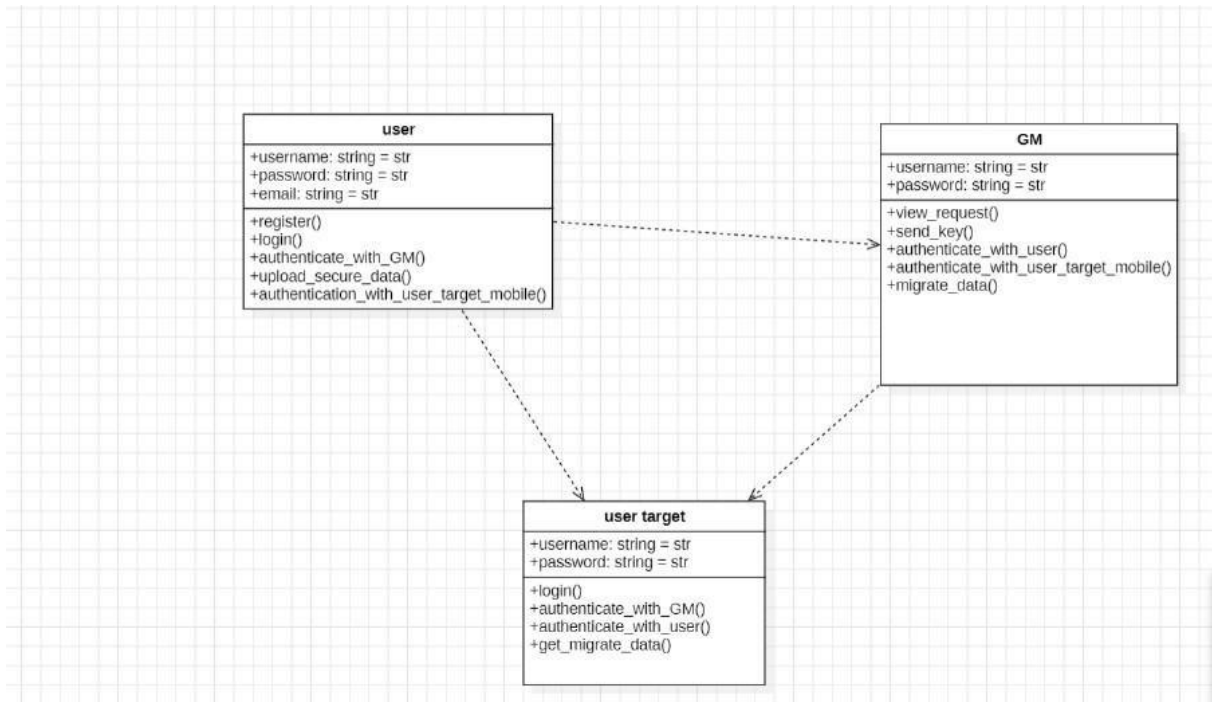


Figure 4.3.3.1: Class Diagram

4.3.4 ACTIVITY DIAGRAM

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

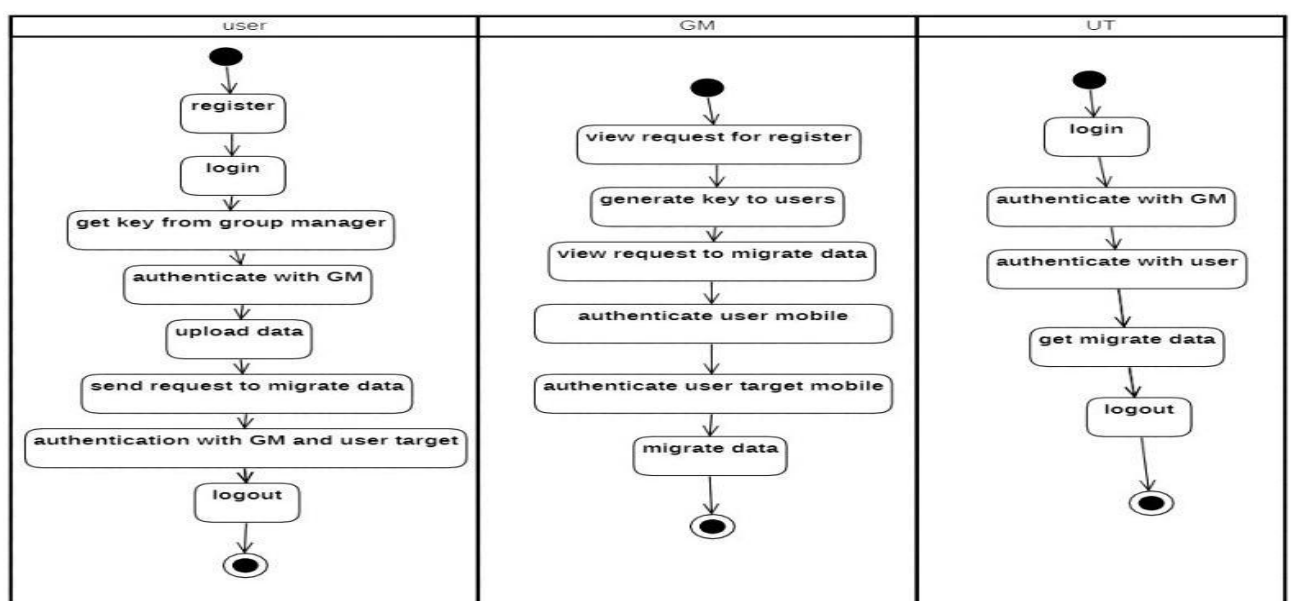
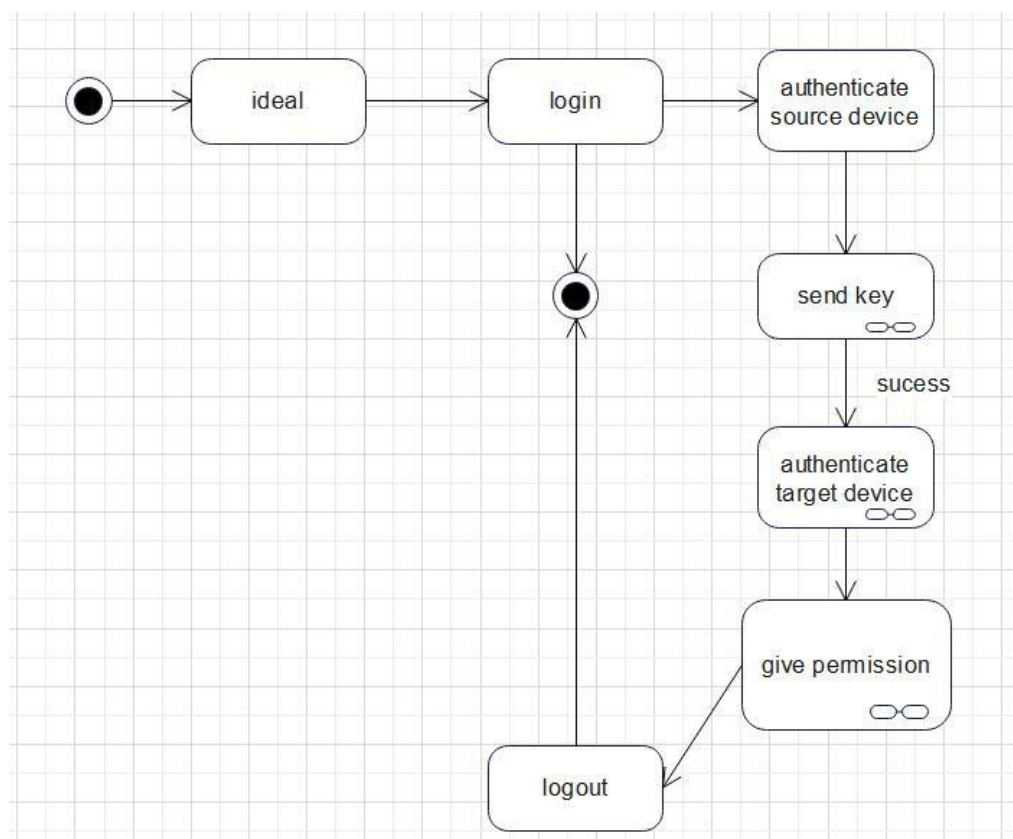
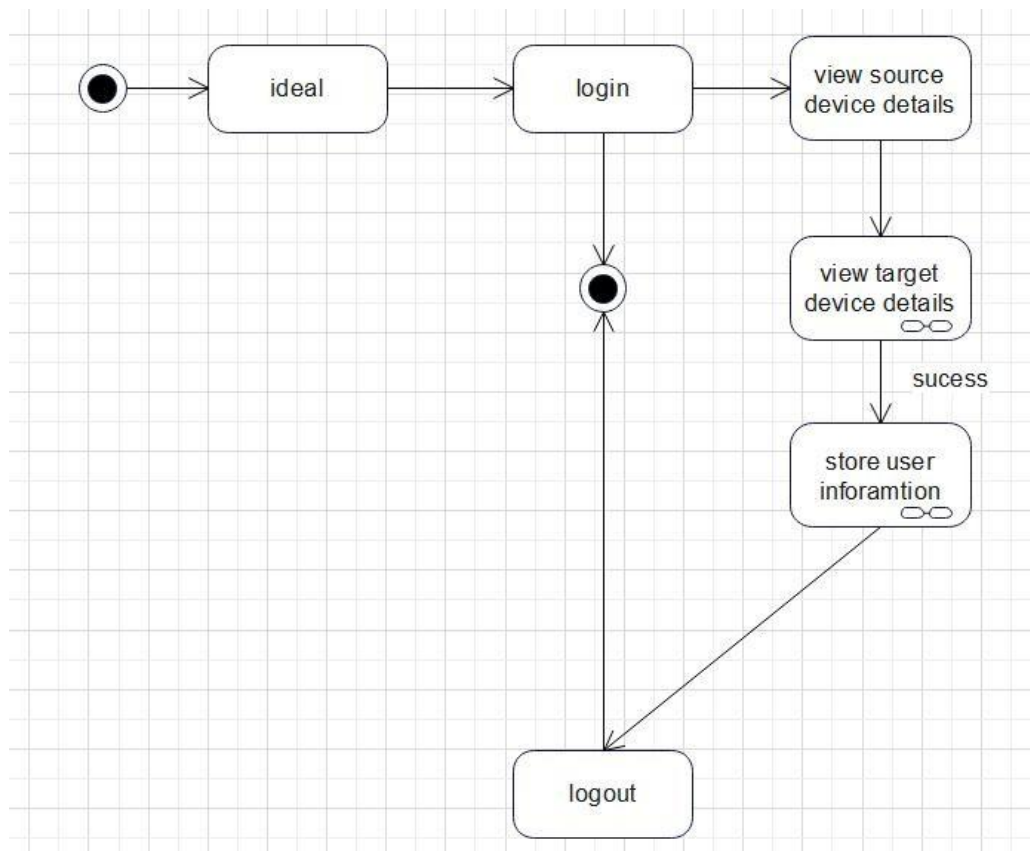


Figure 4.3.4.1: Activity Diagram

4.3.5 STATE CHART DIAGRAM



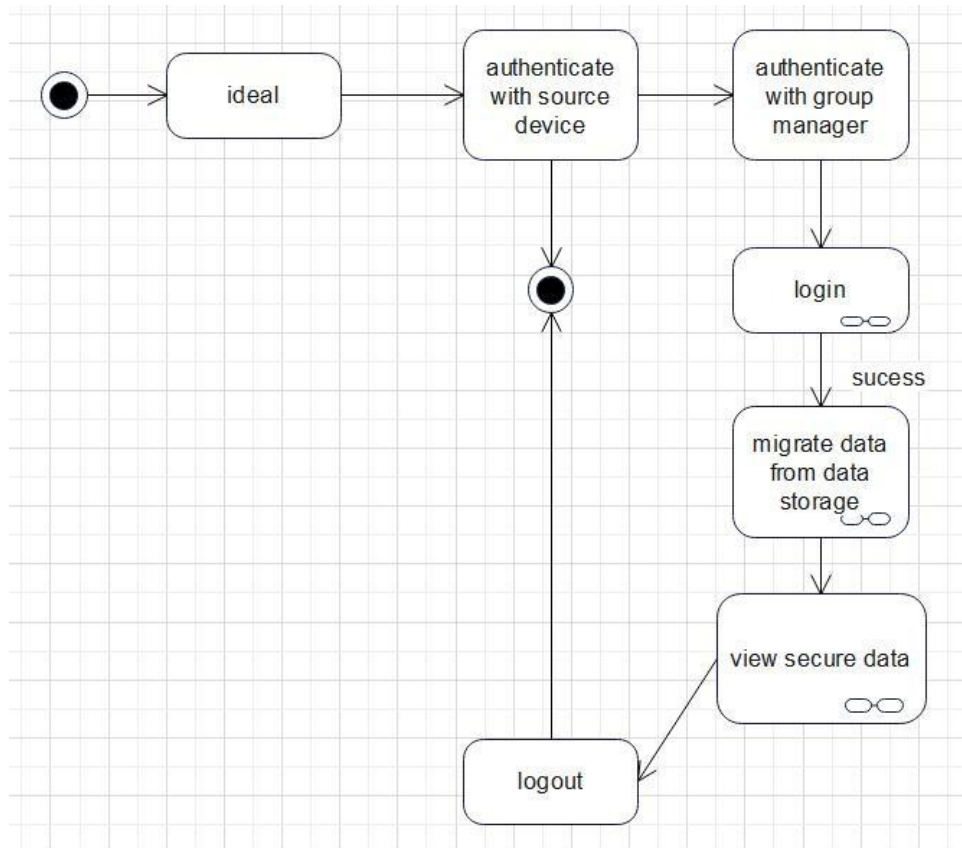


Figure 4.3.5.1: State chart diagram

4.3.6 COLLABORATION DIAGRAM

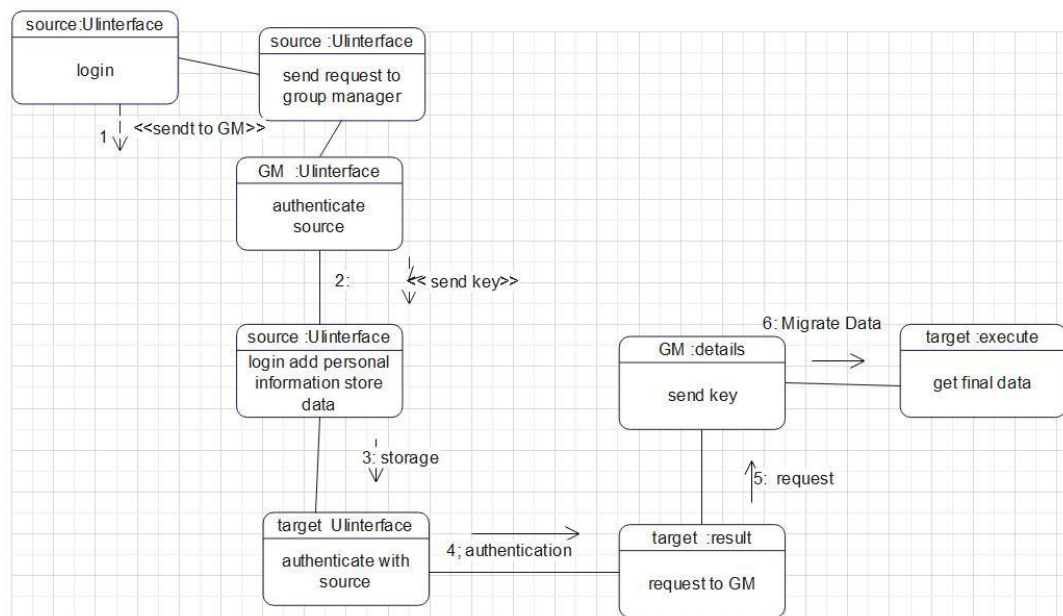


Figure 4.3.6.1: Collaboration Diagram

4.3.7 COMPONENT DIAGRAM

A component diagram is a visual representation of the components in a system, their relationships, and interactions. In the context of the major project, the component diagram illustrates the various components involved in the Mutual Anonymous Authentication (MAA) based credentials migration technology for mobile trusted execution environments. The Credential Migration Module encrypts and compresses data, transferring it to Secure Storage. The TEE Manager initializes and configures the TEE environment, ensuring secure boot and firmware updates. The Network Communication Module establishes secure connections between components, enabling data transfer and authentication.

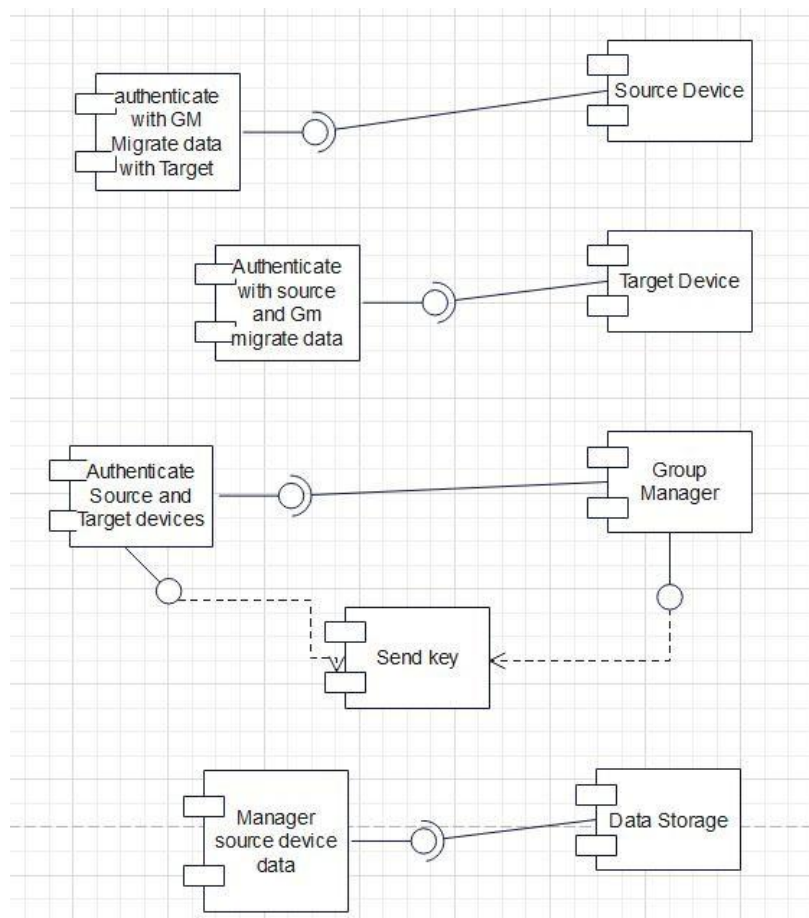


Figure 4.3.7.1: Component Diagram

4.3.8 DEPLOYMENT DIAGRAM

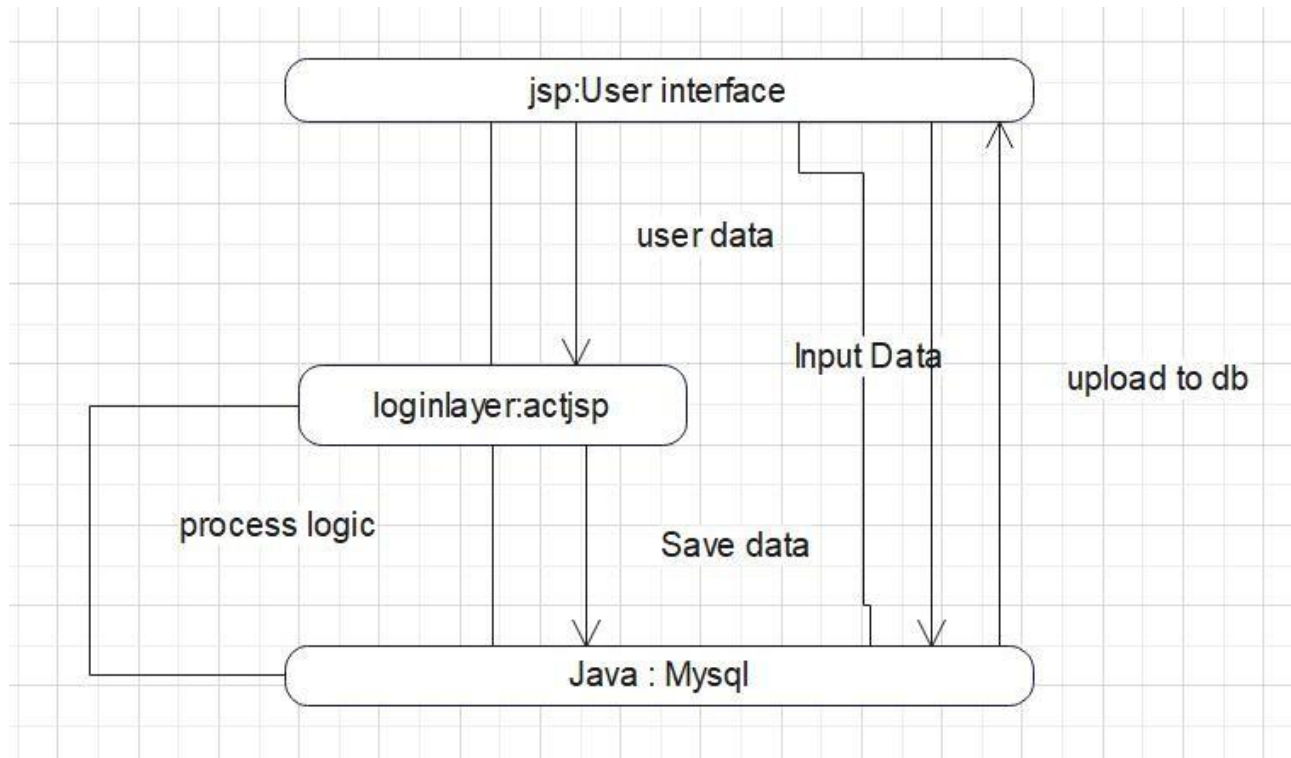


Figure 4.3.8.1: Deployment Diagram

CHAPTER-5

TESTING & VALIDATION

5.1 INTRODUCTION

Testing is the debugging program is one of the most critical aspects of the computer programming triggers, without programming that works, the system would never produce an output of which it was designed. Testing is best performed when user development is asked to assist in identifying all errors and bugs. The sample data are used for testing. It is not quantity but quality of the data used the matters of testing. Testing is aimed at ensuring that the system was accurately an efficiently before live operation commands.

Testing objectives:

The main objective of testing is to uncover a host of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with intent of finding an error.

- 1 A successful test is one that uncovers an as yet undiscovered error.
- 2 A good test case is one that has probability of finding an error, if it exists.
- 3 The test is inadequate to detect possibly present errors.
- 4 The software more or less confirms to the quality and reliable standards.

5.2 LEVELS OF TESTING

Code Testing: This examines the logic of the program. For example, the logic for updating various sample data and with the sample files and directories were tested and verified. A fundamental aspect of code testing is constructing a robust foundation of tests that cover diverse scenarios and edge cases. These tests act as a safety net, providing continuous feedback on the functionality and correctness of the code. By diligently testing their codebase, developers can identify and address issues early on, minimizing the time and effort spent on debugging and maintenance in the long run. Furthermore, code testing plays a pivotal role in guaranteeing the overall quality and reliability of software. Thorough testing not only enhances customer satisfaction but also prevents potential issues that could lead to revenue loss or negative user experiences. By implementing a comprehensive testing plan, the software development process becomes more efficient, instilling confidence in both the development team and end users.

Specification Testing: Executing this specification starting what the program should do and how it should performed under various conditions. Test cases for various situation and combination of conditions in all the modules are tested. Test cases are derived from the specifications by identifying the inputs and outputs of the system and then determining the conditions under which the inputs should produce the expected outputs. These conditions can be specified in terms of equivalence classes, boundary values, or other criteria. Once the test cases have been derived, they can be executed to validate the system's behavior. The specification can be in the form of a requirements document, design document, or code. The tester then writes test cases that exercise the different parts of the specification.

Integration Testing: The integration testing process comes after unit testing. It is mainly used to test the data flow from one module or component to other modules. In integration testing, the test engineer tests the units or separate components or modules of the software in a group. The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units. In simple words, we can say that integration testing aims to evaluate the accuracy of communication among all the modules.

System Testing: The third level of software testing is system testing, which is used to test the software's functional and non-functional requirements. It is end-to-end testing where the testing environment is parallel to the production environment. In the third level of software testing, we will test the application as a whole system. To In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system. In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements. check the end-to-end flow of an application or the software as a user is known as System testing.

Acceptance Testing: The last and fourth level of software testing is acceptance testing, which is used to evaluate whether a specification or the requirements are met as per its delivery. The software has passed through three testing levels (Unit Testing, Integration Testing, System Testing). Some minor errors can still be identified when the end-user uses the system in the actual scenario. In simple words, we can say that Acceptance testing is the squeezing of all the testing processes that are previously done.

Unit Testing: In the unit testing we test each module individually and integrate with the overall system. Unit testing focuses verification efforts on the smallest unit of software design in the module. This is also known as module testing. The module of the system is tested separately. This

testing is carried out during programming stage itself. In the testing step each module is found to work satisfactorily as regard to expected output from the module. There are some validation checks for fields also. For example the validation check is done for varying the user input given by the user which validity of the data entered. It is very easy to find error debut the system. Unit testing helps tester and developers to understand the base of code that makes them able to change defect causing code quickly. Unit testing helps in the documentation. Unit testing fixes defects very early in the development phase that's why there is a possibility to occur a smaller number of defects in upcoming testing levels. It helps with code reusability by migrating code and test cases.

Each Module can be tested using the following two Strategies:

1. Black Box Testing
2. White Box Testing

BLACK BOX TESTING

What is Black Box Testing?

Black box testing is a software testing techniques in which functionality of the software under test (SUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on the software requirements and specifications.

In Black Box Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.



The above Black Box can be any software system you want to test. For example : an operating system like Windows, a website like Google ,a database like Oracle or even your own custom application. Under Black Box Testing , you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

Black box testing - Steps

Here are the generic steps followed to carry out any type of Black Box Testing.

- Initially requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

There are many types of Black Box Testing but following are the prominent ones -

- Functional testing – This black box testing type is related to functional requirements of a system; it is done by software testers.
- Non-functional testing – This type of black box testing is not related to testing of a specific functionality, but non-functional requirements such as performance, scalability, usability.
- Regression testing – Regression testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

WHITE BOX TESTING

White Box Testing is the testing of a software solution's internal coding and infrastructure. It focuses primarily on strengthening security, the flow of inputs and outputs through the application, and improving design and usability. White box testing is also known as clear, open, structural, and glass box testing.

It is one of two parts of the "box testing" approach of software testing. Its counter-part, blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing. The term "whitebox" was used because of the see-through box concept. The clear box or whitebox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "black box testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested

What do you verify in White Box Testing ?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

To give you a simplified explanation of white box testing, we have divided it into two basic steps. This is what testers do when testing an application using the white box testing technique:

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include manual testing, trial and error testing and the use of testing tools as we will explain further on in this article. Whitebox testing, also known as clear box or glass box testing, is a

software testing technique that involves examining the internal workings of a software application or system to ensure that it meets the required specifications and works as expected. Start by reviewing the source code of the software application or system to understand its internal workings. Identify test cases based on the code review, focusing on critical paths, loops, conditional statements, and data structures. Create test data to exercise the code paths and validate the software's behavior. Write test scripts to automate the testing process, using programming languages like Java, Python, or C#. Run the tests, executing the test scripts and observing the software's behavior. Verify the results, comparing the actual output with the expected output. Debug and fix any defects or issues found during testing. Repeat the process until all test cases have been executed and the software meets the required specifications.

5.3 TEST CASES

Below table shows the test case for the admin. The details if admin are stored in the database such as admin name, admin name etc. Admin is responsible for handling the transactions. The admin sets password and user name. Whether all the passwords and usernames are correct or not is checked. Because by using these details the user will login into the cloud and can view the details of file stored.

Test case 1

The test case 1 tests the login of admin's. The passwords and username is given if the correct password and usernames are entered the login will be successful. If any wrong password and username the login will be denied.

Sl # Test Case : -	UTC-1
Name of Test: -	Register Source Module
Items being tested: -	Registration process
Sample Input: -	Fill form enter details
Expected output: -	Registration data stored in db and validation message
Actual output: -	registration successful
Remarks: -	Pass.

Table 5.3.1: Test Case1

Test case 2

The test case 2 is for checking the users. Again for the user's password and username is given based on only the correct input the users will be able to login.

Sl # Test Case : -	UTC-2
Name of Test: -	Key from block manager
Items being tested: -	Key verification from manager
Sample Input: -	View key by source
Expected output: -	Generate security key and send to source user
Actual output: -	View security key
Remarks: -	sucess.

Table 5.3.2 :Test Case 2

Test Case 3

The test case 3 shows for the file upload and it is being stored in different cloudwhich is called block generation and storing and hence this file is divided and stored .If the storage and block generation is not successful than we can store the file

Sl # Test Case : -	ITC-1
Name of Test: -	Verify personal data storage
Item being tested: -	Upload data like bank details , personal data
Sample Input: -	Bank details , personal data
Expected output: -	Data stored in database
Actual output: -	User can view personal data
Remarks: -	Pass.

Table 5.3.3 : Test Case 3

Test case 4

Test case 4 is for file upload where the uploaded is downloaded. During download if any one of the cloud fails the recovery should start automatic. If the recovery is successful all the blocks will be generated and merged together. With the help of network coding the blocks are stored in the cloud.

Sl # Test Case : -	STC-1
Name of Test: -	Target device verify
Item being tested: -	Authenticate target device with manager and source device
Sample Input: -	User details
Expected output: -	Authentication verification status
Actual output: -	Target device can login
Remarks: -	Pass

Table 5.3.4 : Test Case 4

5.4 EXECUTABLE CODE

Databaseconnection

```
package databaseconnection;  
import java.sql.*;
```

```
public class databasecon  
{  
    static Connection co;  
    public static Connection getconnection()  
    {  
        try  
        {  
            Class.forName("com.mysql.jdbc.Driver");  
            co =  
DriverManager.getConnection("jdbc:mysql://localhost:3306/securedata","root","root");  
        }  
        catch(Exception e)  
        {  

```

```

        System.out.println("Database Error"+e);
    }
    return co;
}
}

```

Decryption

```

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;

```

```

import javax.crypto.Cipher;
import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

```

```

public class decryption {
//public static void main(String args[])
//{
//    Scanner s=new Scanner(System.in);
//    System.out.println("Enter encrypted Text and key");
//    String text=s.next();
//    String key=s.next();
//    new decryption().decrypt(text,key);
//}

```

```

    public String decrypt(String txt, String skey)
    { String decryptedtext = null;
    try {
        //converting string to secretkey
        byte[] bs = Base64.decode(skey);
        SecretKey sec = new SecretKeySpec(bs, "AES");

```

```

        System.out.println("converted string to seretkey:" + sec);
        System.out.println("secret key:" + sec);
        Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
        aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating cipher encryption using secretkey
        byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt); //encrypting data
        // System.out.println("cipher text:"+byteCipherText);
        aesCipher.init(Cipher.DECRYPT_MODE, sec, aesCipher.getParameters());//initiating cipher
decryption
        byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText);
        decryptedtext = new String(byteDecryptedText);
        System.out.println("Decrypted Text:" + decryptedtext);
    } catch (Exception e)
    { System.out.println(e);
    }
    return decryptedtext;
}
String decrypt(String str, SecretKey sec) {
    throw new UnsupportedOperationException("Not supported yet."); //To change body of
generated methods, choose Tools | Templates.
}
}
}

```

Download

```

<%@page import="java.sql.*"%>
<%@page import="Distributed.Dbconnection"%>
<%@ page session="true" %>
<%
String username = session.getAttribute("user").toString();
String task = request.getParameter("task");
String location = request.getParameter("location");
String status = null;
String person = null;
try{
    Connection con=Dbconnection.getConnection();
    PreparedStatement ps=con.prepareStatement("insert into task values(?,?,?,?)");
    ps.setString(1,username);

```

```

ps.setString(2,task);
ps.setString(3,location);
ps.setString(4,person);
ps.setString(5,status);
int i=ps.executeUpdate();
if(i>0)
{
response.sendRedirect("owner_upload.jsp?msg=Registered");
}
else { response.sendRedirect("ownerreg.jsp?msg1=Failed")
;
}
}%>
<%
}
catch(Exception e)
{
out.println(e);
}
}%>

```

View task

```

<%@page import="java.sql.ResultSet"%>
<%@page import="java.sql.PreparedStatement"%>
<%@page import="Distributed.Dbconnection"%>
<%@page import="java.sql.Connection"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Distributed Time-Sensitive Task Selection in Mobile Crowd Sensing</title>
<meta http-equiv="Content-type" content="text/html; charset=utf-8" />
<link rel="stylesheet" href="css/style.css" type="text/css" media="all" />
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/jquery.jcarousel.js"></script>

```



```

<script src="js/cufon-yui.js" type="text/javascript" charset="utf-8"></script>
<script src="js/Chaparral_Pro.font.js" type="text/javascript" charset="utf-8"></script>
<script type="text/javascript" src="js/jquery-func.js"></script>
<link rel="shortcut icon" type="image/x-icon" href="css/images/favicon.ico" />
</head>

<%
if(request.getParameter("mg")!=null){%>

    <script>alert('Login Sucessfully..!')</script>
}

<%}
if(request.getParameter("msg1")!=null){%>

    <script>alert('Owner Registration Failed..!')</script>
}
<%
}
%>
<%
if(request.getParameter("m3")!=null){%>

    <script>alert('username already exists')</script>
}

<%}
if(request.getParameter("")!=null){%>

    <script>alert('Owner Registration Failed..!')</script>
}
<%
}
%>
<body>
<!-- START PAGE SOURCE -->
<div id="header">

```

```

<br>
<div class="shell">
  <h1>Distributed Time-Sensitive Task Selection in Mobile Crowd Sensing</h1>
  <div class="search">

    </div>
  </div>
</div>
<div id="navigation">
  <div class="shell">
    <ul>
      <li><a href="ownerhome.jsp">HOME</a></li>
      <li><a href="owner_upload.jsp">Upload Task</a></li>
      <li><a class="active" href="owner_viewtask.jsp">View Task</a></li>
      <li><a href="owner_assign_reward.jsp">Assign Reward</a></li>
      <li><a href="owner_download.jsp">Delivery Status</a></li>
      <li><a href="logout.jsp">Logout</a></li>
    </ul>
  </div>
</div>
<div id="featured">
  <div class="shell">
    <div class="slider-carousel">
      <ul>
        <li>
          <div class="info">
            <p> We provide a new efficient RDPC protocol based on homomorphic hash function. The
            new scheme is provably secure against forgery attack, replace attack and replay attack based on a
            typical security model. To support data dynamics, an operation record table (ORT) is introduced to
            track operations on file blocks. We further give a new optimized implementation for the ORT which
            makes the cost of accessing ORT nearly constant. Moreover, we make the comprehensive
            performance analysis which shows that our scheme has advantages in computation and
            communication costs. Prototype implementation and experiments exhibit that the scheme is feasible
            for real applications. </p>
          </div>
          <div class="image"> <a href="#"></a> </div>

```

```

        <div class="cl">&nbsp;</div>
    </li>
    <li>
        <div class="info">
            <p>We provide a new efficient RDPC protocol based on homomorphic hash function. The
            new scheme is provably secure against forgery attack, replace attack and replay attack based on a
            typical security model. To support data dynamics, an operation record table (ORT) is introduced to
            track operations on file blocks. We further give a new optimized implementation for the ORT which
            makes the cost of accessing ORT nearly constant. Moreover, we make the comprehensive
            performance analysis which shows that our scheme has advantages in computation and
            communication costs. Prototype implementation and experiments exhibit that the scheme is feasible
            for real applications.</p>
        </div>
        <div class="image"> <a href="#"></a> </div>
        <div class="cl">&nbsp;</div>
    </li>

</ul>
</div>
</div>
<%

```

```

String username = session.getAttribute("user").toString();
String task = request.getParameter("task");
String location = request.getParameter("location");
Connection con = null;
con = Dbconnection.getConnection();
PreparedStatement pst=con.prepareStatement("select * from taskhandler where location =
"+location+" ");
ResultSet rs=pst.executeQuery();
%>
<div id="main">
    <div class="shell">
        <div id="main-boxes">
            <center>

```

```

    <p align="justify">
    <p><font color="black" size="5"> Assign Task </font></p><br/>

    <form name="myform" autocomplete="off" action="owner_viewtask2.jsp" method="post"
onsubmit="return validateform()">
    <table align="center" width="321">
    <tr>
    <td width="191" height="43"><font color="black">User Name </td>
    <td width="218"><input autocomplete="off" value="<%=username%>" name="username"
required="" placeholder="User Name" /></td>
    </tr>
    <tr>
    <td height="43"><font color="black">Task </td>
    <td width="218"><input type="text" value="<%=task%>" name="task" required=""
placeholder="Password" /></td>
    </tr>
    <tr>
    <td height="43"><font color="black"> Location</td>
    <td><input name="location" autocomplete="off" value="<%=location%>" required=""
placeholder="Email ID"/></td>
    </tr>
    <tr>
    <td height="43"><font color="black">Select Task Handler</td>
    <td><select name="th" style="width:170px;" required="">
    <option>--Select--</option>
    <%
while(rs.next()){
%>
    <option><%=rs.getString("username")%></option>
    <%
}
%>
</select></td>
    </tr>
    <tr>
    <td height="43" rowspan="3">

```



```

String username = request.getParameter("username");
String password = request.getParameter("password");
try{
    Connection con=Dbconnection.getConnection();
    Random s = new Random();
    int otp = s.nextInt(10000 - 5000) + 25000 ;
    Statement st = con.createStatement();
    ResultSet rs = st.executeQuery("select * from taskhandler where email= '"+username+"' and
password='"+password+"'");
    if(rs.next())
    {
        String user = rs.getString("username");
        session.setAttribute("user",user);
        String email = rs.getString("email");
        session.setAttribute("email",email);
        response.sendRedirect("taskhandlerhome.jsp?m1");
    }
    else
    {
        response.sendRedirect("taskhandler.jsp?msg1=LoginFail");
    }
}
catch(Exception e)
{
    System.out.println("Error in emplogact"+e.getMessage());
}
%>

```

Web page

```

!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">

<head>

    <title>Online Collaborative Data Caching using edge computing</title>

```

```

<meta http-equiv="Content-type" content="text/html; charset=utf-8" />

<link rel="stylesheet" href="css/style.css" type="text/css" media="all" />

<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>

<script type="text/javascript" src="js/jquery.jcarousel.js"></script>

<script src="js/cufon-yui.js" type="text/javascript" charset="utf-8"></script>

<script src="js/Chaparral_Pro.font.js" type="text/javascript" charset="utf-8"></script>

<script type="text/javascript" src="js/jquery-func.js"></script>

<link rel="shortcut icon" type="image/x-icon" href="css/images/favicon.ico" />

</head>

<body>

<!-- START PAGE SOURCE -->

<div id="header">

    <br>

    <div class="shell">

        <h1> COLLABORATIVE DATA CACHING </h1>

        <div class="search">

            </div>

        </div>

    </div>

    <div id="navigation">

        <div class="shell">

            <ul>

                <li><a class="active" href="index.html">HOME</a></li>

                <li><a href="mobilesource.jsp"> MOBILE SOURCE</a></li>

                <li><a href="eserver.jsp">EDGE SERVER</a></li>

                <li><a href="cloud.jsp">CLOUD SERVER</a></li>

                <li><a href="mobiledestination.jsp">MOBILE DESTINATION</a></li>

```

```

</ul>

</div>

</div>

<div id="featured">

  <div class="shell">

    <div class="slider-carousel">

      <ul>

        <li>

          <div class="info">

            <p style="font-size: 20px;">Collaborative data caching is a testament to the fact
that in the world of distributed computing, cooperation and coordination can yield
significant benefits and Efficiently Managing Frequently Accessed Data.</p>

            </div>

            <div class="image"> <a href="#"></a>

          </div>

          <div class="cl">&nbsp;</div>

        </li>

        <li>

          <div class="info">

            <p style="font-size: 20px;">Collaborative data caching in edge computing is the
key to faster, more efficient web page delivery by enabling distributed servers to work
together to store and serve frequently accessed content closer to the end-users. </p>

            </div>

            <div class="image"> <a href="#"></a>

          </div>

          <div class="cl">&nbsp;</div>

        </li>

      </ul>

```



```

        </div>

    </div>

</div>

<div id="main">

    <div class="shell">

        <div id="main-boxes">

            <br><br><br><br><br>        <br><br><br>

<center><p><font size="5" color="black">Welcome to Develop a Secure and Trust Based
Key Management Protocol for Cloud Computing Environments</font></p><br></center>

        </div>

        <br>

        <div class="cl">&nbsp;</div>

    </div>

</div>

<div class="footer">

    <div class="shell">

        <p class="rf"></a></p>

        <div style="clear:both;"></div>

    </div>

</div>

<script type="text/javascript">pageLoaded();</script>

<!-- END PAGE SOURCE -->

</body>

</html>

```

Database connection

```
package databaseconnection;
import java.sql.*;
public class databasecon
{
    static Connection co;

    public static Connection getconnection()
    {
        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            co =DriverManager.getConnection("jdbc:mysql://localhost:3306/securedata","root","root");
        }
        catch(Exception e)
        {
            System.out.println("Database Error"+e);
        }
        return co;
    }
}
```

Decryption

```
package action;
```

```
/**
```

```

*

* @author java2

*/

import com.sun.org.apache.xerces.internal.impl.dv.util.Base64;
import java.io.ByteArrayOutputStream;
import java.io.FileInputStream;
import java.io.FileWriter;
import java.util.Scanner;
import javax.crypto.Cipher;

import javax.crypto.KeyGenerator;
import javax.crypto.SecretKey;

import javax.crypto.spec.SecretKeySpec;
import javax.swing.JOptionPane;
import sun.misc.BASE64Decoder;
import sun.misc.BASE64Encoder;

public class decryption {

    //public static void main(String args[])

    //{

    //    Scanner s=new Scanner(System.in);

    //    System.out.println("Enter encrypted Text and key");

    //    String text=s.next();

    //    String key=s.next();

    //new decryption().decrypt(text,key);

    //}

    public String decrypt(String txt, String skey)
    { String decryptedtext = null;
    try {

        //converting string to secretkey
        byte[] bs = Base64.decode(skey);

```

```

        SecretKey sec = new SecretKeySpec(bs, "AES");
        System.out.println("converted string to seretkey:" + sec);
        System.out.println("secret key:" + sec);
    Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
    aesCipher.init(Cipher.ENCRYPT_MODE, sec);//initiating ciper encryption
    using secretkey

    byte[] byteCipherText = new BASE64Decoder().decodeBuffer(txt); //encrypting
    Data

    // System.out.println("ciper text:"+byteCipherText); aesCipher.init(Cipher.DECRYPT_MODE,
    sec,
    aesCipher.getParameters());//initiating ciper decryption

    byte[] byteDecryptedText = aesCipher.doFinal(byteCipherText); decryptedtext = new
    String(byteDecryptedText);

    System.out.println("Decrypted Text:" + decryptedtext);

    } catch (Exception e)
    { System.out.println(e);
    }

    return decryptedtext;

    }

    String decrypt(String str, SecretKey sec) {

    throw new UnsupportedOperationException("Not supported yet."); //To change body of
    generated methods, choose Tools | Templates.
    }
    }

```

Encryption

```

    */

    package collaborative;

    import com.sun.org.apache.xerces.internal.impl.dv.util.Base64; import
    java.io.ByteArrayOutputStream;

```

```

import java.io.FileInputStream;
import java.io.FileWriter; import
java.util.Scanner;
import javax.crypto.Cipher;

import javax.crypto.KeyGenerator; import
javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec; import javax.swing.JOptionPane;
import sun.misc.BASE64Encoder;
public class encryption {
public String encrypt(String text, SecretKey secretkey)
{ String plainData = null, cipherText = null;
try {
plainData = text; Cipher aesCipher = Cipher.getInstance("AES");//getting AES instance
    aesCipher.init(Cipher.ENCRYPT_MODE, secretkey);//initiating cipher
    encryption using secretkey

byte[] byteDataToEncrypt = plainData.getBytes();

byte[] byteCipherText = aesCipher.doFinal(byteDataToEncrypt);//encrypting

cipherText = new BASE64Encoder().encode(byteCipherText);//converting encrypted data to
    string
System.out.println("\n Given text : " + plainData + " \n Cipher Data : " + cipherText);

} catch (Exception e) { System.out.println(e);
}

return cipherText;

}
}

```

Download

```

<%@page import="java.sql.*"%>

<%@page import="Distributed.Dbconnection"%>

<%@ page session="true" %>

```

```

<%
String username = session.getAttribute("user").toString(); String task =
request.getParameter("task");

String location = request.getParameter("location"); String status
= null;
String person = null; try{
Connection con=Dbconnection.getConnection();

PreparedStatement ps=con.prepareStatement("insert into task values(?,?,?,?)")
ps.setString(1,username);
ps.setString(2,task);
ps.setString(3,location);

ps.setString(4,person);
ps.setString(5,status);
int i=ps.executeUpdate(); if(i>0)
{
response.sendRedirect("owner_upload.jsp?msg=Registered");
}

else{ response.sendRedirect("ownerreg.jsp?msg1=Failed");
}
}%>

<%
}

catch(Exception e)
{
out.println(e);
}

}%>

package collaborative;

```

```

import java.io.File;

import java.io.FileInputStream;

import org.apache.commons.net.ftp.FTPClient;

/*
 *      To change this license header, choose License Headers in Project Properties.
 *
 *      To change this template file, choose Tools | Templates
 *
 *      and open the template in the editor.
 */

/**
 *
 *      @author java2
 */

public class Ftpcon {

    FTPClient client = new FTPClient();
    FileInputStream fis = null;
    boolean status;

    public boolean upload(File file,String fname) { try
    { client.connect("ftp.drivehq.com"); client.login("shival8",
    "shivagmail@8"); client.enterLocalPassiveMode();
    fis = new FileInputStream(file);

    status = client.storeFile("/cloud/" + fname, fis);

    client.logout(); fis.close();
    } catch (Exception e)

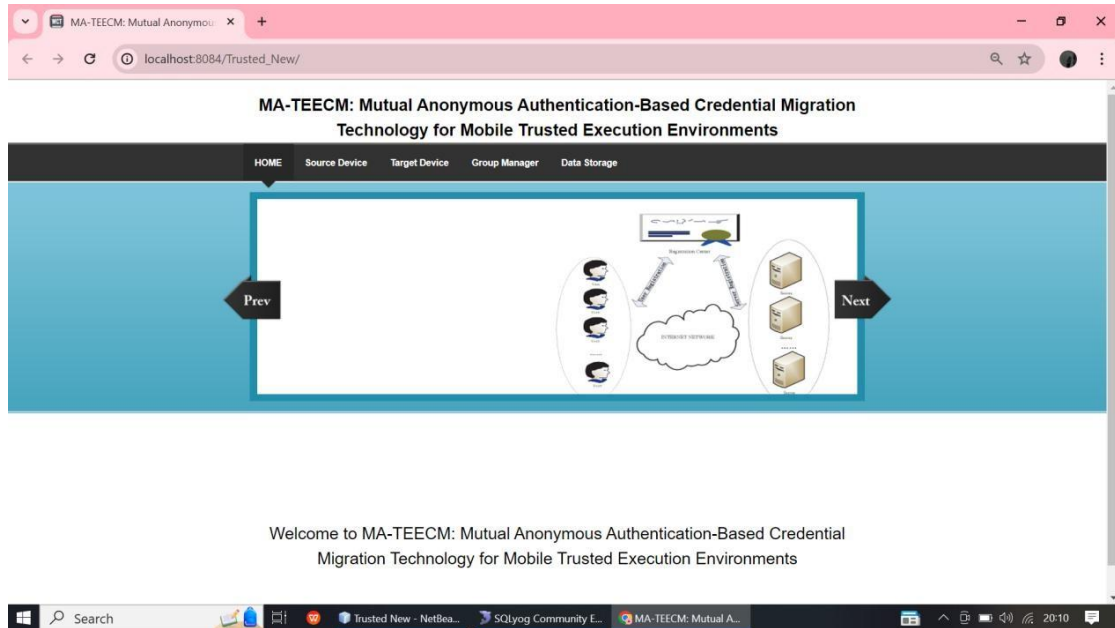
```

```
{ System.out.println(e);  
}  
  
if (status) { System.out.println("success");  
return true;  
} else {  
  
System.out.println("failed"); return false;  
}  
  
}
```

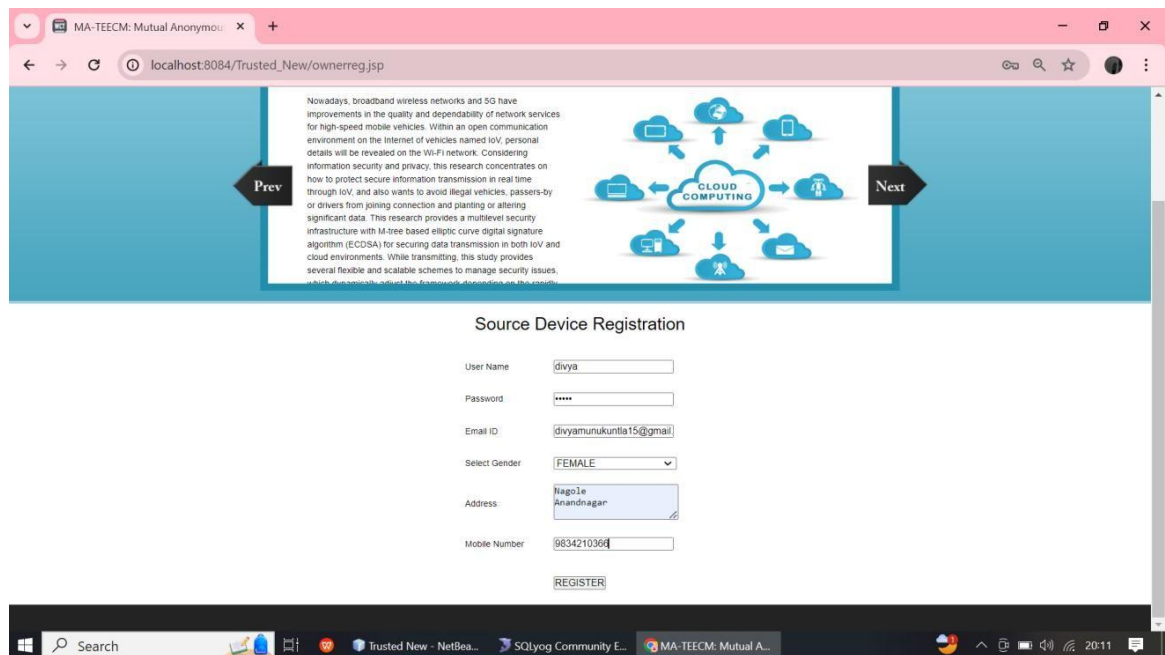

CHAPTER - 6

SCREENSHOTS

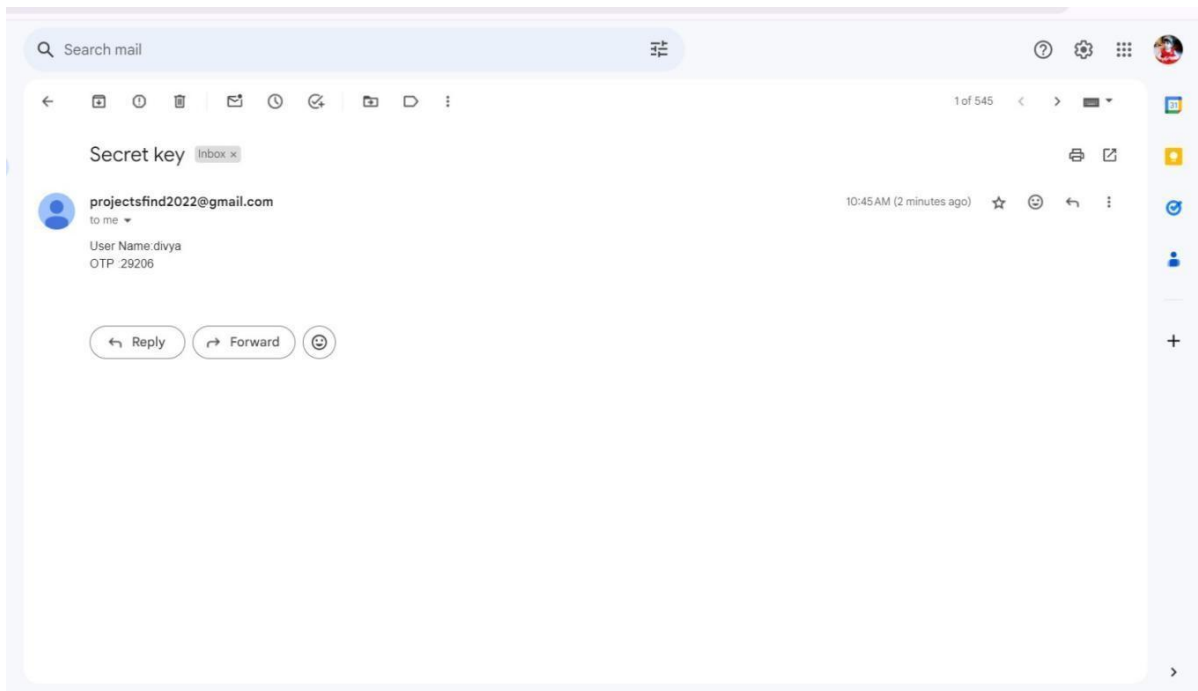
6.1 SCREENSHOTS OF MA-TEECM



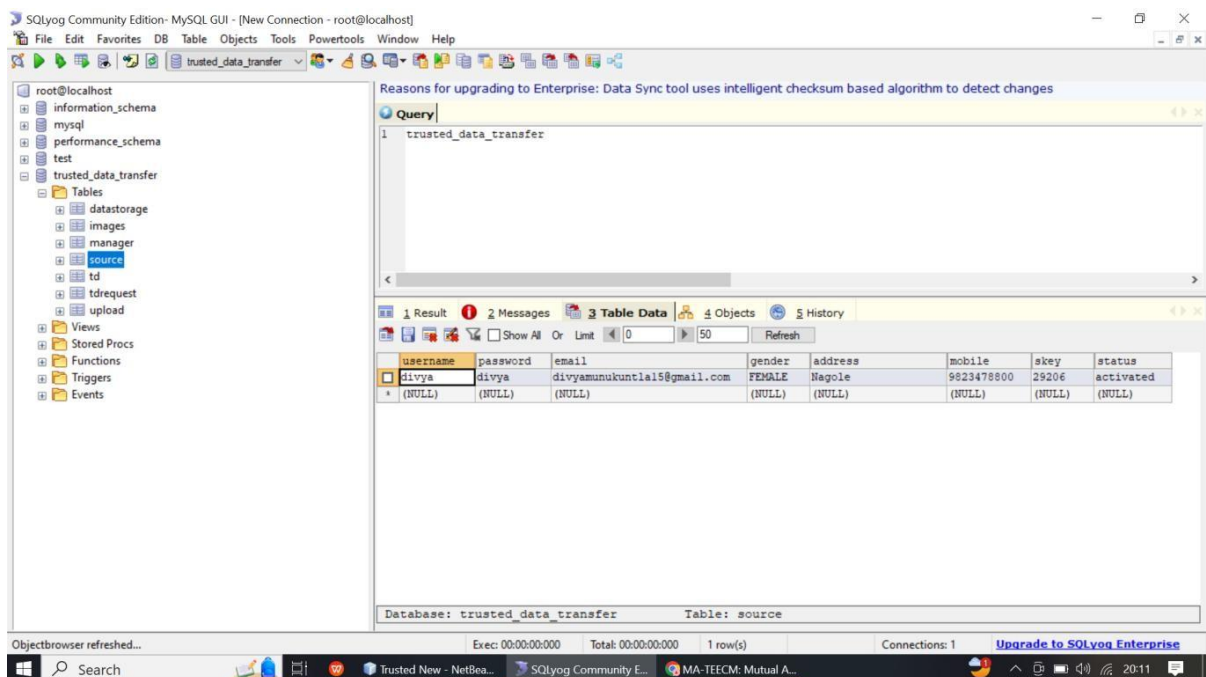
Screenshot 6.1.1: home page of MA-TEECM



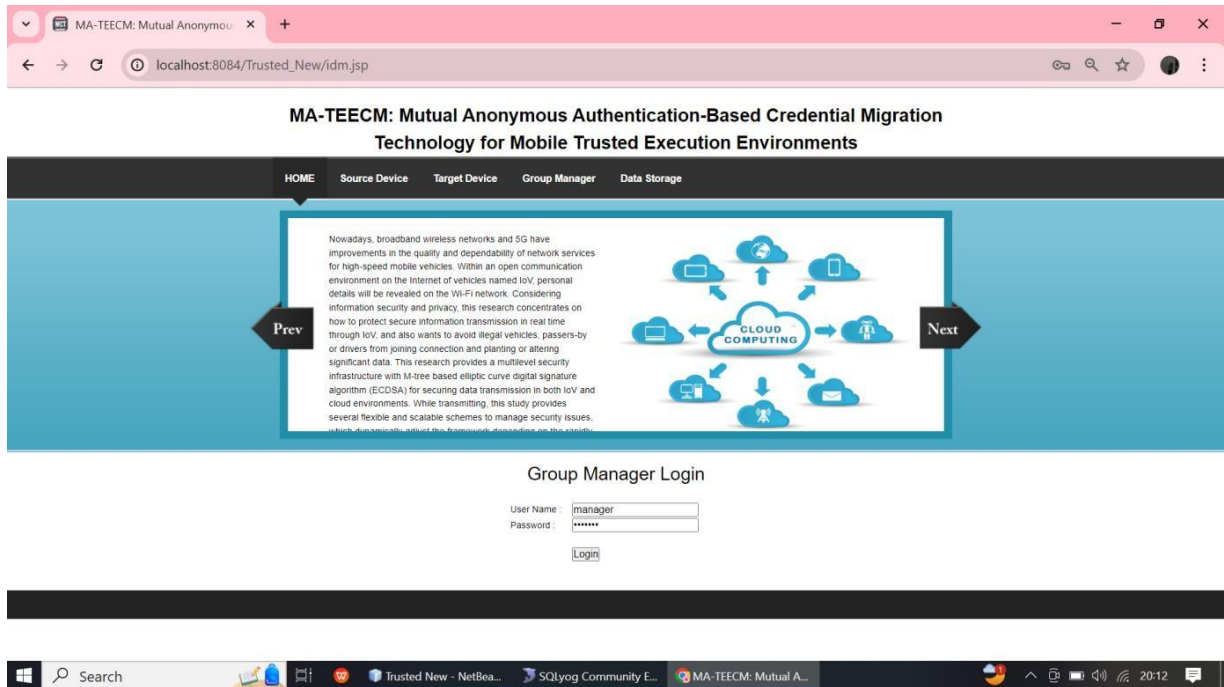
Screenshot 6.1.2: Source Device Registration



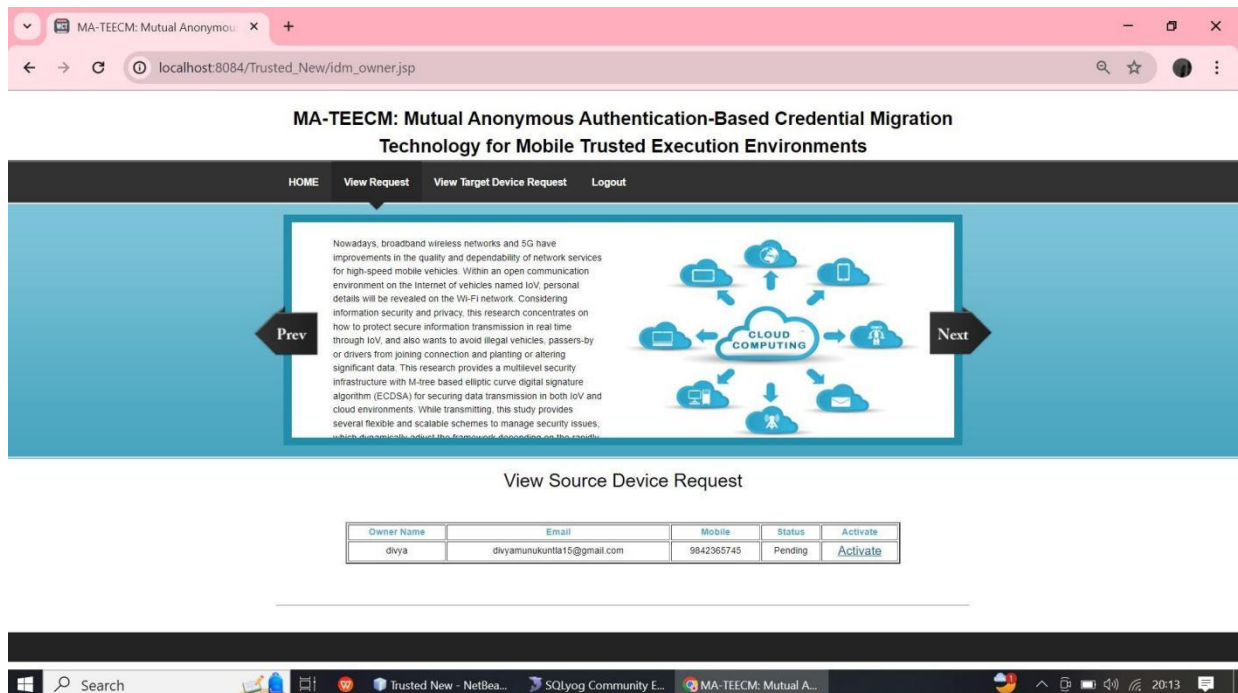
Screenshot 6.1.3: sending a skey by email



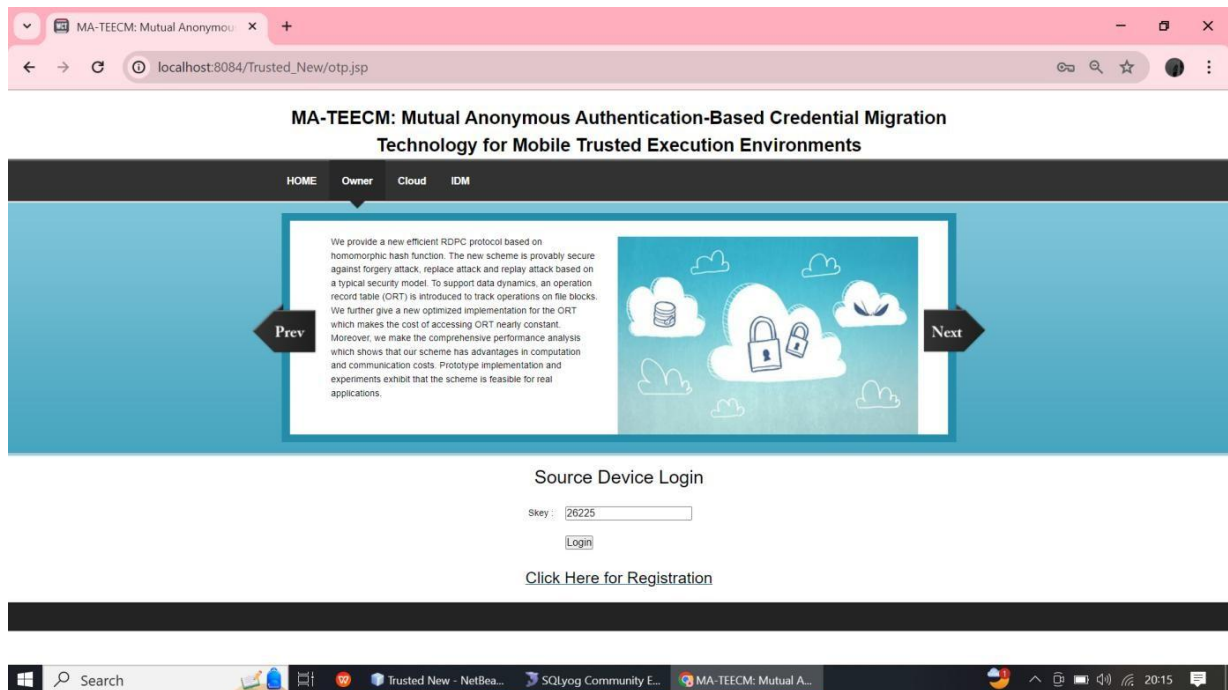
Screenshot 6.1.4: view skey in sql



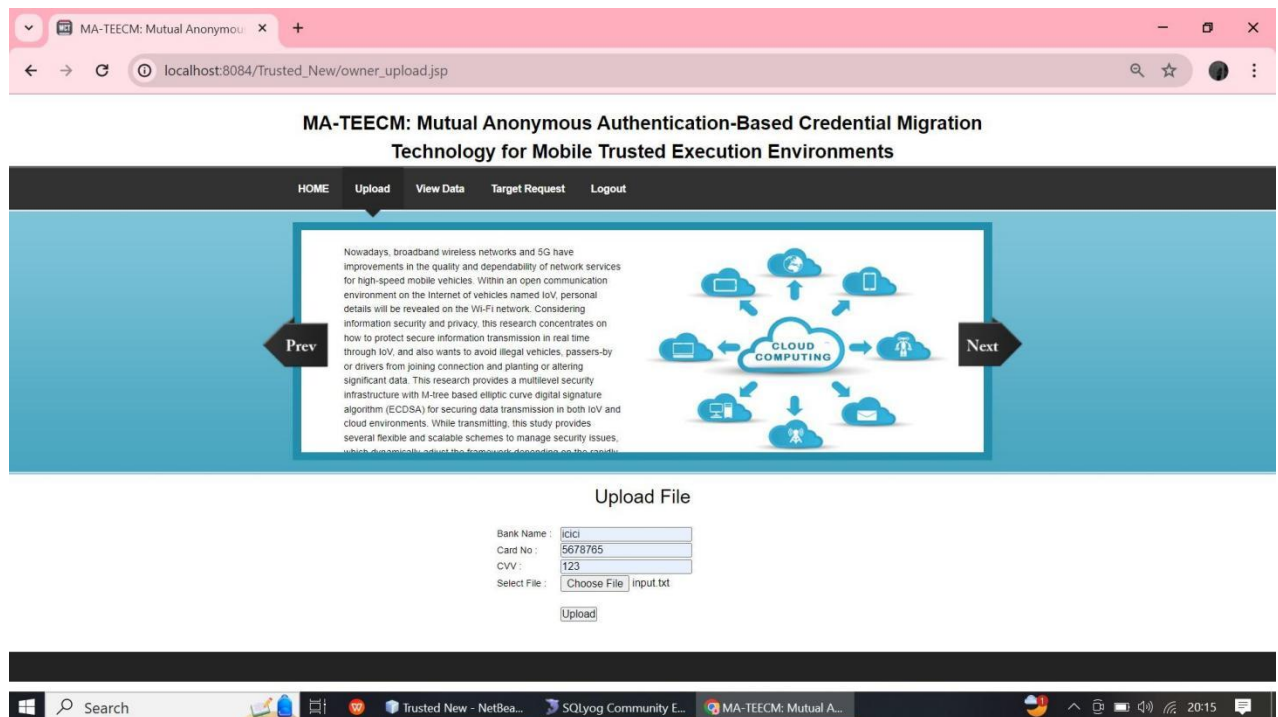
Screenshot 6.1.5: login group manager



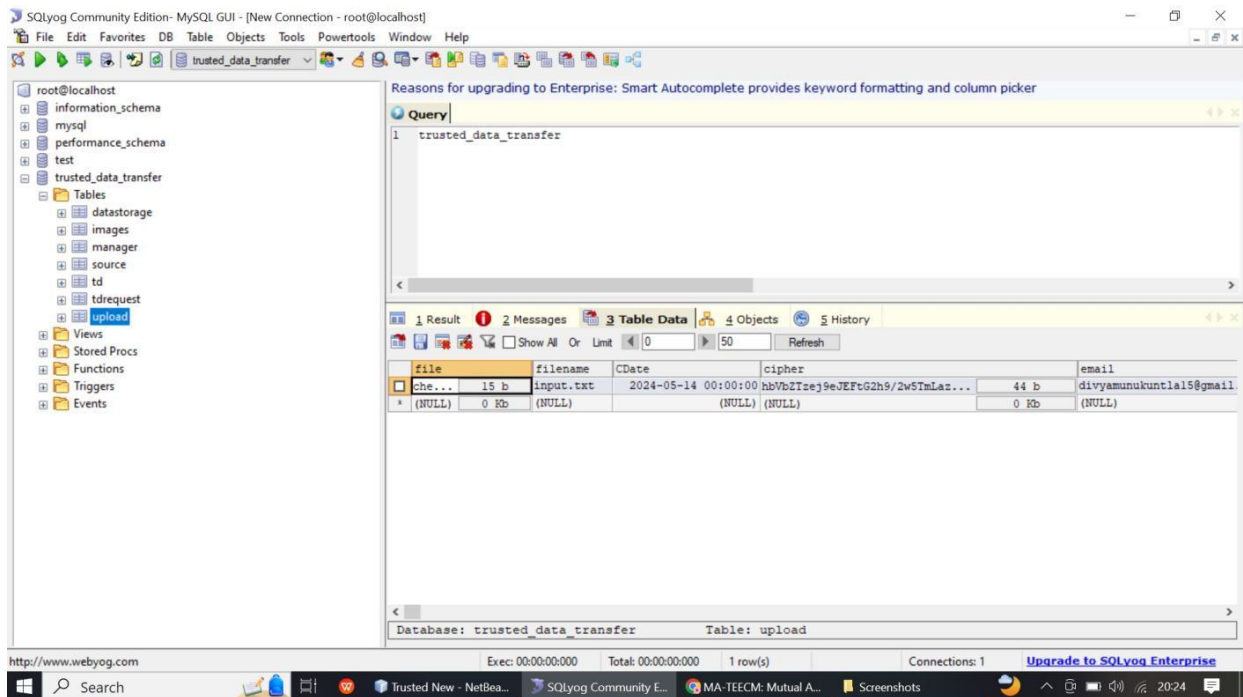
Screenshot 6.1.6: Activate the source device request



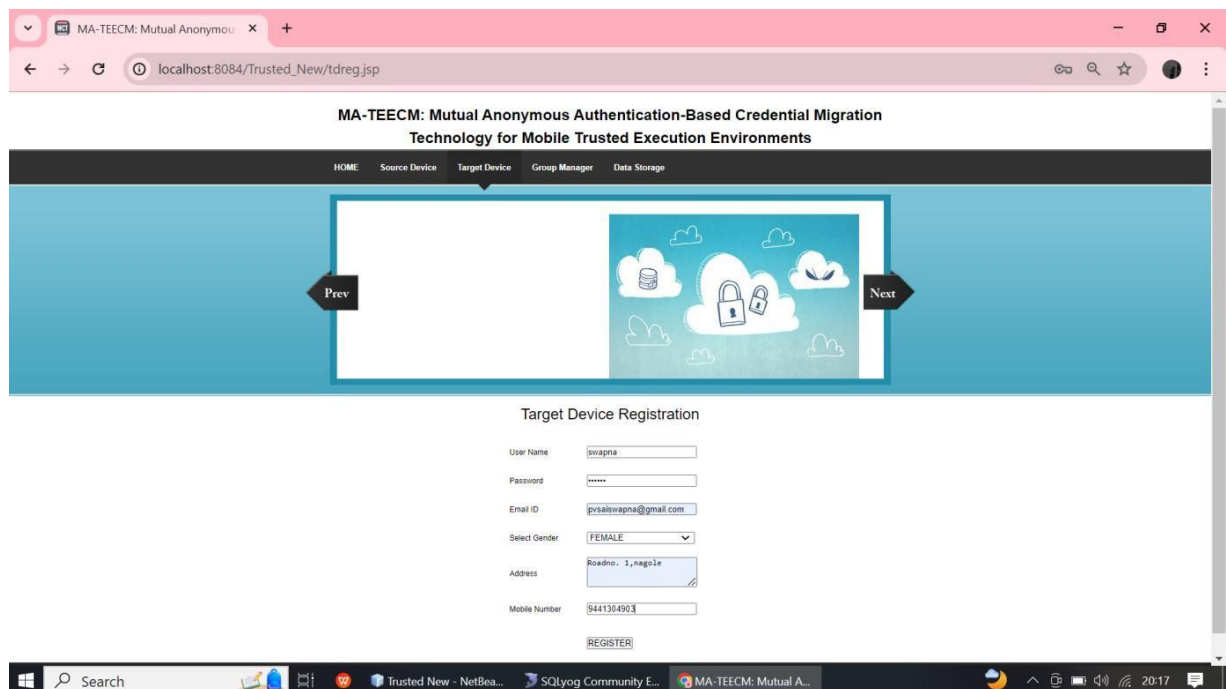
Screenshot 6.1.7: Login source device using skey



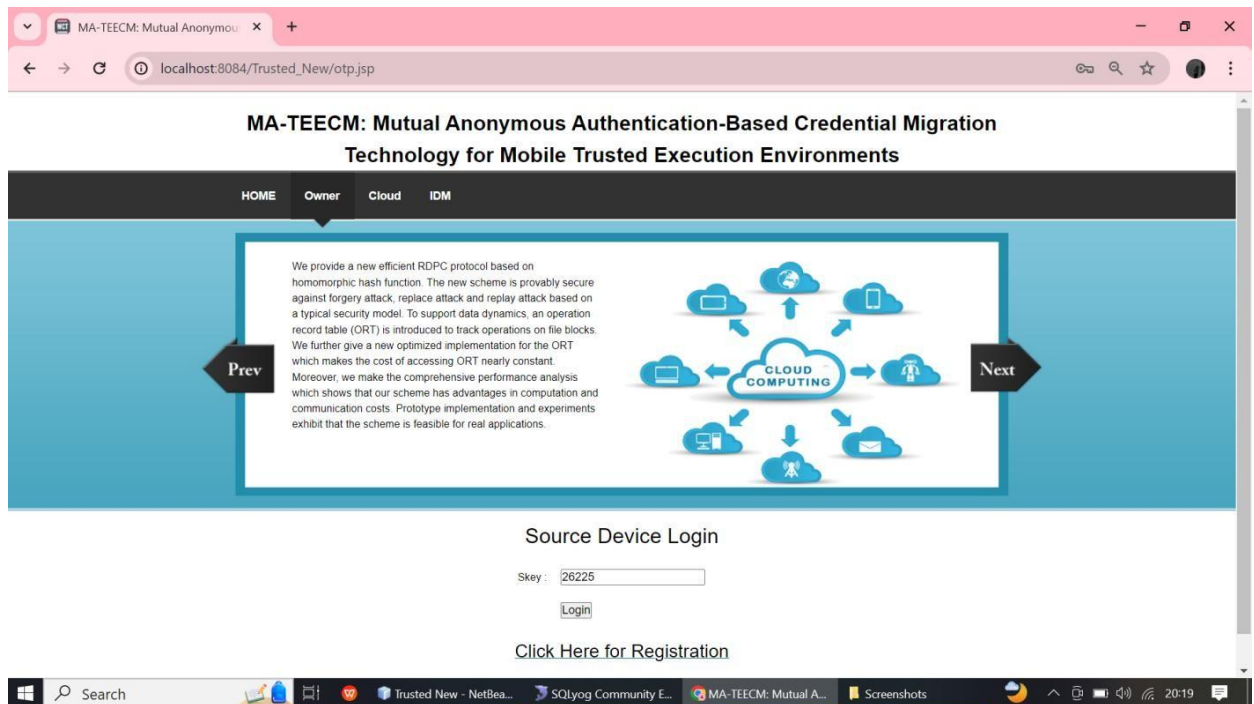
Screenshot 6.1.8: Fill details and upload a file



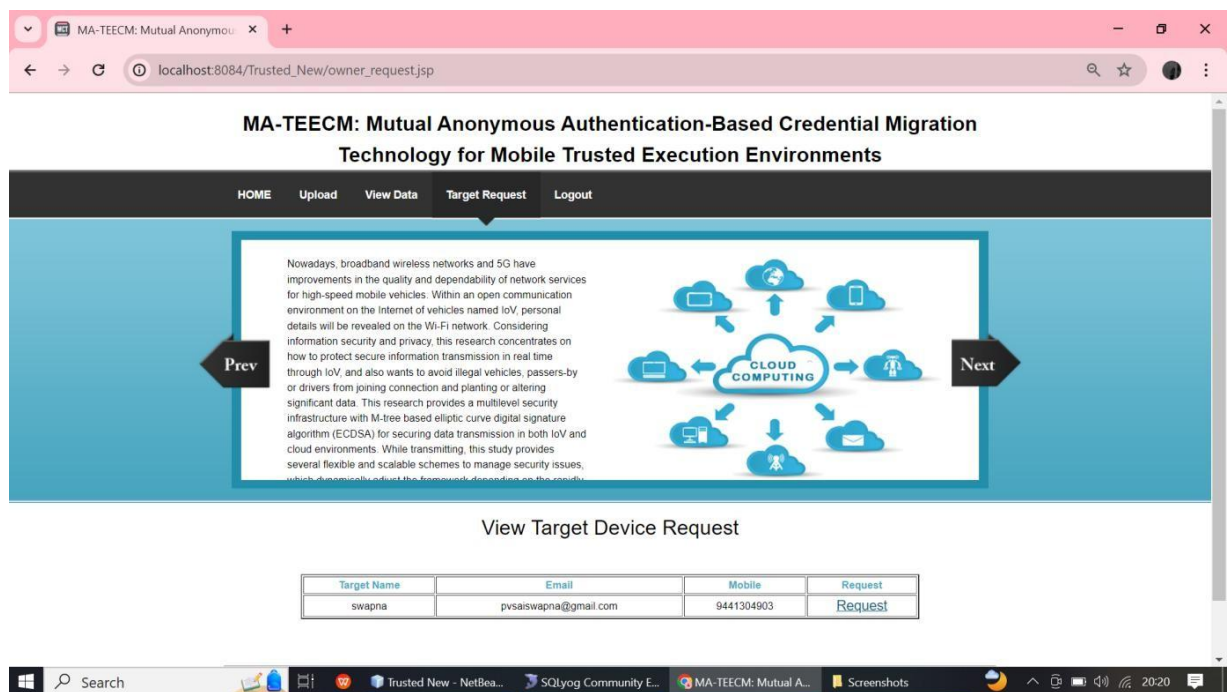
Screenshot 6.1.9: view the uploaded file in sql



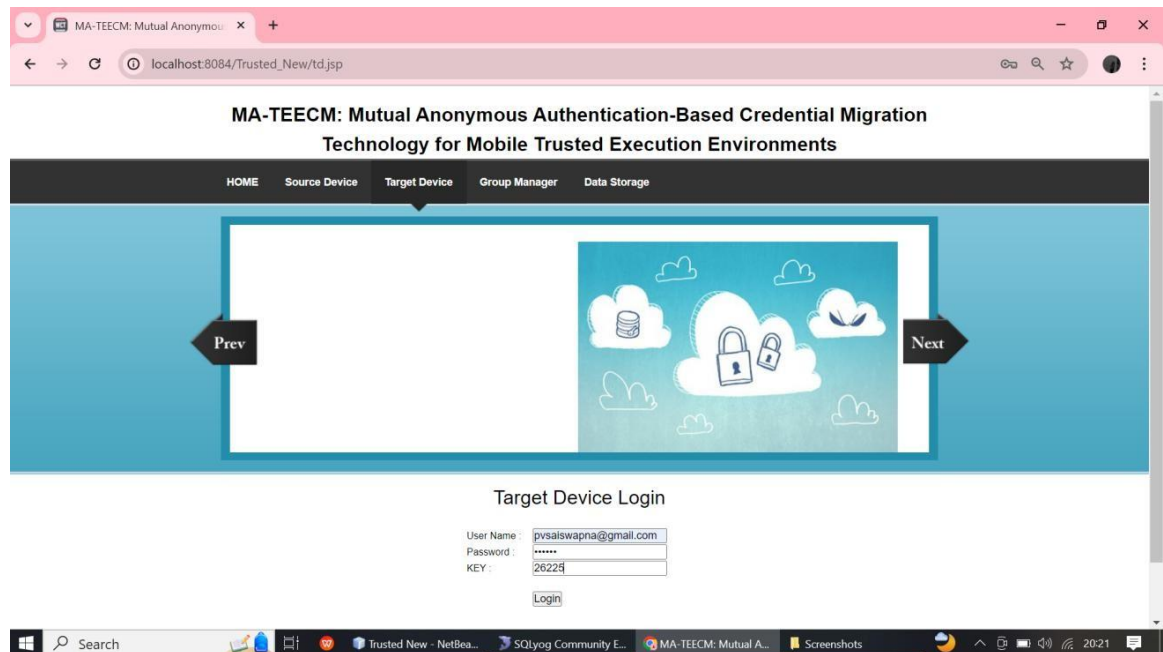
Screenshot 6.1.10: Registration of Target Device



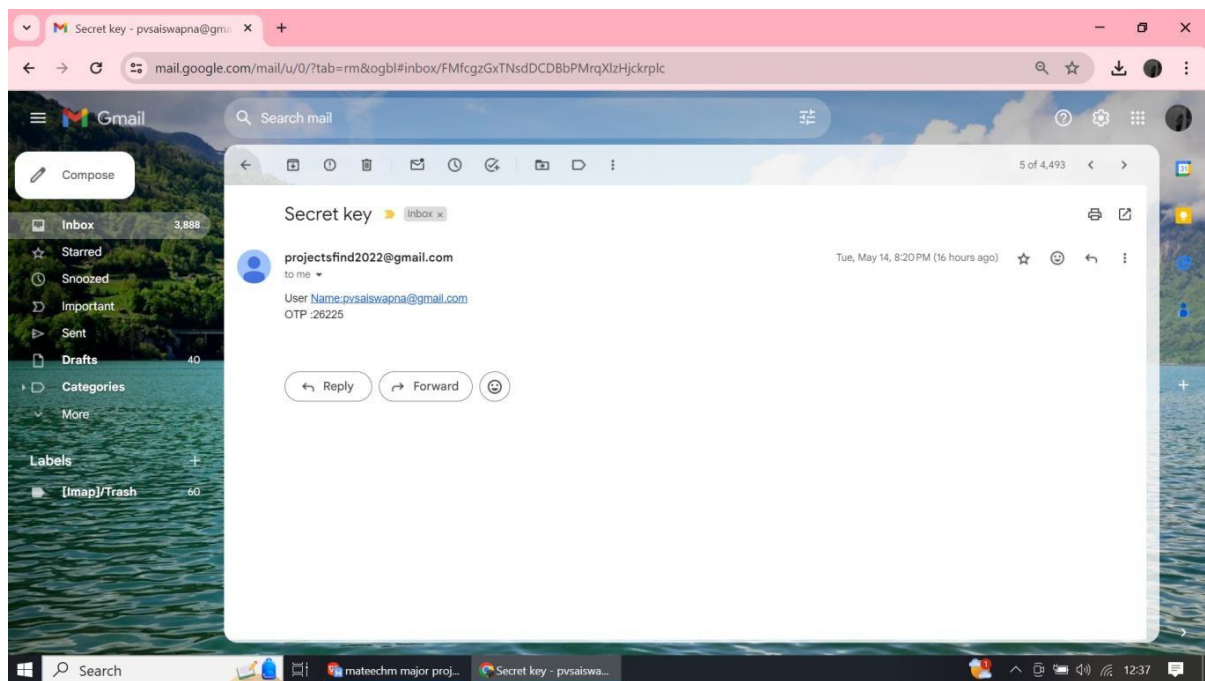
Screenshot 6.1.11: Login to source device with the skey



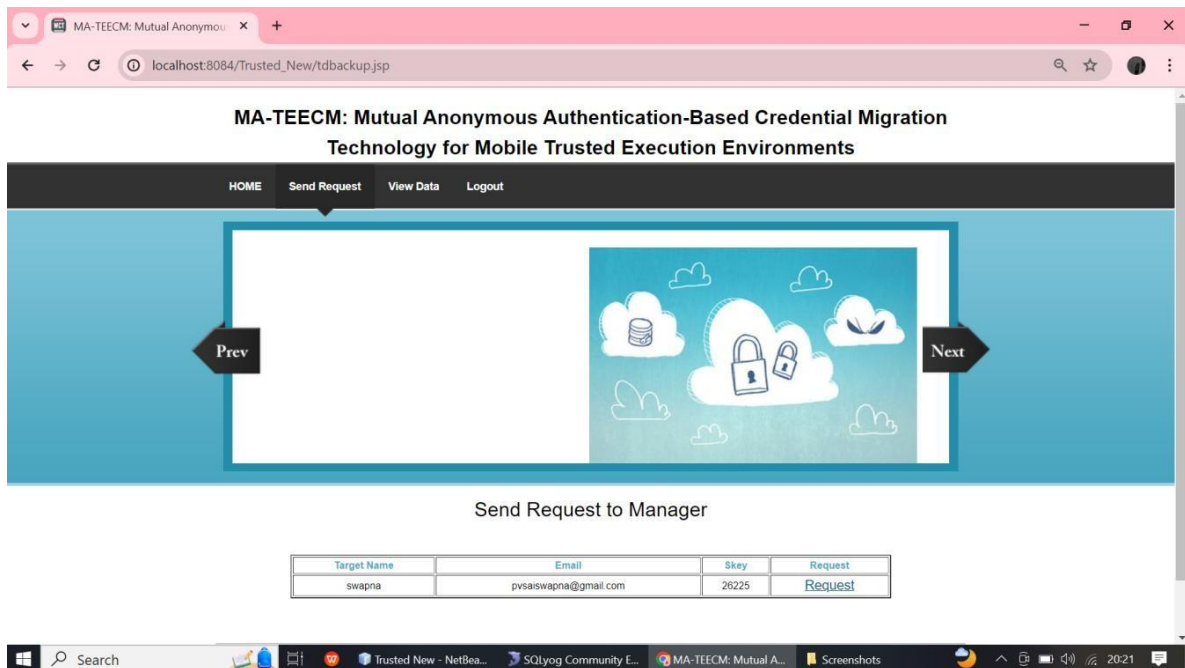
Screenshot 6.1.12: view target device request



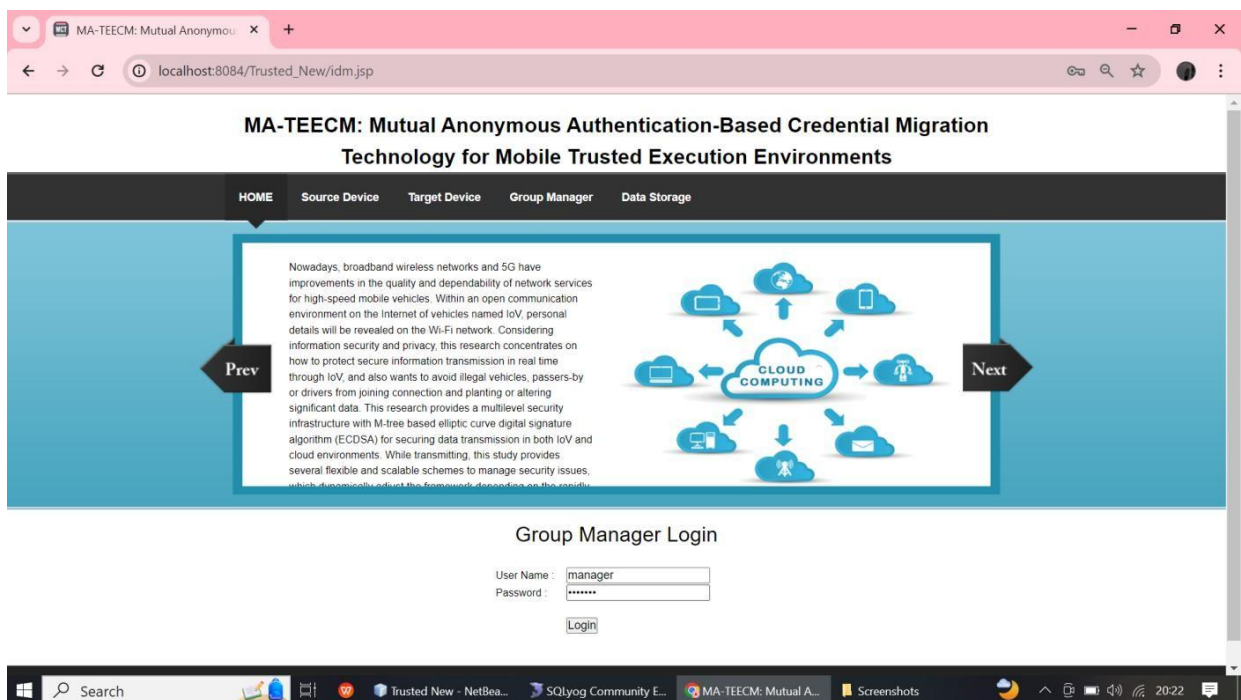
Screenshot 6.1.13: Login Target Device using skkey



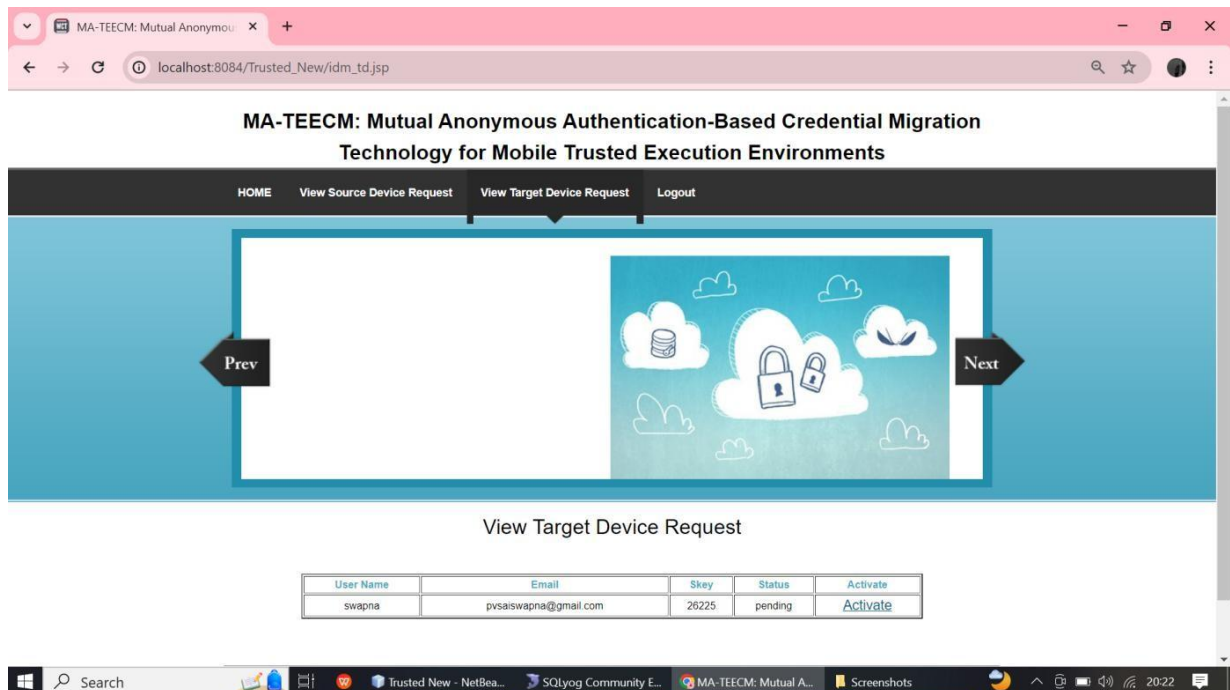
Screenshot 6.1.14: View the skkey from the registered mail in target device



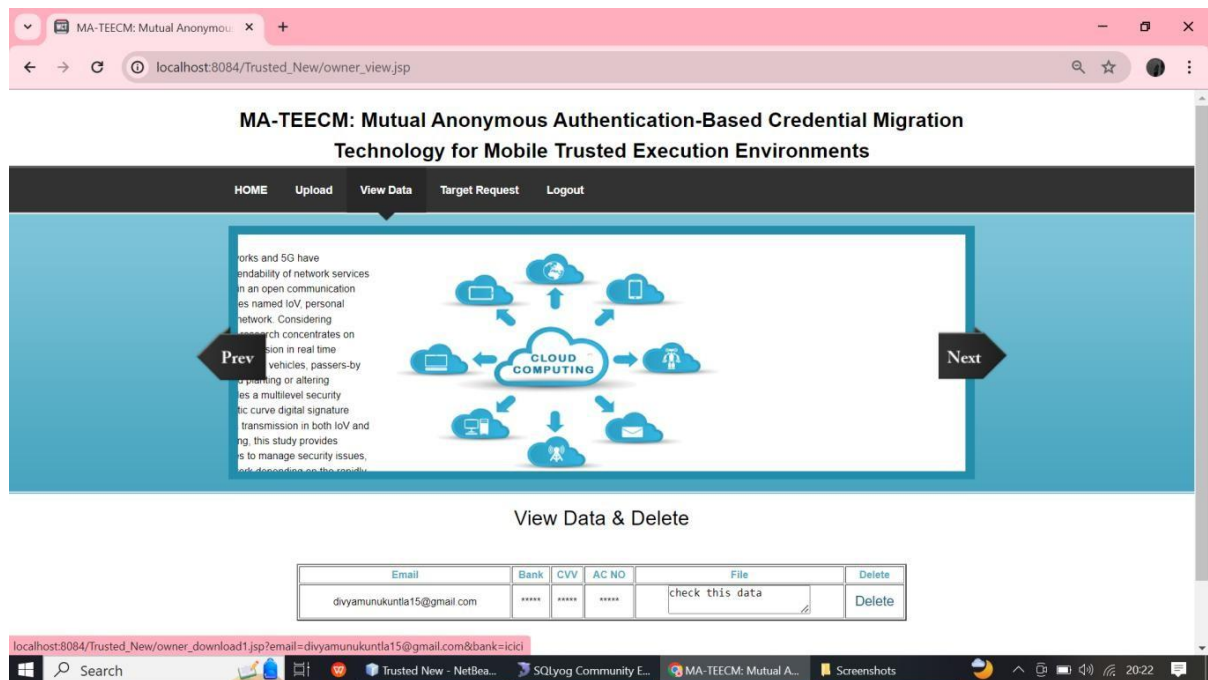
Screehshot 6.1.15: Send request to Group Manager



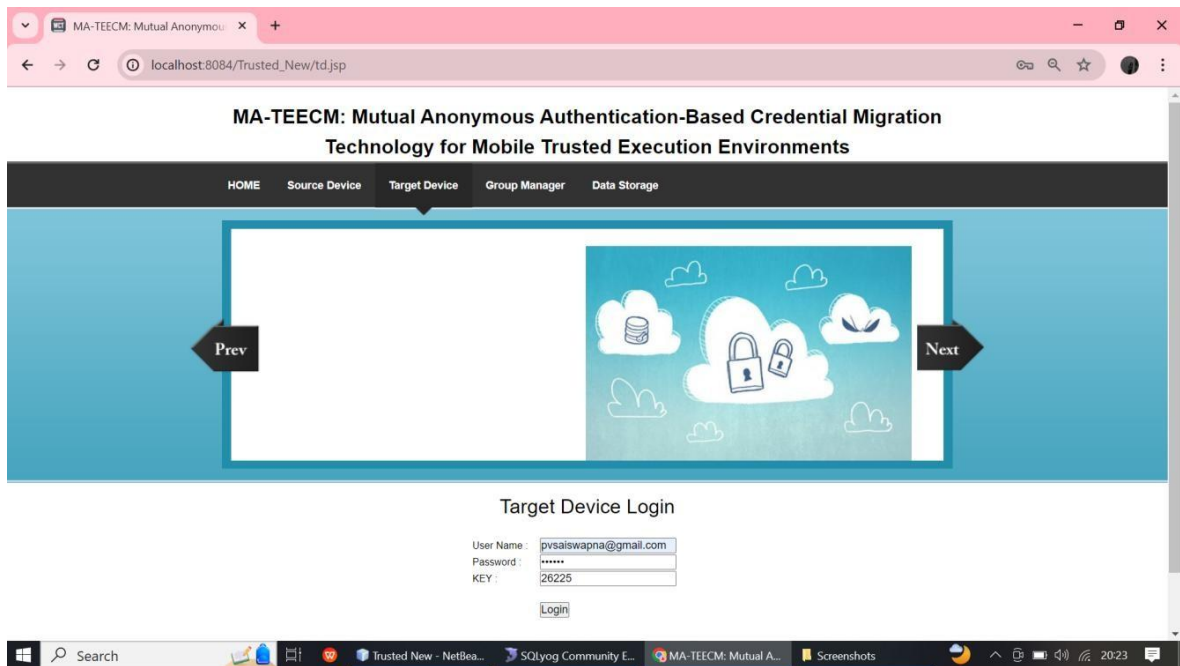
Screehshot 6.1.16: Group Manager Login



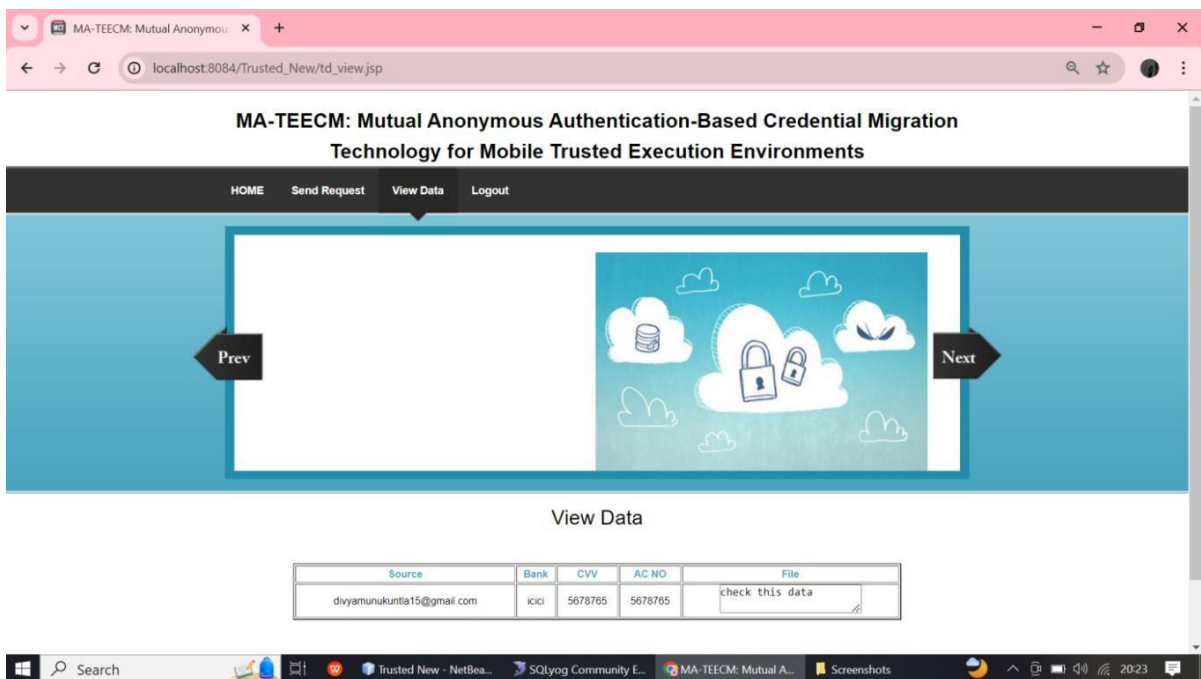
Screenshot 6.1.17: Activate the Target Device request



Screenshot 6.1.18: Delete the data from Source Device



Screenshot 6.1.19: Login Target Device



Screenshot 6.1.20: View data in Target Device

CHAPTER- 7

RESULTS

7.1 RESULTS

MA-TEECM: Mutual Anonymous Authentication-Based Credential Migration Technology for Mobile Trusted Execution Environments

HOME Send Request View Data Logout

Prev Next

View Data

Source	Bank	CVV	AC NO	File
divyamunukuntia15@gmail.com	icici	5678765	5678765	hello result of major

Figure 7.1.1: Output of MA-TEECM

CHAPTER- 8

CONCLUSION

Trusted Execution Environment is emerging as a flexible mobile security mechanism that can provide enhanced security guarantees for security-critical applications, credential files, and other types of sensitive data on any mobile device. This paper proposed a model framework that enables peer-to-peer credential migration between personal mobile devices to address credential availability issues caused by device lifecycle events. A third party, insulated from sensitive data, was introduced in the channel establishment process of credential migration, which is responsible for assisting two mobile devices in the local area network to establish group membership. Furthermore, a peer-to-peer credential migration protocol based on the mutual authentication scheme was designed, and the algorithm and model of credential migration in TEE were created. Security analysis showed that MA-TEECM could guarantee the confidentiality and integrity of credential data. Finally, AVISPA's back-end automated verification tools, OFMC and ATSE, were used to verify the security of the proposed protocol successfully.

CHAPTER - 9

FUTURE SCOPE

The current project has successfully implemented mutual anonymous authentication-based credentials migration technology for mobile trusted execution environments, enabling secure data uploading in document format. However, with the increasing demand for multimedia content, there is a growing need to expand the project's capabilities to support video and audio uploads.

Future Development Objectives:

1. Enhance the existing technology to support multimedia content uploads (video and audio) while maintaining the highest level of security and anonymity.
2. Develop a robust and efficient compression algorithm to optimize video and audio file sizes for seamless uploading and downloading.
3. Integrate advanced encryption techniques to ensure the confidentiality and integrity of multimedia content during transmission and storage.
4. Design and implement a scalable and fault-tolerant architecture to handle increased data volumes and user traffic.
5. Explore the integration of emerging technologies like blockchain and edge computing to further enhance security, privacy, and performance.
6. Conduct thorough security audits and penetration testing to identify and address potential vulnerabilities.
7. Collaborate with industry partners to ensure compatibility and interoperability with various mobile devices and platforms.

CHAPTER-10

REFERENCES

- [1] Hideo Nishimura, Yoshihiko Omori and Takao Yamashita, “Secure Authentication key sharing between personal mobile devices based on owner identity”, in proc. IEEE Truscom, vol 28 292-301 (apr.2020).
- [2] M. Sabt, M. Achemlal, and A. Bouabdallah, “Trusted execution environment: What it is, and what it is not,” in Proc. IEEE Trustcom/BigDataSE/ISPA, vol. 1, Aug. 2015, pp. 57–64.
- [3] L. Karlsson and M. Hell, “Enabling key migration between noncompatible TPM versions,” in Proc. Int. Conf. Trust Trustworthy Comput. Cham, Switzerland: Springer, 2016, pp. 101–118
- [4] K. Kostiainen, N. Asokan, and A. Afanasyeva, “Towards user-friendly credential transfer on open credential platforms,” in Proc. Int. Conf. Appl. Cryptogr. Netw. Secur. Berlin, Germany: Springer, 2011, pp. 395–412.
- [5] G. Arfaoui, S. Gharout, J.-F. Lalande, and J. Traoré, “Practical and privacy-preserving tee migration,” in Proc. IFIP Int. Conf. Inf. Secur. Theory Pract. Cham, Switzerland: Springer, 2015, pp. 153–168.
- [6] H. Li, Z. Li, Z. Wang, and X. Chang, “Authorization credential migration method, terminal device, and service server,” U.S. Patent 16 476 988, Nov. 21, 2019. [10] N. Kumar, “Identity authentication migration between different authentication systems,” U.S. Patent 10 412 077, Sep. 10, 2019.
- [7] C. Shepherd, R. N. Akram, and K. Markantonakis, “Remote credential management with mutual attestation for trusted execution environments,” in Proc. IFIP Int. Conf. Inf. Secur. Theory Pract. Cham, Switzerland: Springer, 2018, pp. 157–173.
- [8] T. Liang and S. Min, “TPM2.0 key migration-protocol based on duplication authority,” J. Softw., vol. 30, no. 8, pp. 2287–2313, 2019.
- [9] M. Song and L. Tan, “A TPM2. 0 key migration protocol and security analysis,” ACTA ELECTRONICA SINICA, vol. 47, no. 7, p. 1449, 2019.

- [10] H. Nishimura, Y. Omori, and T. Yamashita, “Secure authentication key sharing between personal mobile devices based on owner identity,” *J. Inf. Process.*, vol. 28, pp. 292–301, Apr. 2020.
- [15] J. Guerreiro, R. Moura, and J. N. Silva, “TEEnder: SGX enclave migration using HSMs,” *Comput. Secur.*, vol. 96, Sep. 2020, Art. no. 101874.
- [11] V. A. B. Pop, S. Virtanen, P. Sainio, and A. Niemi, “Secure migration of WebAssembly-based mobile agents between secure enclaves,” M.S. thesis, Univ. Turku, 2021.
- [12] J. Wang, P. Mahmood, F. Brasser, P. Jauernig, A.-R. Sadeghi, D. Yu, D. Pan, and Y. Zhang, “VirTEE: A full backward-compatible TEE with native live migration and secure I/O,” in *Proc. 59th ACM/IEEE Design Autom. Conf.*, Jul. 2022, pp. 241–246.
- [13] T. Hardjono and G. Kazmierczak. (2008). Overview of the TPM Key Management Standard. TCG Presentations. [Online]. Available: <https://www.trustedcomputinggroup.org/news>