

Data-Enhanced Authentication for Sensor Based Access Control System

Submitted in partial fulfillment of the requirements
for the award of

Bachelor of Technology degree in Information
Technology

by

Shanmugapriyan N (37120069)

Mohamed Mudassir S (37120045)



**DEPARTMENT OF INFORMATION TECHNOLOGY
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

Accredited with Grade "A" by NAAC

**JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI - 600 119**

MARCH – 2021



SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Accredited with "A" Grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai - 600 119.
Phone: 044 - 2450 3150 / 51 / 52 / 54 / 55 Fax: 044 - 2450 2344
www.sathyabama.ac.in



SCHOOL OF COMPUTING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of N. Shanmugapriyan (37120069) and S. Mohamed Mudassir (37120045) who carried out the project entitled "Data-Enhanced Authentication for Sensor Based Access Control System" under our supervision from November 2020 to March 2021.

Internal Guide

Mr.P. Saravanan, M.E.,

Head of the Department

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

Submitted for Viva voice Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I am N.Shanmugapriyan and S.Mohamed Mudassir hereby declare that the project report entitled on “Data-Enhanced Authentication for Sensor Based Access Control System” done by me under the guidance of Mr. P. Saravanan at Sathyabama Institute of Science And Technology, Semmancheri, Chennai – 600 119 and Mr. Sathish, Senior Embedded Engineer, LCS Controls PVT. LTD. is submitted in partial fulfillment of the requirements for the award of Bachelor of Technology degree in Information Technology.

DATE:

PLACE: Chennai

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

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

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

I would like to express my sincere and deep sense of gratitude to my **Project Guide Mr. P. Saravanan, M.E**, for his valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

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

ABSTRACT

Smartphones have become crucial for our daily life activities and are increasingly loaded with our personal information to perform several sensitive tasks including mobile banking, communication, and are used for storing private photos and files. Therefore, there is a high demand for applying usable authentication techniques that prevent unauthorized access to sensitive information. In this work, we propose AUToSen, a deep learning-based active authentication approach that exploits sensors in consumer-grade smartphones to authenticate a user. Unlike conventional approaches, AUToSen is based on deep learning to identify user distinct behavior from the embedded sensors with and without the user's interaction with the smartphone. We investigate different deep learning architectures in modelling and capturing users' behavioral patterns for the purpose of authentication. Moreover, we explore the sufficiency of sensory data required to accurately authenticate users. We evaluate AUToSen on a real-world dataset that includes sensors data of 84 participants' smartphones collected using our designed data collection application. The experiments show that AUToSen operates accurately using readings of only three sensors (accelerometer, gyroscope, and magnetometer) with a high authentication frequency, e.g., one authentication attempt every 0.5 seconds.

Using sensory data of one second enables an authentication F1- score of approximately 98%, FAR of 0.95%, FRR of 6.67%, and EER of 0.41%. While using sensory data of half a second enables an authentication F1-score of 97.52%, FAR of 0.96%, FRR of 8.08%, and EER of 0.09%. Moreover, we investigate the effects of using different sensory data at variable sampling periods on the performance of the authentication models under various settings and learning architectures.

TITLE OF CONTENT

CHAPTER NO.	TITLE	PAGE NO.
	ABSTRACT	v
	LIST OF FIGURES	ix
1	INTRODUCTION	1
	1.1 GENERAL	1
	1.2 AUTHENTICATION METHODS	1
2	LITERATURE SURVEY	3
	2.1 CONTROLLING USER ACCESS TO CLOUD-CONNECTED MOBILE APPLICATIONS BY MEANS OF BIO-METRICS	3
	2.2 DYNAMIC PASSWORD AUTHENTICATION PROTOCOL USING ANDROID DEVICE AND ONE-WAY FUNCTION	3
	2.3 SENSOR-BASED CONTINUOUS AUTHENTICATION OF SMARTPHONES	3
	2.4 USING DATA AUGMENTATION IN CONTINUOUS AUTHENTICATION ON SMARTPHONES	3
	2.5 A NEW FEATURE SCORING METHOD IN KEYSTROKE DYNAMICS-BASED USER AUTHENTICATIONS	3
	2.6 AN INTELLIGENT SCHEME FOR CONTINUOUS AUTHENTICATION OF SMARTPHONE USING DEEP AUTO ENCODER AND SOFT-MAX REGRESSION MODEL EASY FOR USER BRAIN	4
	2.7 INFERRING USER PROFILE ATTRIBUTES FROM MULTIDIMENSIONAL MOBILE PHONE SENSORY DATA	4
3	SYSTEM DESIGN	5
	3.1 STRUCTURE OF THE SYSTEM	5

	3.2 SYSTEM ARCHITECTURE	6
	3.3 MODULES	7
	3.3.1 ENROLLMENT PHASE	8
	3.3.2 ROTATION RECOGNITION	8
	3.3.2.1 DATA PRE-PROCESSING	9
	3.3.2.2 FEATURE VECTOR EXTRACTION	9
	3.3.2.3 F-VECTORS MATCHING	10
	3.3.3 SERVER VERIFICATION	11
	3.3.4 ACCESSING SERVICE	11
4	SYSTEM ANALYSIS	18
	4.1 EXISTING SYSTEM	18
	4.1.1 DISADVANTAGES	18
	4.2 PROPOSED SYSTEM	18
	4.2.1 ADVANTAGES	19
5	SOFTWARE DESCRIPTION	20
	5.1 SOFTWARE DESCRIPTION	20
	5.1.1 JAVA TECHNOLOGY	20
	5.1.1.1 THE JAVA PROGRAMMING LANGUAGE	20
	5.1.1.2 THE JAVA PLATFORM	21
	5.1.1.3 ACTIVITIES OF JAVA	22
	5.1.2 NETWORKING	23
	5.1.2.1 TCP/IP Stack	24

	5.1.3 IP DATAGRAM'S	24
6	RESULT AND DISCUSSION	36
7	CONCLUSION	37
8	REFERENCE	38
9	APENDEIX	39
	A . SOURCE CODE	39
	B . SCREENSHOTS	41
	C .PLAGARISM REPORT	45

LIST OF FIGURES

FIGURE NO.	FIGURE NAME	PAGE NO.
Fig. 3.1	SYSTEM ARCHITECTURE	7
Fig. 3.2	ENROLLMENT PHASE	8
Fig. 3.3	ROTATION RECOGNITION	10
Fig. 3.4	SERVER VERIFICATION	11
Fig. 3.5	ACCESSING SERVICE	11
Fig. 3.6	LEVEL 0 DATA FLOW DIAGRAM	12
Fig. 3.7	LEVEL 1 DATA FLOW DIAGRAM	13
Fig. 3.8	LEVEL 2 DATA FLOW DIAGRAM	13
Fig. 3.9	USE CASE DIAGRAM	14
Fig. 3.10	ACTIVITY DIAGRAM	15
Fig. 3.11	SEQUENCE DIAGRAM	16
Fig. 3.12	COLLABORATION DIAGRAM	17
Fig. 5.1	WORKING OF JAVA	21
Fig. 5.2	THE JAVA PLATFORM	22
Fig. 5.3	JAVA 2 SDK	23
Fig. 5.4	TCP/IP STACK	24
Fig. 5.5	ADDRESSING	25

CHAPTER 1

INTRODUCTION

GENERAL

Access control is a mechanism which enables an authority to control access to restricted areas and resources at a given physical facility or computer-based information system. In general, authentication methods in access control systems can be divided into two broad categories.

AUTHENTICATION METHODS

The first category is based on mechanical matching, such as keys and combination locks. Individuals are authenticated in these access control systems if and only if the blade of the key matches the keyway of the lock or the correct numerical sequence for combination lock has been dialed. Due to the physical constraints of mechanical matching systems, they are insufficient to meet the demanding requirements of access control authentication for critical infrastructures. On the other hand, it is also very hard to frequently change the interior structure of such matching mechanisms for security enhancement. The other category of authentication for access control systems is electronic authentication including barcode, magnetic stripe, biometrics and etc. Compared with mechanical matching authentications, the electronic authentications such as RFID-based smart card offer much more convenience and flexibility for both administrators and users of access control systems. However, it still suffers from similar problem of key loss since authentication is only based on the encoded identification data on the card. Anyone who carries the card will be granted the access and the security of the system still can be compromised. In order to further enhance the security of access control systems, various biometric authentication mechanisms have been introduced to identify the authorized personnel. Although these biometric authentication methods such as fingerprint, iris and voice recognitions are able to provide personal identification, they have high infrastructure cost and access privileges cannot be transferred among trusted users. In this work, we aim at bridging the gap between insufficiency of existing electronic authentication solutions and the increasing demand of high security guarantee for access control systems. We design a novel electronic proximity authentication framework that enhances the security level of

existing RFID-based access control systems with backward compatibility. Specifically, we add dynamic data into the traditional authentication information by using sensors such as accelerometer, gyroscope and etc. In summary, on top of the previous conference paper, our contributions in this work are as follows:

- We design and implement a dynamic authentication framework with sensory information for the access control systems.
- We demonstrate the proposed framework with two Digital Object case studies and theoretically prove that our dynamic authentication significantly increases the key space for proximity authentication systems with the integration of low-cost sensors.
- We have fully implemented and built a running prototype of the proposed dynamic authentication framework on the Intel Wireless Identification and Sensing Platform (WISP). Based on the running prototype, we have extensively evaluated our design in terms of system accuracy and usability in the real-world settings.

CHAPTER 2

LITERATURE SURVEY

CONTROLLING USER ACCESS TO CLOUD-CONNECTED MOBILE APPLICATIONS BY MEANS OF BIOMETRICS

A continuous-authentication approach integrating physical (face) and behavioral (touch and hand movements) biometrics to control user access to cloud-based mobile services, going beyond one-time login. Experimental results show the security-usability tradeoff achieved by the approach.

DYNAMIC PASSWORD AUTHENTICATION PROTOCOL USING ANDROID DEVICE AND ONE-WAY FUNCTION

An authentication scheme in which using an android phone we can tackle any such attacks and provide a secure authentication over insecure and low-security networks.

SENSOR-BASED CONTINUOUS AUTHENTICATION OF SMARTPHONES

User authentication is a task of paramount importance that grants access to legitimate users at the point of entry and continuously through the usage session. This task is made possible with today's smartphones' embedded sensors that enable continuous and implicit user authentication by capturing behavioral biometrics and traits.

USING DATA AUGMENTATION IN CONTINUOUS AUTHENTICATION ON SMARTPHONES

A novel authentication system, SensorAuth, for continuous authentication of users based on their behavioral patterns, by leveraging the accelerometer and gyroscope ubiquitously built into smartphones.

A NEW FEATURE SCORING METHOD IN KEYSTROKE DYNAMICS-BASED USER AUTHENTIFICATIONS

A novel filter-based feature-selection method in the context of keystroke dynamics authentication. In particular, here propose a new feature-scoring method and apply it to keystroke-dynamics-based authentications. Here implement keystroke-dynamics-

based authentications multi-factored with PIN-based authentications and collect data from actual users' testing experiments.

AN INTELLIGENT SCHEME FOR CONTINUOUS AUTHENTICATION OF SMARTPHONE USING DEEP AUTO ENCODER AND SOFTMAX REGRESSIONMODEL EASY FOR USER BRAIN

A continuous authentication technique is designed with the Deep Auto Encoder and Softmax Regression techniques (DAE-SR). The DAE-SR achieved 0.950% and 0.970% of accuracy on predicting the users on different states (walking and sitting) of users.

INFERRING USER PROFILE ATTRIBUTES FROM MULTIDIMENSIONAL MOBILE PHONE SENSORY DATA

Here it is proposed to infer user profile, including age, gender, and personality traits based on mobile phone sensory data. Specifically, here capture data when unlocking screen, playing games as well as some basic mobile phone information, app usage, and screen status by using common available sensors in commodity mobile phones. By analyzing the differences in users' phone usage, we extracted features for user profile inference.

CHAPTER 3

SYSTEM DESIGN

STRUCTURE OF THE SYSTEM

The existing electronic proximity authentication of access control systems is mainly based on the exchange of encoded identification information stored on the access card. The security and integrity of such static and passive authentication mechanisms suffer from problems such as access card loss and unauthorized duplications.

In this work, we propose to use sensory information obtained from wireless rechargeable sensors on access cards to further enhance the security and robustness of existing electronic proximity authentication systems. The main idea of our system design when an access card integrated with wireless rechargeable sensors enters the communication range of an access control client; the access card piggybacks its sensory data to conventional identification information and transmits it (i.e. the electronic key) to the access control client. The information received by the access control client is then forwarded to the network server for authentication. If both sensory data and identification match a valid record in the authentication database, the network server then instruments the actuator and grants the card holder the access to the system. In this way, even an authentic access card is in possession of a unauthorized personnel or has been illegally duplicated, as long as the unauthorized card holder does not know how to generate the correct sensory data, he or she still cannot access the system. Moreover, we successfully remove the system vulnerable period between loss/stolen of access card and the deactivation of the card after users' report.

On the contrary, trusted users can share the cards and predefined actions with each other which is unavailable in biometric authentication systems. Different from existing authentication methods such as combining RFID and an additional keypad near the reader, we propose an orthogonal design in this paper and the new authentication framework only revises authentication algorithm on the network server without any modification of access clients. In fact, since we piggyback sensory data to ID information before transmitting them to the reader, most existing works on communication encryption for RFID system can be easily adopted into our

authentication method and therefore deal with several security vulnerabilities such as replay attack and eavesdropping. All the identification information on access cards normally is static.

With the addition of dynamic sensory data from onboard sensors, we are able to significantly increase the security key space P and hence the level of security for existing electronic authentication systems. A wide variety of sensors including accelerometer, gyroscope and etc. can be used in our system. To illustrate the basic concept and the resulting security enhancement of our sensory data enhanced access control system design, we use both three-axis accelerometer and gyroscope as examples in the following sections. In particular, we utilize the sensory data generated from the rotation of accelerometer and gyroscope to introduce a reference design for the proposed sensory data enhanced authentication scheme. Through our prototyping system and real world experiments, we demonstrate such a rotation-based design is a feasible and practical option for the proposed generic dynamic authentication framework.

SYSTEM ARCHITECTURE

Here, the user uses the accelerometer sensor in order to obtain values. The accelerometer sensor works based on 3 mechanisms: Data Pre-Processing, Feature Vector Extraction and F-Vectors Matching. From the obtained values a password is generated and is stored in the database by the server and is also sent to the user who wants to access it. The user tries to login using the password received. If the password matches with the one in the database then the user is allowed to access the site. If the password does not match then the user is denied access. This gives a dynamic authentication method. Every time a dynamic password is generated. The followin figure 5.1 shows the system architecture diagram.

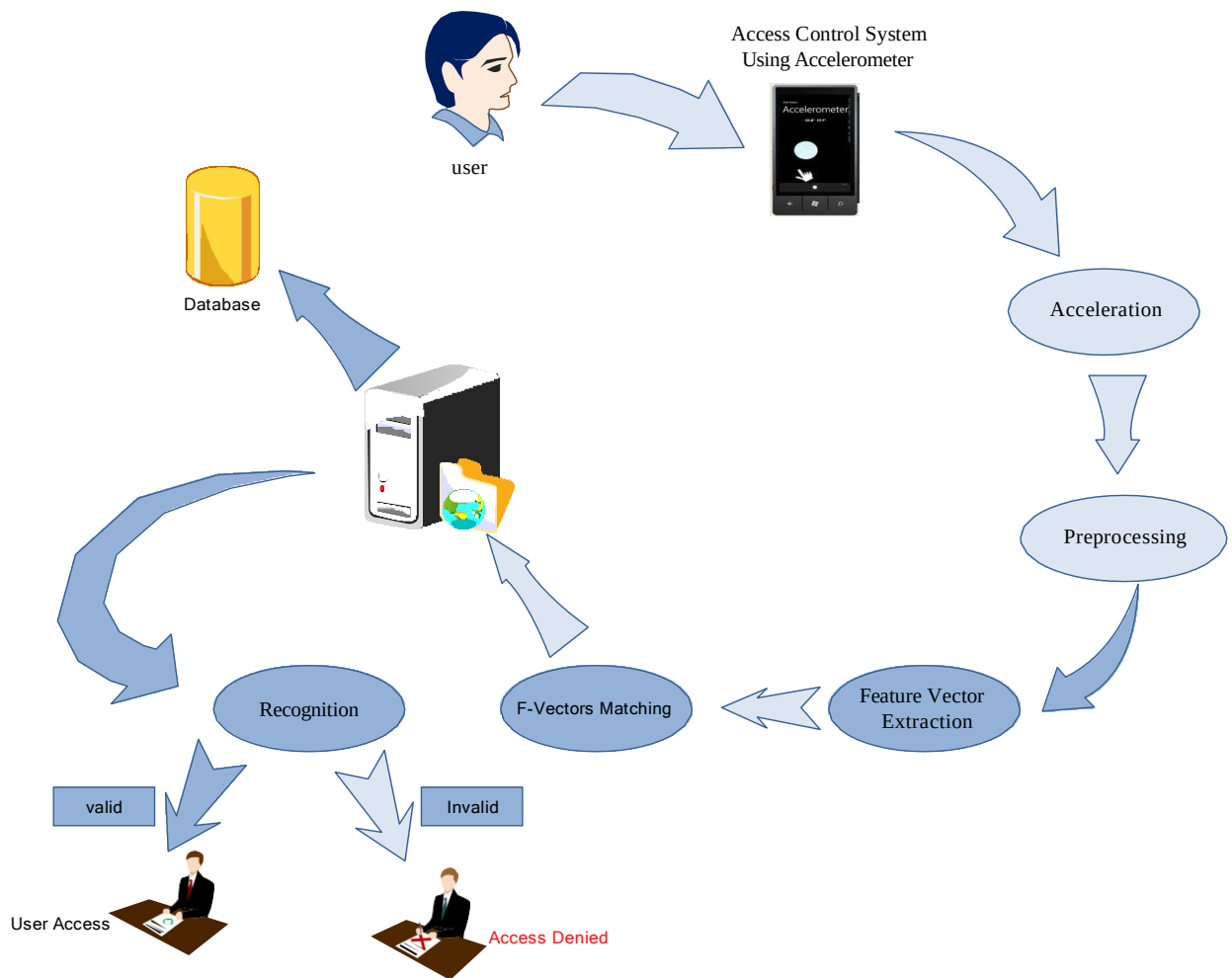


Fig 3.1 System Architecture

MODULES

- Enrollment Phase
- Rotation Recognition
 - ❖ Data Pre-Processing
 - ❖ Feature Vector Extraction
 - ❖ F-Vectors Matching
- Server Verification
- Accessing Service

Enrollment Phase

The aim of this phase is to allow a user and a server to negotiate a shared secret to authenticate succeeding logins for this user. The user begins using Accelerometer sensor for the Access Control Systems. The following figure 3.2 shows the enrollment phase.

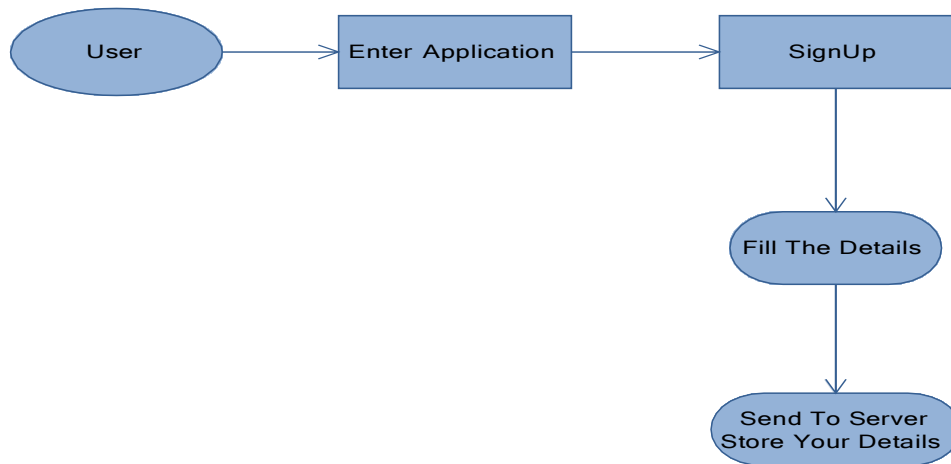


Fig 3.2 Enrollment Phase

Rotation Recognition

Dynamic authentication with sensory information design. In this section, we further elaborate on the detailed sensor rotation recognition algorithms. By comparing the sample data of accelerometer. We find that output of the accelerometer exhibits a more complex behavior. This is because gyroscope measures the angular velocity and tends to generate an impulse during one single basic rotation, which could be treated as a special case of the output of the accelerometer. Therefore in this section, we use the sensory data of accelerometer to illustrate the whole rotation recognition algorithms and discuss how to deal with the sensory data of gyroscope. One complete dynamic authentication process consists of a sequence of basic rotations. In order to accurately identify each individual basic rotation from raw accelerometer data, we perform following three operations in the network server.

Data Pre-Processing

The first step of rotation recognition is data pre-processing. The main goals are to separate and filter each individual basic rotation from a series of raw accelerometer data. In order to separate the individual basic rotations, we first need to identify the pause between two consecutive rotations. During such pauses, the three-axis readings of an accelerometer would remain relatively stable and unchanged for a short period of time. In order to accurately recognize such pauses and separate different basic rotations, we adopt a sliding window approach. In this approach, the accelerometer readings in the first tw second are buffered into the sliding window. All data in the sliding window are then fitted by a first-order polynomial function. If the coefficient of first-order polynomial is less than a threshold (1 in our implementation), we consider the accelerometer remain stationary within the time frame of this window. Followed by this pause detection in the current window, the window would slide for a step of ts seconds, with ts duration of new data appended to the end of the sliding window while the first ts duration of sensory data are discarded. Empirically, we set $tw = 1s$ and $ts = 0.3s$ in our system implementation. In this way, we have achieved accurate separation of basic rotations in one complete authentication. To visualize above data pre-processing step, one authentication with 4 basic rotations that performed slowly on our prototype implementation. The shaded regions represent sliding windows at three pauses. The accelerations on three axes of the accelerometer are rather stable during pauses between different basic rotations.

After identifying pauses between basic rotations, we then use least square estimation to fit the raw readings for each individual basic rotation from the accelerometer.

Feature Vector Extraction

After separating basic rotations for one single authentication, we match them with standard feature vectors. As feature based classification of time-series data has a simple model and lower computation, we choose this method for rotation recognitions. First, feature vectors (F-Vectors) for each individual basic rotation are extracted based on their fitting functions created in the previous section. Specifically, we extract the start and end sensory data, the maximal and minimal sensor readings

and the corresponding time of these events within one basic rotation for a three-axis accelerometer.

A sufficiently large feature vector for use in the authentication protocol. In our approach, the feature vector will be used to authenticate a key or to directly generate a key, and thus it needs to be of high entropy from an attacker's point of view, i.e. involve a large amount of uncertainty. We argue that shaking is an appropriate movement for creating entropy: it creates varying sensor readings, because it is one of the human movement patterns that includes the highest frequency components. Slower movements will intuitively not generate as much entropy.

F-Vectors Matching

After extracting feature vectors, we then try to match the extracted feature vector with standard feature vectors in the database to recognize a specific basic rotation. Standard feature vectors with given n could be mathematically calculated and automatically generated since the acceleration components on three axes represent a trigonometric relationship with acceleration of gravity. Taking the rotation as an example, after the accelerometer clockwise rotates π degrees, the acceleration components A_x and A_y during such rotation can be calculated as $A_x = G\cos\theta$ and $A_y = G\sin\theta$ ($\theta \in [\alpha, \alpha+\pi]$). Therefore, it is easy for users to reset their keys without any modification on access cards. In order to match extracted F-vectors of a basic rotation to standard ones in database, we use Euclidean distance to measure the closeness of these two vectors. The figure 3.3 shows Rotation Recognition phase.

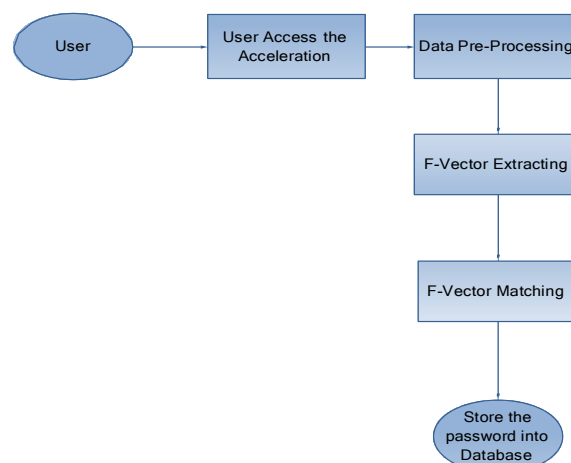


Fig 3.3 Rotation Recognition

Server Verification

Server can verify the authenticity of the registration details and then obtain with the key. Server also compares the source of received key. Then server verifies both password, the password match or not. The following figure 3.4 illustrates the Server Verification phase.



Fig 3.4 Server Verification

Accessing Service

User enter the browser and Register to server then server through mail on password then user receive the mail and send to server .then server verify both password ,if correct the password open the view all detail ,else if not match that password means you won't allow the site inside. The following figure 3.5 shows the accessing service module.

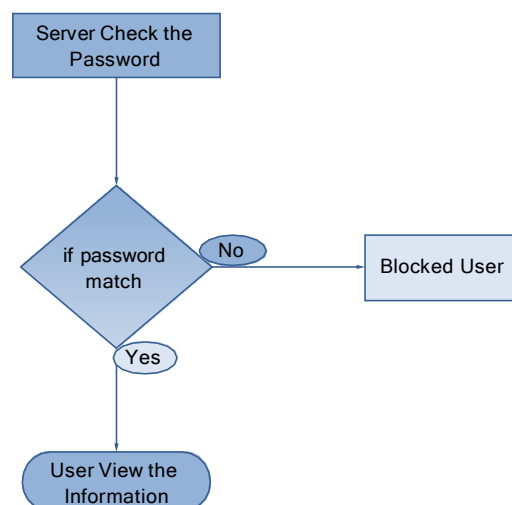


Fig 3. 5 Accessing Service

DATA FLOW DIAGRAM

A data-flow diagram (DFD) is a graphical representation of the "flow" of data through an information system. DFDs can also be used for the visualization of data processing (structured design). On a DFD, data items flow from an external data source or an internal data store to an internal data store or an external data sink, via an internal process.

A DFD provides no information about the timing or ordering of processes, or about whether processes will operate in sequence or in parallel. It is therefore quite different from a flowchart, which shows the flow of control through an algorithm, allowing a reader to determine what operations will be performed, in what order, and under what circumstances, but not what kinds of data will be input to and output from the system, nor where the data will come from and go to, nor where the data will be stored (all of which are shown on a DFD).

a) Level 0

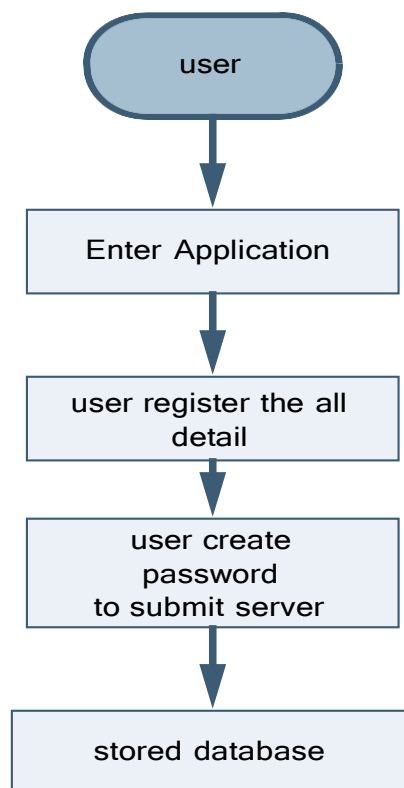


Fig 3.6 Level 0 Data Flow Diagram

b) Level 1

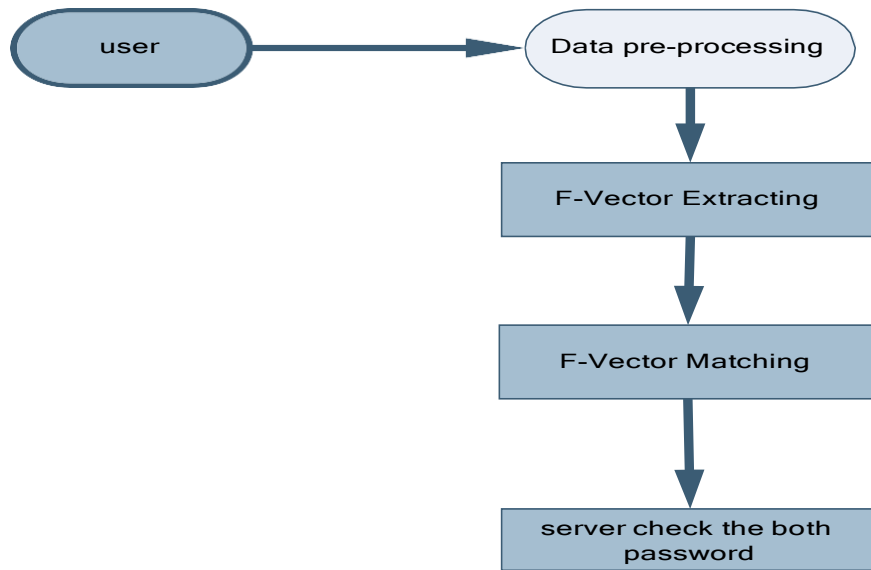


Fig 3.7 Level 1 Data Flow Diagram

c) Level 2

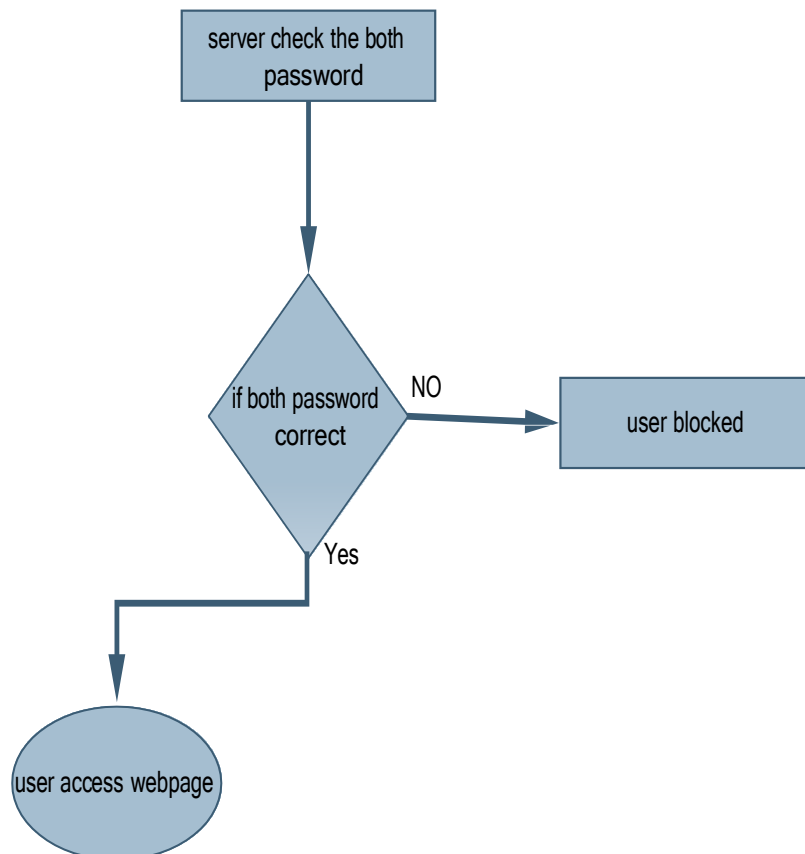


Fig 3.8 Level 2 Data Flow Diagram

UML DIAGRAM

Use Case 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. The following figure 3.9 shows the use case diagram of the overall working system.

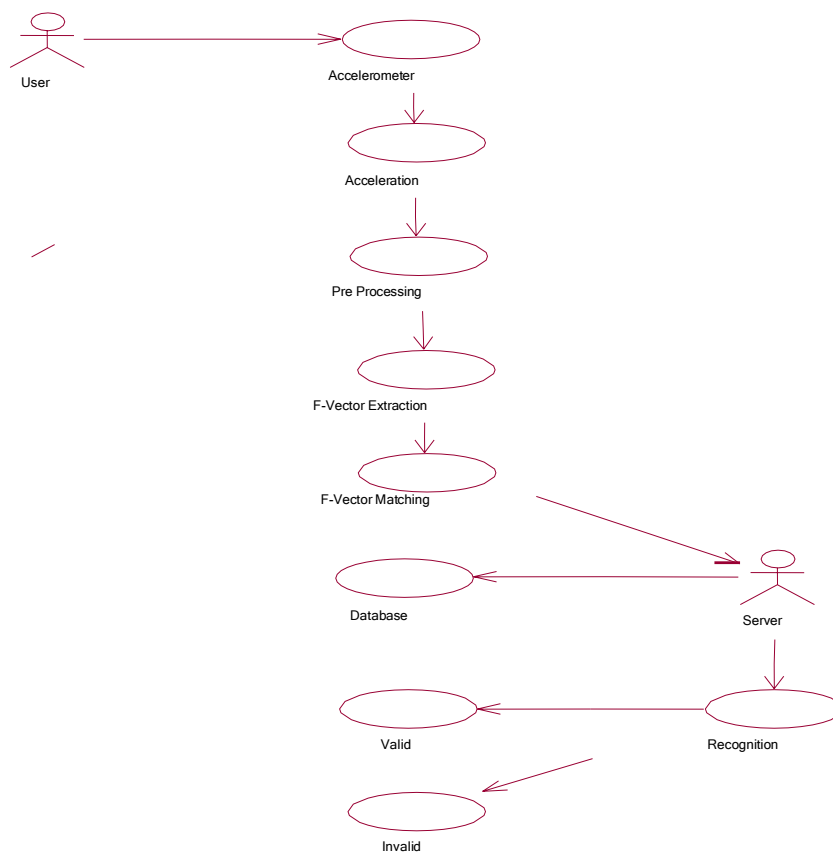


Fig 3.9 Use Case Diagram

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 workflow of components in a system. An activity diagram in figure 3.10 shows the overall flow of control.

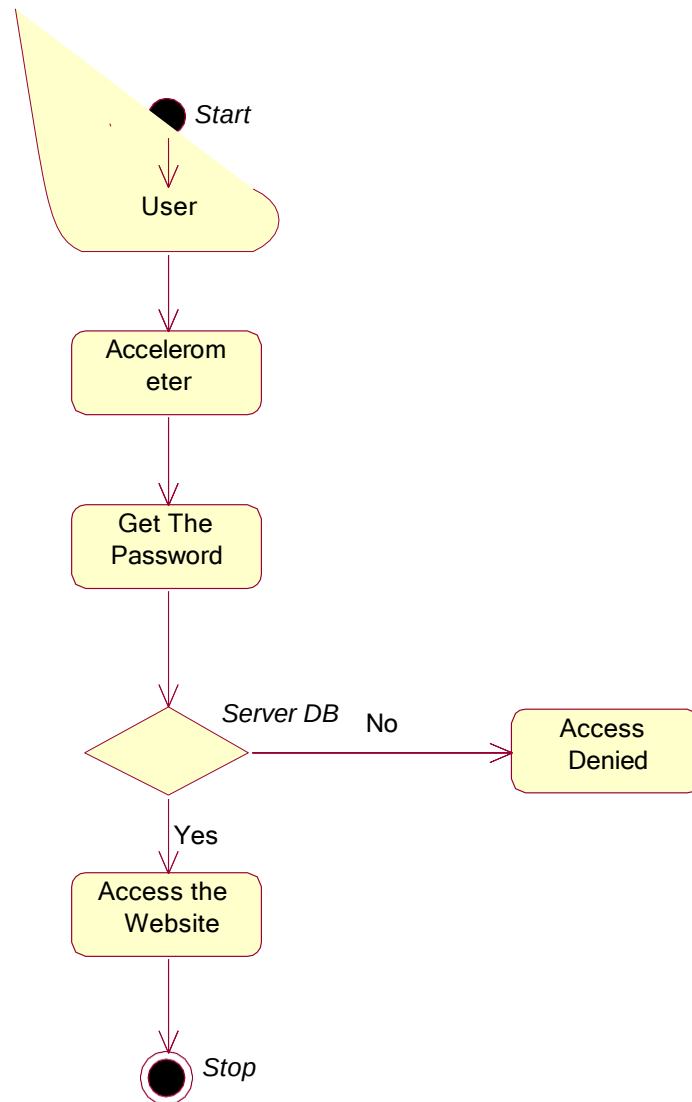


Fig 3.10 Activity Diagram

Sequence Diagram

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-trace diagrams, event scenarios, and timing diagrams.

A sequence diagram shows, as parallel vertical lines (*lifelines*), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. This allows

the specification of simple runtime scenarios in a graphical manner. The following figure 3.11 illustrates the overall working of the sensor.

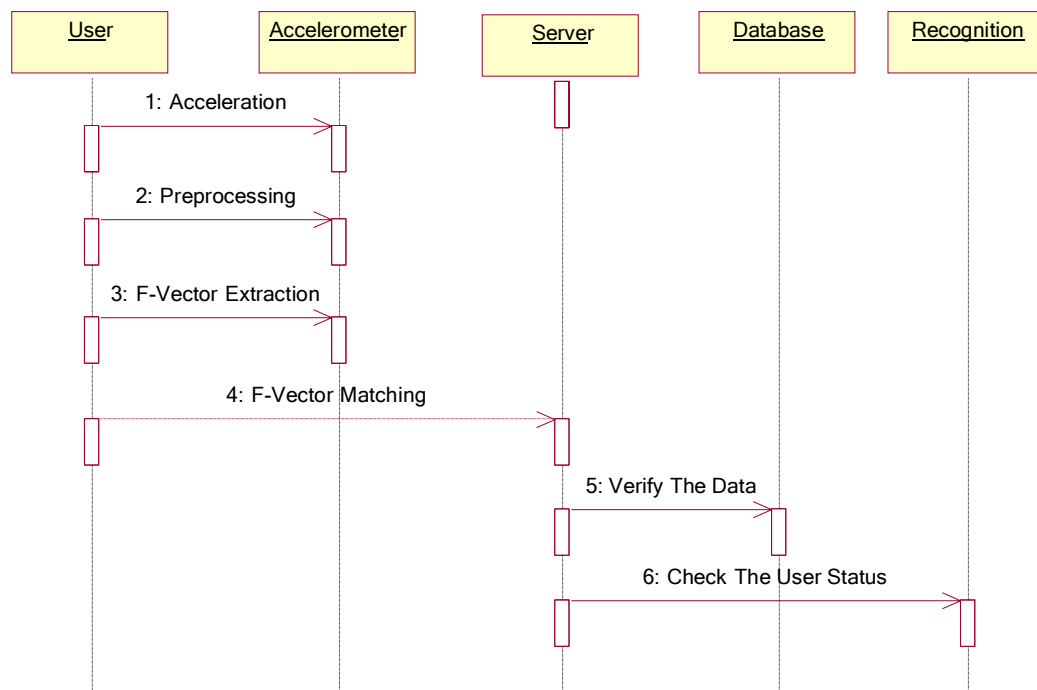


Fig 3.11 Sequence Diagram

Collaboration Diagram

Collaboration diagrams belong to a group of UML diagrams called Interaction Diagrams. Collaboration diagrams, like Sequence Diagrams, show how objects interact over the course of time. However, instead of showing the sequence of events by the layout on the diagram, collaboration diagrams show the sequence by numbering the messages on the diagram. This makes it easier to show how the objects are linked together, but harder to see the sequence at a glance. The following figure 3.12 gives the collaboration diagram of the entire working of the system.

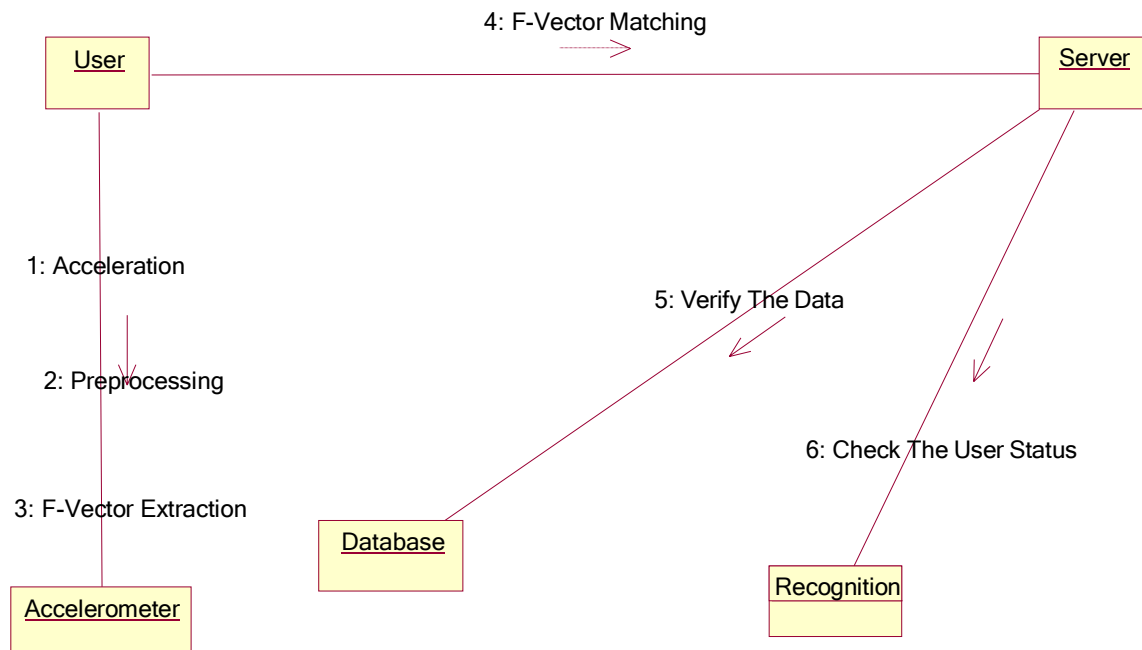


Fig 3.12 Collaboration Diagram

CHAPTER 4

SYSTEM ANALYSIS

EXISTING SYSTEM

- Access card authentication is critical and essential for many modern access control systems, which have been widely deployed in various governments, commercial and residential environments.
- Access control systems can be divided into two broad categories. The first category is based on mechanical matching, such as keys and combination locks.
- Access control systems if and only if the blade of the key matches the keyway of the lock or the correct numerical sequence for combination lock has been dialed.
- Electronic authentication including barcode, magnetic stripe, biometrics and etc.
- Although advanced biometric authentication methods such as fingerprint and iris identification can further identify the user who is requesting authorization, they incur high system costs and access privileges cannot be transferred among trusted users.
- A novel electronic proximity authentication framework that enhances the security level of existing RFID-based access control systems with backward compatibility.

Disadvantages

- Stolen or unauthorized duplications of the access cards.
- Sensors on the access cards effectively tackle the problems such as access card loss, stolen and duplication.
- High system costs and access privileges cannot be transferred among trusted users.

PROPOSED SYSTEM

- Dynamic data into the traditional authentication information by using sensors such as accelerometer, gyroscope and etc.

- This authentication framework is adaptive to the change of encryption complexity of the access control systems and could be adopted with minor modification of existing infrastructure.
- Dynamic authentication significantly increases the key space for proximity authentication systems with the integration of low-cost sensors.
- We have fully implemented and built a running prototype of the proposed dynamic authentication framework.
- The Intel Wireless Identification and Sensing Platform (WISP). Based on the running prototype, we have extensively evaluated our design in terms of system accuracy and usability in the real-world settings.
- The identification information on access cards normally are static.
- The addition of dynamic sensory data from onboard sensors.
- Significantly increase the security key space P and hence the level of security for existing electronic authentication systems. A wide variety of sensors including accelerometer, gyroscope and etc. Can be used in our system.

Advantages

- Secure Registration and Recovery.
- Backward-compatible with existing access control systems and significantly increases the key spaces for authentication.
- The potential key space increases with sensory information.
- Key space by more than 1, 000, 000.
- Authentication accuracy of 90%.
- Implemented our design on wisp to experimentally verify the system performance.

CHAPTER 5

SOFTWARE DESCRIPTION

SOFTWARE DESCRIPTION

Java Technology

Java technology is both a programming language and a Platform.

The Java Programming Language

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

- Simple
- 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 5.1 illustrates how this works.

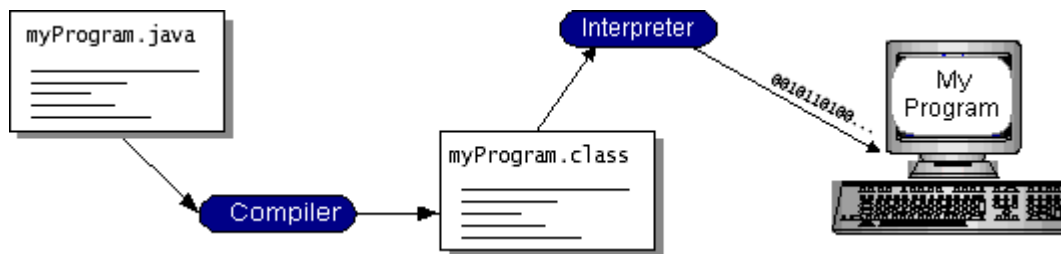


Fig 5.1- Working Of JAVA

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.

The Java Platform

A platform is the hardware or software environment in which a program runs. 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 following figure 5.2 depicts a program that's running on the Java platform. As the figure 5.2 shows, the Java API and the virtual machine insulate the program from the hardware.

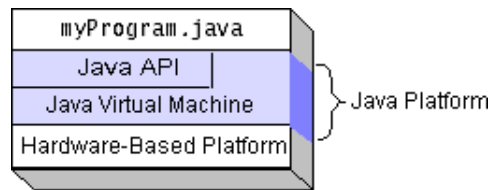


Fig 5.2 The JAVA Platform

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

Activities of Java

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.
- **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 5.3 depicts what is included in the Java 2 SDK.

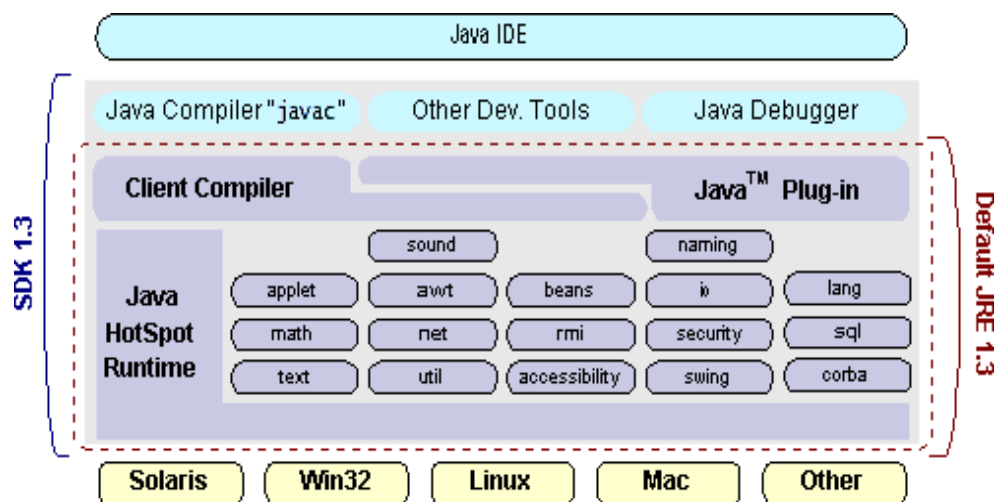


Fig 5.3 – JAVA 2 SDK

Networking

TCP/IP stack

The TCP/IP stack is shorter than the OSI one: The following figure 4.4 shows the TCP/IP Stack Structure.

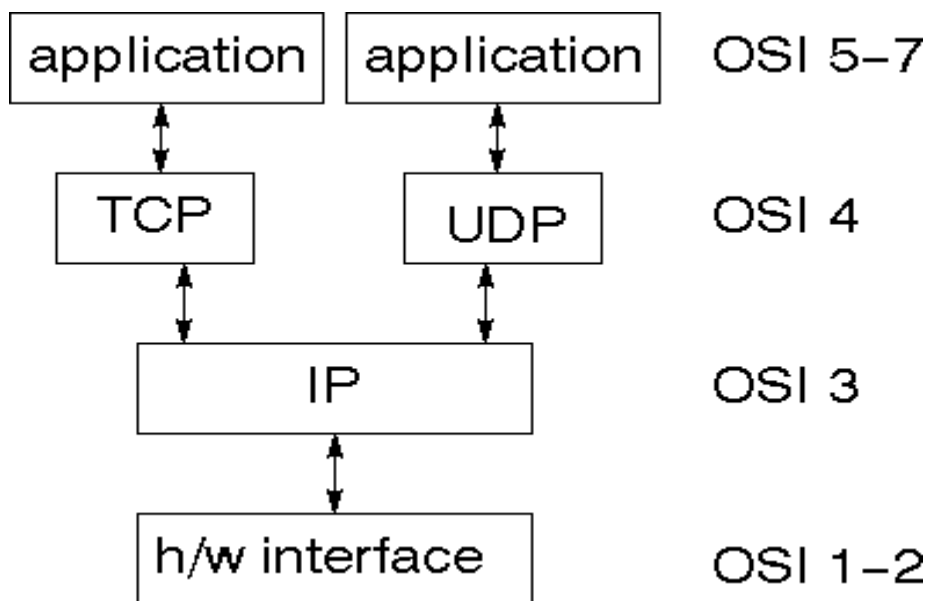


Fig 5.4 – TCP/IP Stack

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.

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

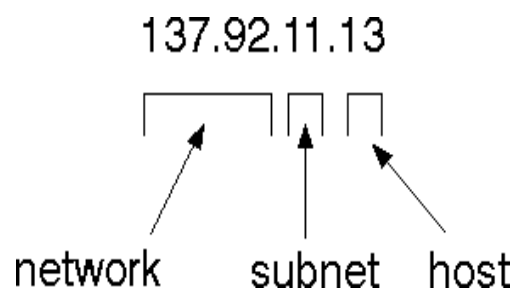


Fig 5.5- IP Addressing

The 32 bit address is usually written as 4 integers separated by dots. The above figure 5.5 indicates the IP Addressing.

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.

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.

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 errors appear at runtime. So these kind of techniques

7. Keep the common cases simple

Because more often than not, the usual SQL calls used by the programmer are simple SELECT’s, INSERT’s, DELETE’s and UPDATE’s, these queries should be simple to perform with JDBC. However, more complex SQL statements should also be possible.

IMPLEMENTATION

System Implementation is the stage in the project where the theoretical design is turned into a working system. The most critical stage is achieving a successful system and in giving confidence on the new system for the user that it will work efficiently and effectively.

The existing system was long time process. The proposed system was developed using Java Swing. The existing system caused long time transmission process but the system developed now has a very good user-friendly tool, which has a menu-based interface, graphical interface for the end user.

After coding and testing, the project is to be installed on the necessary system. The executable file is to be created and loaded in the system. Again the code is tested in the installed system. Installing the developed code in system in the form of executable file is implementation.

USER DOCUMENTATION

Getting started

If you do not have Java installed already, follow the instructions provided with the download in order to install Java.

Once Java has been installed, you will be able to run Java program by running the java executable that was installed. You may find it convenient to add the location of this file to your operating system's path variable, otherwise you will need to explicitly refer to the absolute location of java.

Setting up the path

Windows 2000/XP users may set their path by right-clicking on 'My Computer' and selecting 'Properties'. Under the 'Advanced' tab, there is a button that allows you to set the 'Environment variables'. Click on this and alter the 'Path' variable so that it also contains the path to the Java executable. For example, if you have installed Java in c:\jdk and your path is currently set to C:\WINDOWS\SYSTEM32, then you would change your path to read C:\WINDOWS\SYSTEM32;c:\jdk\bin

When you open a *new* command prompt, it will reflect these changes and allow you to run java programs by typing "java". If you have installed the SDK, then you will also be able to run "javac" to compile stuff.

Windows 95/98/ME users may find that their path variable is stored in a different place. Edit the c:\autoexec.bat file and add the following line at the end: SET PATH = % PATH % ; c : \jdk \ bin

(This also assumes that you have installed Java in c:\jdk)

In addition to setting up the path, you also need to tell Java where to find compiled class files at runtime. You will probably want to set the classpath to include at least the current working directory (.)

Eg: SET CLASSPATH=%CLASSPATH%;

The classpath can also contain other directories that may contain compiled class files. Note that if you are using classes that are contained inside a .jar file, you will need to specify the full path and name of this file in the classpath, rather than just the name of the directory it is contained within.

INSTALL AND UNINSTALL

The MySQL Installation Wizard

An excellent .msi installation wizard is available for MySQL. The wizard creates the my.ini configuration file and installs MySQL as a service. This option is certainly recommended for novice users or perhaps those installing MySQL for the first time.

Manual Installation

Manual installation offers several benefits:

- Backing up, reinstalling, or moving databases can be achieved in seconds (see 8 Tips for Surviving PC Failure).
- You have more control over how and when MySQL starts.
- You can install MySQL anywhere, such as a portable USB drive (useful for client demonstrations).

Step 1: Download MySQL

Download MySQL from dev.mysql.com/downloads/. Follow *MySQL Community Server, Windows* and download the “Without installer” version. As always, virus scan the file and check the its MD5 checksum using a tool.

Step 2: Extract the files.

We will install MySQL to C:\mysql, so extract the ZIP to your C: drive and rename the folder from “mysql-x.x.xx-win32” to “mysql”. MySQL can be installed anywhere on your system. If you want a lightweight installation, you can remove every sub-folder except for bin, data, scripts and share.

Step 3: Move the data folder.

For the purposes of this example, we will create a folder called D:\MySQLdata and move the contents of C:\mysql\data into it. You should now have two folders, D:\MySQLdata\mysql and D:\MySQLdata\test. The original C:\mysql\data folder can be removed.

Step 4: Create a configuration file.

MySQL provides several configuration methods but, in general, it is easiest to create a my.ini file in the mysql folder. There are hundreds of options to tweak MySQL to your exact requirements, but the simplest my.ini file is:

1. [mysqld]
2. # installation directory
3. basedir="C:/mysql/"
4. # data directory
5. datadir="D:/MySQLdata/"

(Remember to change these folder locations if you have installed MySQL or the data folder elsewhere.)

Step 5: Test your installation.

The MySQL server is started by running C:\mysql\bin\mysqld.exe. Open a command box (Start > Run > cmd) and enter the following commands:

1. cd \mysql\bin
2. mysqld

This will start the MySQL server which listens for requests on localhost port 3306. You can now start the MySQL command line tool and connect to the database. Open another command box and enter:

1. cd \mysql\bin
2. mysql -u root

This will show a welcome message and the `mysql>` prompt. Enter “show databases;” to view a list of the pre-defined databases.

Step 6: Change the root password.

The MySQL root user is an all-powerful account that can create and destroy databases. If you are on a shared network, it is advisable to change the default (blank) password. From the `mysql>` prompt, enter:

1. `UPDATE mysql.user SET password=PASSWORD("my-new-password") WHERE User='root';`
2. `FLUSH PRIVILEGES;`

You will be prompted for the password the next time you start the MySQL command line.

Enter “exit” at the `mysql>` prompt to stop the command line client. You should now shut down MySQL with the following command:

1. `mysqladmin.exe -u root shutdown`

Step 7: Install MySQL as a Windows service.

The easiest way to start MySQL is to add it as a Windows service. From a command prompt, enter:

1. `cd \mysql\bin`
2. `mysqld --install`

Open the Control Panel, Administrative Tools, then Services and double-click MySQL. Set the Startup type to “Automatic” to ensure MySQL starts every time you boot your PC.

Alternatively, set the Startup type to “Manual” and launch MySQL whenever you choose using the command “`net start mysql`”.

SOFTWARE USAGE

The procedure for using FEDERATED tables is very simple. Normally, you have two servers running, either both on the same host or on different hosts. (It is possible for a FEDERATED table to use another table that is managed by the same server, although there is little point in doing so.)

First, you must have a table on the remote server that you want to access by using a FEDERATED table. Suppose that the remote table is in the federated database and is defined like this:

```
CREATE TABLE test_table (  
  id INT(20) NOT NULL AUTO_INCREMENT,
```

```

name VARCHAR(32) NOT NULL DEFAULT "",
other INT(20) NOT NULL DEFAULT '0',
PRIMARY KEY (id),
INDEX name (name),
INDEX other_key (other)
)
ENGINE=MyISAM
DEFAULT CHARSET=latin1;

```

The example uses a MyISAM table, but the table could use any storage engine. Next, create a FEDERATED table on the local server for accessing the remote table.

```

CREATE TABLE federated_table (
id INT(20) NOT NULL AUTO_INCREMENT,
name VARCHAR(32) NOT NULL DEFAULT "",
other INT(20) NOT NULL DEFAULT '0',
PRIMARY KEY (id),
INDEX name (name),
INDEX other_key (other)
)
ENGINE=FEDERATED
DEFAULT CHARSET=latin1
CONNECTION='mysql://fed_user@remote_host:9306/federated/t
est_table';

```

(Before MySQL 5.0.13, use COMMENT rather than CONNECTION.)

The basic structure of this table should match that of the remote table, except that the ENGINE table option should be FEDERATED and then the CONNECTION table option is a connection string that indicates that to the FEDERATED engine how to connect to the remote server.

Note

You can improve the performance of a FEDERATED table by adding indexes to the table on the host. The optimization will occur because the query sent to the remote server will include the contents of the WHERE clause and will be sent to the remote server and subsequently executed locally. This reduces the network traffic that would otherwise request the entire table from the server for

local processing.

The FEDERATED engine creates only the test_table.frm file in the federated database.

The remote host information indicates the remote server to which your local server connects, and the database and table information indicates which remote table to use as the data source. In this example, the remote server is indicated to be running as remote_host on port 9306, so there must be a MySQL server running on the remote host and listening to port 9306.

The general format of the connection string in the CONNECTION option is as follows:

scheme://user_name[:password]@host_name[:port_num]/db_name/tbl_name

Only mysql is supported as the *scheme* value at this point; the password and port number are optional.

Sample connection strings:

```
CONNECTION='mysql://username:password@hostname:port/database/tablename'
```

```
CONNECTION='mysql://username@hostname/database/tablename'
```

```
CONNECTION='mysql://username:password@hostname/database/tablename'
```

The use of CONNECTION for specifying the connection string is non optimal and is likely to change in future. Keep this in mind for applications that use FEDERATED tables. Such applications are likely to need modification if the format for specifying connection information changes.

Because any password given in the connection string is stored as plain text, it can be seen by any user who can use SHOW CREATE TABLE or SHOW TABLE STATUS for the FEDERATED table, or query the TABLES table in the INFORMATION_SCHEMA database.

CHAPTER 6

RESULTS AND DISCUSSION

Thus our deep learning based authentication system is able to give more security than traditional authentication system because of its Three-Way authenticating policy. At first we need to register and once the registration is done we should login with the password we set during registration which can be valid only first five times. Then it will send grid password to the user mail id and it will direct user to the page where the user should enter the grid password not directly but split those into three. Then in our android application user should login and should start to rotate the mobile to generate the values and one of those generated values will be send to user mail id and the user has to enter it to access the information in the application. The reason we are saying that our authenticating method is when we compare it with other authentication methods like Bio-Metric or Card type authenticating systems. The main drawback in traditional OTP is it will be only integer value whereas our project will give values with different data types. Drawbacks in Bio-Metric or Card type authentication is that someone who is not the owner can access the information. This project is not just about giving security, but also it is about less cost because we do not need any hardware explicitly like Bio-Metric and other authentication systems and we can use our in-built sensors for information access.

CHAPTER 6

CONCLUSION

In this paper, we propose a dynamic authentication with sensory information for the access control systems. Different from existing schemes of authentication in access control systems, which mainly based on static information on cards, our dynamic authentication method combines sensory information from onboard sensors and conventional static ID information. Two case studies of the dynamic authentication are proposed. We theoretically analyzes their highly increased key space, which exponentially multiplied static key space in existing authentication methods. To evaluate performance of our design, we built a prototype system and validate authentication mechanism experimentally. In experiments, the proposed authentication algorithm showed a 95% high accuracy rate within different users. In the simulation part, we comprehensively study the impact of sensory data sample size and sensory data loss, which found to be critical factors from experiments on authentication algorithm. Most simulation results validate our algorithm effectively. Growing popularity of electronically based authentication in proximity access control systems calls for a higher security level and greater ubiquity. We believe that authentication bound with dynamic sensory information can effectively enhanced security level of access control systems and will take an important step towards electronically access authentication in the future.

REFERENCES

- [1] Gianni Fenu, Mirko Marras "Controlling User Access to Cloud-Connected Mobile Applications by Means of Biometrics", IEEE Cloud Computing, Vol 5, issue 4, 2018, pp. 47-57.
- [2] Geetanjali Bhola, Divjot Kaur, Mahesh Raj "Dynamic Password Authentication Protocol Using Android Device and One-Way Function", 2017 International Conference on Wireless Communications, Signal Processing and Networking (WiSPNET), 2017, pp. 1863-1866.
- [3] Mohammed Abuhamad, Ahmed Abusnaina, Daehun Nyang, David Mohaisen "Sensor-Based Continuous Authentication of Smartphones' Users Using Behavioral Biometrics: A Contemporary Survey", IEEE Internet of Things Journal ,Vol 8, issue 1, 2021, pp. 65 - 84.
- [4] Yantao Li, Hailong Hu, Gang Zhou "Using Data Augmentation in Continuous Authentication on Smartphones", IEEE Internet of Things Journal, Vol 6. issue 1, 2019, pp. 628-640.
- [5] DONG IN KIM, SHINCHEOL LEE, JI SUN SHIN "A New Feature Scoring Method in Keystroke Dynamics-Based User Authentications", IEEE Access, Vol 8, 2020, pp.27901 - 27914.
- [6] VISHNU SHANKAR, KARAN SINGH "An Intelligent Scheme for Continuous Authentication of Smartphone Using Deep Auto Encoder and Softmax Regression Model Easy for User Brain", IEEE Access, Vol 7, 2019, pp. 48645 - 48654.
- [7] Zhiwen Yu, En Xu, He Du, Bin Guo, Lina Yao "Inferring User Profile Attributes From Multidimensional Mobile Phone Sensory Data" IEEE INTERNET OF THINGS JOURNAL, Vol. 6, issue. 3, 2019, pp. 5153 - 5161.
- [8] Roozbeh Jafari, Jian Wu "Orientation Independent Activity/Gesture Recognition Using Wearable Motion Sensors", IEEE Internet of Things Journal, vol. 6, issue 2, 2018, pp. 1427-1437.
- [9] Yantao Li, Hailong Hu, Gang Zhou, Shaojiang Deng, "Sensor-Based Continuous Authentication Using Cost-Effective Kernel Ridge Regression", Access IEEE, vol. 6, 2018, pp. 32554-32565.
- [10] Sudip Vhaduri, Christian Poellabauer, "Multi-Modal Biometric-Based Implicit Authentication of Wearable Device Users", Information Forensics and Security IEEE Transactions on, vol. 14, no. 12, 2019, pp. 3116-3125.

APPENDIX

A. SOURCE CODE

```
<%@ page import="java.sql.*" %>
<%@ page import="javax.sql.*" %>
<%
String uname=request.getParameter("userName");
session.setAttribute("userName",uname);
String pass=request.getParameter("password");

Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DinamicAuthen
","root","root123");
Statement st=con.createStatement();

ResultSet rs=st.executeQuery("select * from register where UserName='"+uname+"'
and password='"+pass+"'");
if(rs.next())
{
    int count=rs.getInt(8);
    if(count==5)
    {
        %>
        <center>
        <p><h2><font color="BLACK">Your Password Expired..Reset Your
Password..</font></h2></p>
        </center>
        <%@ include file="ResetPassword.jsp"%>
        <%
    }
    else
    {
        count++;
        st.executeUpdate("Update register set Count='"+count+"' where
UserName='"+uname+"'");
        response.sendRedirect("getPass.jsp");
    }
}
else
{
    %>

    <center>
    <p><h2><font color="RED">****Login Failed****</font></h2></p>
    </center>

    <%@ include file="index.jsp" %>
    <%
}
```

```

%>

<%@ page import ="java.sql.*"%>
<%@ page import ="javax.sql.*"%>
<%
String fname=request.getParameter("firstName");
String lname=request.getParameter("lastName");
String uname=request.getParameter("userName");
String mobile=request.getParameter("mobile");
String mail=request.getParameter("email");
String password=request.getParameter("password");
String dynamic="!@#%$%^&*";
int count=0;
Class.forName("com.mysql.jdbc.Driver");
Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/DinamicAuthent","root","root123");
Statement st=con.createStatement();
ResultSet rs=st.executeQuery("Select * from register where
UserName='"+uname+"'");
if (rs.next())
{
    %>
    <center>
    <p><h2><font color="RED">User Name Already Available..Please Change Your
User Name..</font></h2></p>
    </center>
    <%@ include file="register.jsp" %>
    <%

}
else
{
    int i=st.executeUpdate("Insert into register
values('"+fname+"','"+lname+"','"+uname+"','"+mobile+"','"+mail+"','"+password+"','"+
dynamic+"','"+count+"')");

    if(i>0)
    {
        %>
        <center>
        <p><h2><font color="BLUE">Registration Success</font></h2></p>
        </center>
        <%@ include file="index.jsp"%>
        <%

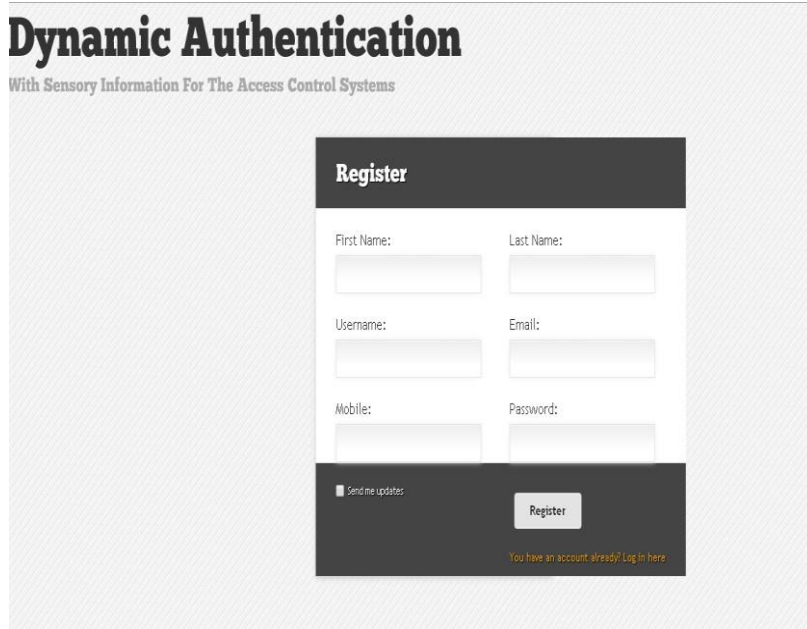
    }
    else
    {
        response.sendRedirect("index.jsp");
    }
}%>

```

B. SCREENSHOTS

1. REGISTRATION

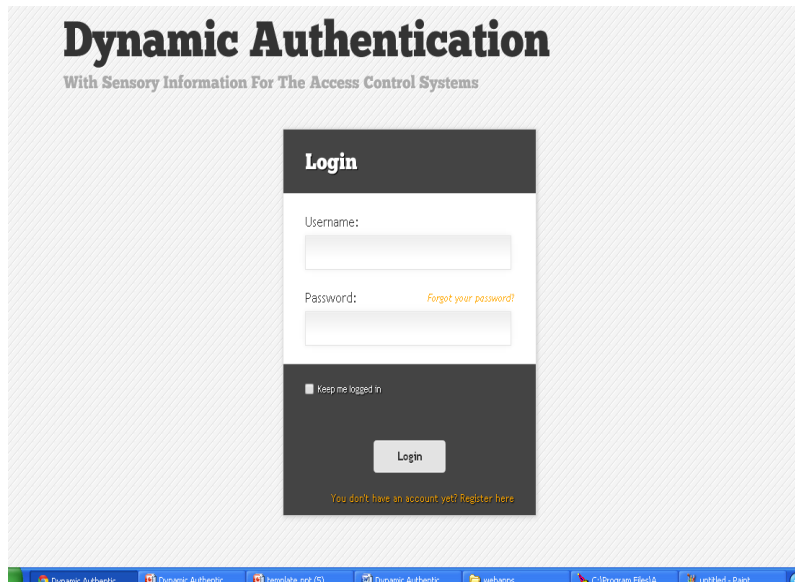
The initial registration is shown



The screenshot shows the 'Dynamic Authentication' registration page. The page has a light gray background with a subtle diagonal line pattern. At the top, the title 'Dynamic Authentication' is in a large, bold, black font, followed by the subtitle 'With Sensory Information For The Access Control Systems' in a smaller, gray font. Below this is a dark gray rectangular box with the word 'Register' in white. Inside this box, there are six input fields arranged in two columns: 'First Name:', 'Last Name:', 'Username:', 'Email:', 'Mobile:', and 'Password:'. Each field has a light gray border and a small shadow. At the bottom of the box, there is a checkbox labeled 'Send me updates' and a 'Register' button. Below the button, there is a link that says 'You have an account already? Log in here'.

2. LOGIN

The login page is displayed. This allows initial login

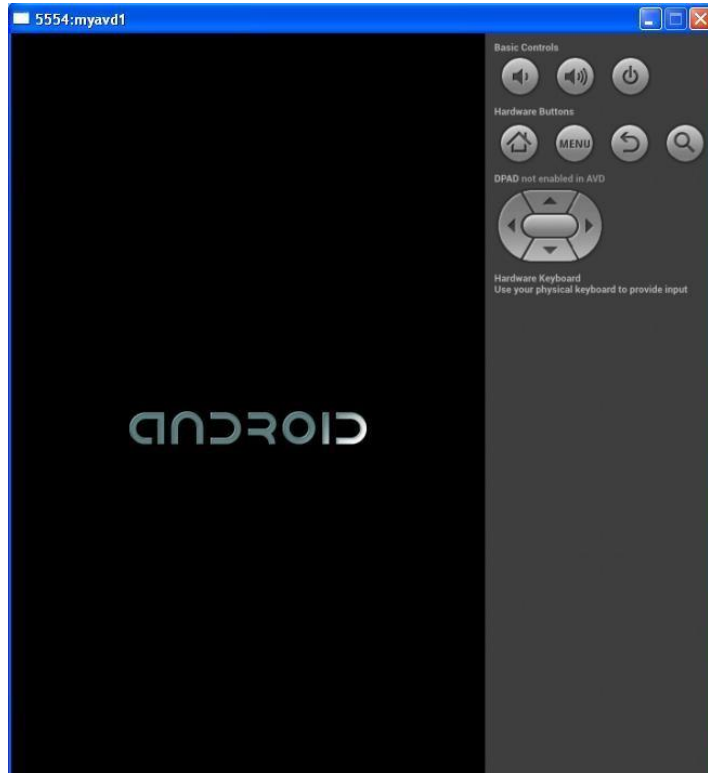


The screenshot shows the 'Dynamic Authentication' login page. The page has the same light gray background and diagonal line pattern as the registration page. At the top, the title 'Dynamic Authentication' is in a large, bold, black font, followed by the subtitle 'With Sensory Information For The Access Control Systems' in a smaller, gray font. Below this is a dark gray rectangular box with the word 'Login' in white. Inside this box, there are two input fields: 'Username:' and 'Password:'. The 'Password:' field has a small orange link that says 'Forgot your password?'. Below the input fields, there is a checkbox labeled 'Keep me logged in' and a 'Login' button. Below the button, there is a link that says 'You don't have an account yet? Register here'. At the bottom of the screenshot, a Windows taskbar is visible with several open applications, including 'Dynamic Authentication', 'Internet Explorer', 'Windows Explorer', 'Webcam', 'Call Program Files', and 'Untitled - Paint'.

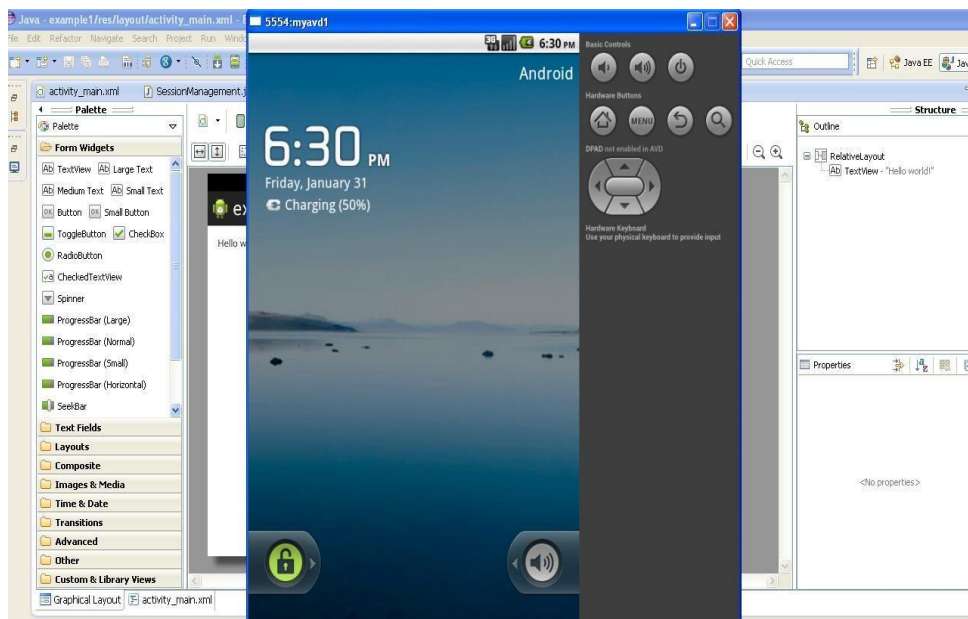
3. ANDROID DESIGNS

The Application designing pages are shown

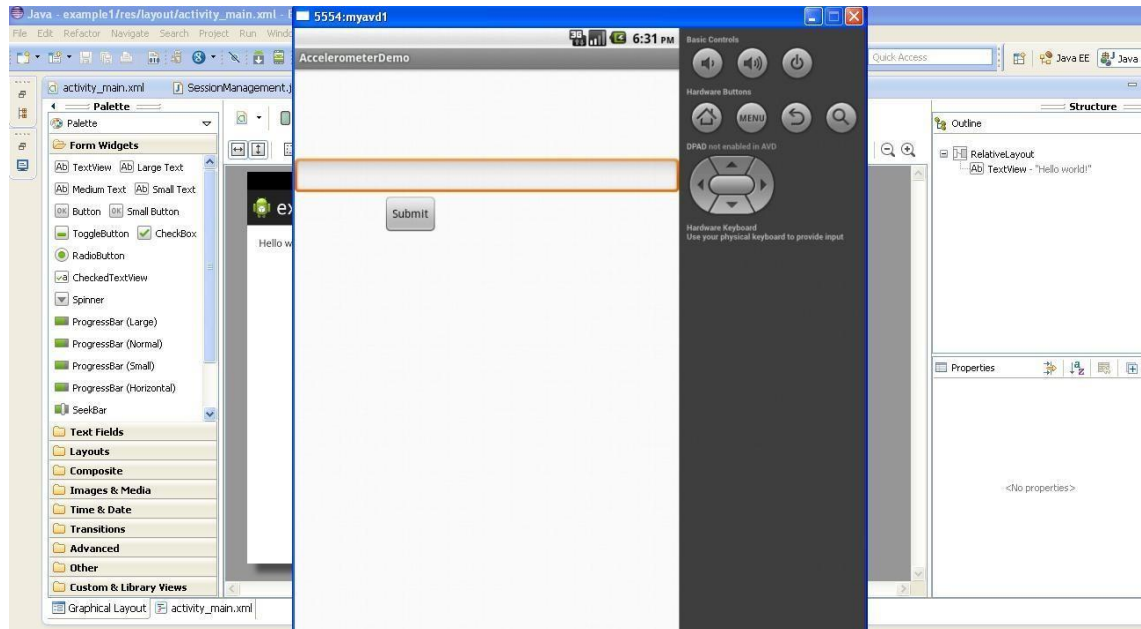
a) Design 1



b) Design 2

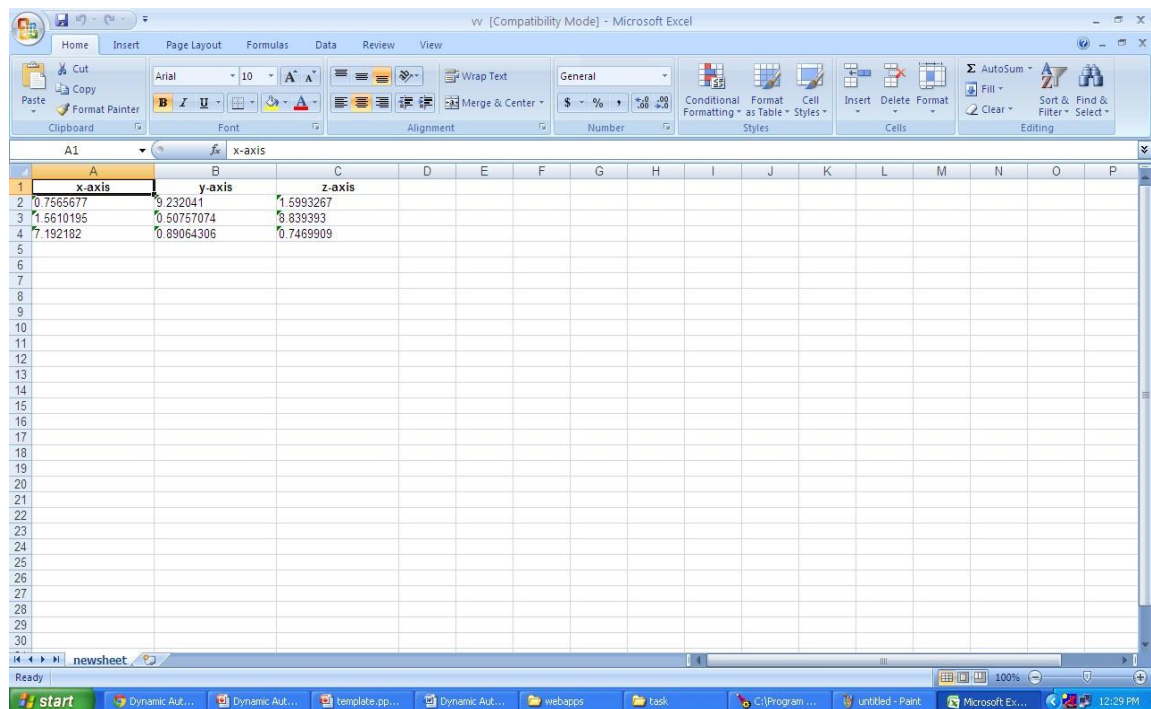


c) Design 3



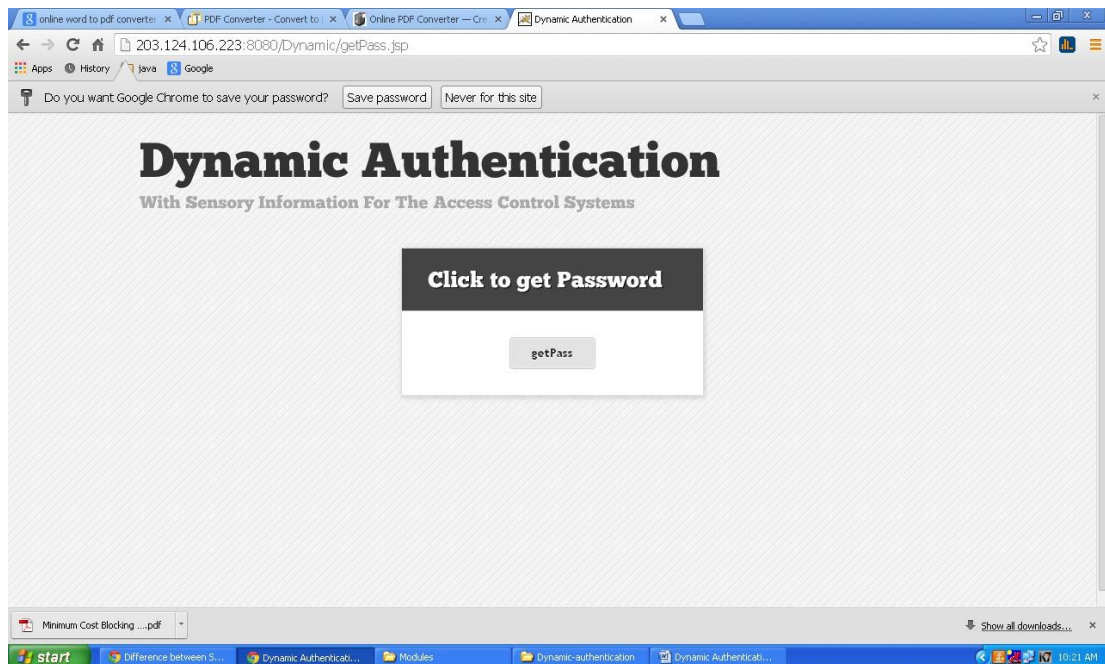
4. DATABASE

The values obtained during rotation are stored in the database



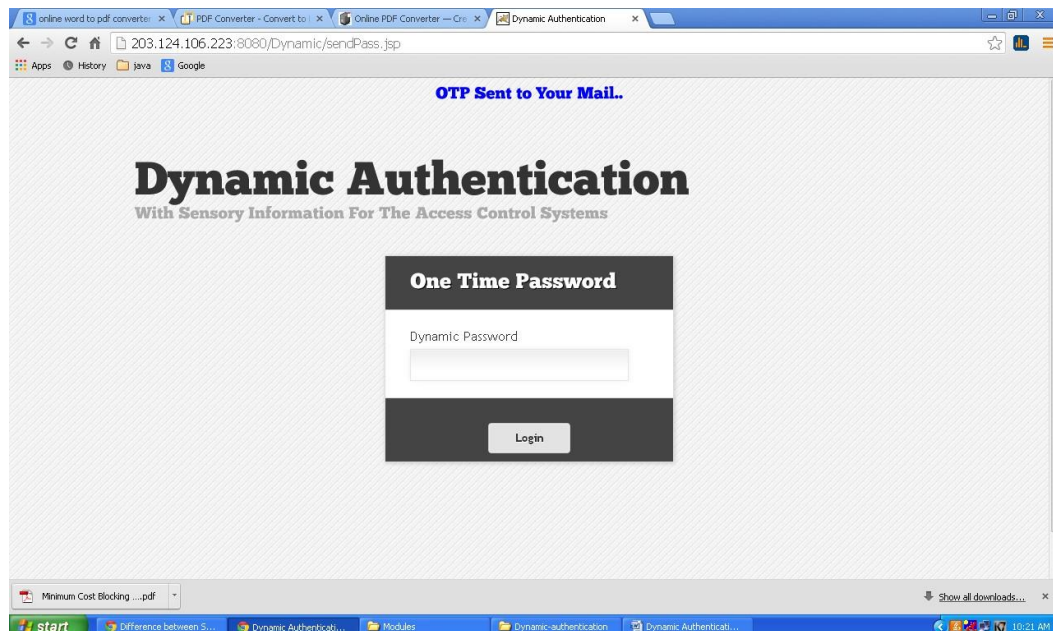
5. GETTING THE DYNAMIC PASSWORD

From the values obtained a common password is generated



6. ACCESSING USING DYNAMIC PASSWORD

User logs in using the dynamic password received through mail



C. PLAGARISM REPORT

