

Private Identity-Based Encryption for Key Management

Submitted in partial fulfilment of the requirements for the award of Bachelor
of Engineering in Computer Science and Engineering

Project was done by

Abhrajyoti Das(38110008)

Mriganka Das(38110329)



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

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 - 2022



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited "A" Grade by NAAC | 12B Status by UGC | Approved by AICTE

www.sathyabama.ac.in

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this Project Report is the bonafide work of MRIGANKA DAS (38110329) ABHRAJYOTI DAS (38110008) who carried out the project entitled "**Private Identity-Based Encryption for Key Management**" under my supervision from December 2021 to March 2022.

Internal Guide

(Ms S.R.SRIVIDHYA)

Head of the Department

(Dr. L. LAKSHMANAN)

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

ACKNOWLEDGEMENT

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

We convey our thanks to **Dr. T.Sasikala M.E., Ph.D, Dean**, School of Computing **Dr.L.Lakshmanan M.E., Ph.D.**, and **Dr.S.VigneshwariM.E., Ph.D. Head** of the Department of Computer Science and Engineering for providing unnecessary 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 **Ms.S.R.Srividhya** for her valuable guidance, suggestions and constant encouragement that 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 Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

An Identity-Based Encryption (IBE) scheme uses public identities of entities for cryptographic purposes. Unlike that, we introduce a new scheme which is based on private identities, and we call it Private Identity-Based Encryption. A Private IBE scheme makes sure the adversaries cannot get the information that somebody uses for encryption in order to decrypt the data. Moreover, thanks to using identities as secret keys, a user-friendly system can be designed to support users in protecting data without storing any keys privately. This allows builds decentralized applications to manage keys that are often long and difficult to remember.

INDEX

Chapter No.	Title	Page No.
1	Introduction	1
	1.1. Existing system	2
	1.1.1. Disadvantages of the existing system	2
	1.2. Proposed system	3
	1.2.1. Disadvantages of the proposed system	3
	1.3. System specification	3
	1.3.1. Hardware Requirements	
	1.3.2. Software Requirements	
2	Literature Survey	9
3	Modules	13
	3.1 Module Description	13
	3.1.1 Server Module	13
	3.1.2 Client Module	13
	3.1.3 Encryption	14
	3.1.4 Decryption	14
4.	Language Specification	15
	4.1 ODBC	20
	4.2 JDBC	22
	4.3 Netbeans	25

	4.4 MySQL Server	35
5.	System Testing	38
	5.1 Feasibility Study	38
	5.1.1. Economical Feasibility	39
	5.1.2 Technical Feasibility	39
	5.1.3. Social Feasibility	39
	5.2 System, Testing and Maintenance	39
	5.3 Unit Testing	41
	5.4 Integration Testing	41
6.	Conclusion	42
7.	Appendix	43
	7.1 Screenshots	43
	7.2 Source Code	50
8.	References	56

LIST OF FIGURES

Serial No.	Name	Page No:
1.1	Architecture Diagram	4
1.2.1	Usecase Diagram	5
1.2.2	Class Diagram	6
1.2.3	Data Flow Diagram	7
1.2.4	Activity Diagram	8
4.3.1	JDBC Architecture	25
4.4.1	Architecture of Apache Tomcat	29
4.4.2	Working of Server	31
4.5.1	My SQL Server Architecture Diagram	35

CHAPTER-1

INTRODUCTION:

An identity is a set of information which can be used to identify an entity. If an identity matches an entity, there is only one such entity, i.e. there are not two different entities that have the same identity. This implies that identities are unique. The uniqueness of identities can be used as keys in cryptographic schemes. Applying this property to the public-key cryptography, Adi Shamir proposed a scheme called Identity-Based Encryption (IBE). In this scheme, the public key of the receiver can be created based on his identity, and then the sender can use the key to encrypt and send messages to the receiver. The sender needs to know the identity, e.g. names or email addresses, in order to send to the receiver. Thus, the identity, in this case, is public. However, applying that property to secret-key cryptography is another problem. In secret-key cryptography, the keys are secret, so the identities are also secret.

To protect the secret, a cryptographically secure pseudorandom number generator is used to generate keys, but cannot be applied to identity information in the same way, because it is predetermined and unchanged. For that reason, instead of generating, we design a scheme that selects at random the information from the private identity through a questionnaire. To ensure the safety of the questionnaires, we also propose security scenarios and prove that our method is secure under them. After all, we construct an application using our scheme for key management. In the application, the keys do not need to be stored, but only the information to remind users of the generating process is stored. Therefore, the application can be built in a decentralized manner.

Various decentralized systems are designed based on Blockchain technology such as Bitcoin, Ethereum. To create transactions on Blockchain, and user need to have an address associated with a public/secret key pair. But in our application, the key pair is used one time to create only one transaction, and then the user does not need to keep it. Hence, the key pair is considered as a one-time account. In summary, we make the following contributions: A scheme uses questionnaires to get private identities to create keys for a secret-key encryption. Security scenarios are used to evaluate the security of

the scheme is based on questionnaires. A decentralized application based on the scheme is used for key management.

1.1 EXISTING SYSTEM:

In the existing system, a sender can encrypt a message or information for a receiver knowing just the identity of the receiver and importantly, without obtaining and verifying the receiver's public-key certificate. Applying that property to secret-key cryptography is another problem. To protect the secret, a cryptographically secure pseudorandom number generator is used to generate, but cannot be applied to identity information in the same way, because it is predetermined and unchanged.

1.1.1. DISADVANTAGES OF EXISTING SYSTEM:

- Consequently, with a standardized public-key string format, an IBE scheme completely eliminates the need for public-key certificates.
- The identity, in this case, is public. However, applying that property to secret-key cryptography is another problem.

1.2. PROPOSED SYSTEM:

In the proposed system, a private key generator will generate the public key and private key for the certified users. So that the users will receive each other's public key for identification and the data has been encrypted through private key. We propose security scenarios and prove that our method is secure under them. After all, we construct an application using our scheme for key management. In the application, the keys do not need to be stored, but only the information to remind users of the generating process is stored. Therefore, the application can be built in a decentralized manner.

1.2.1 ADVANTAGES OF THE PROPOSED SYSTEM:

- Scheme uses questionnaires to get private identities to create keys for a secret-key encryption.
- A decentralized application based on the scheme is used for key management.

1.3. SYSTEM SPECIFICATION

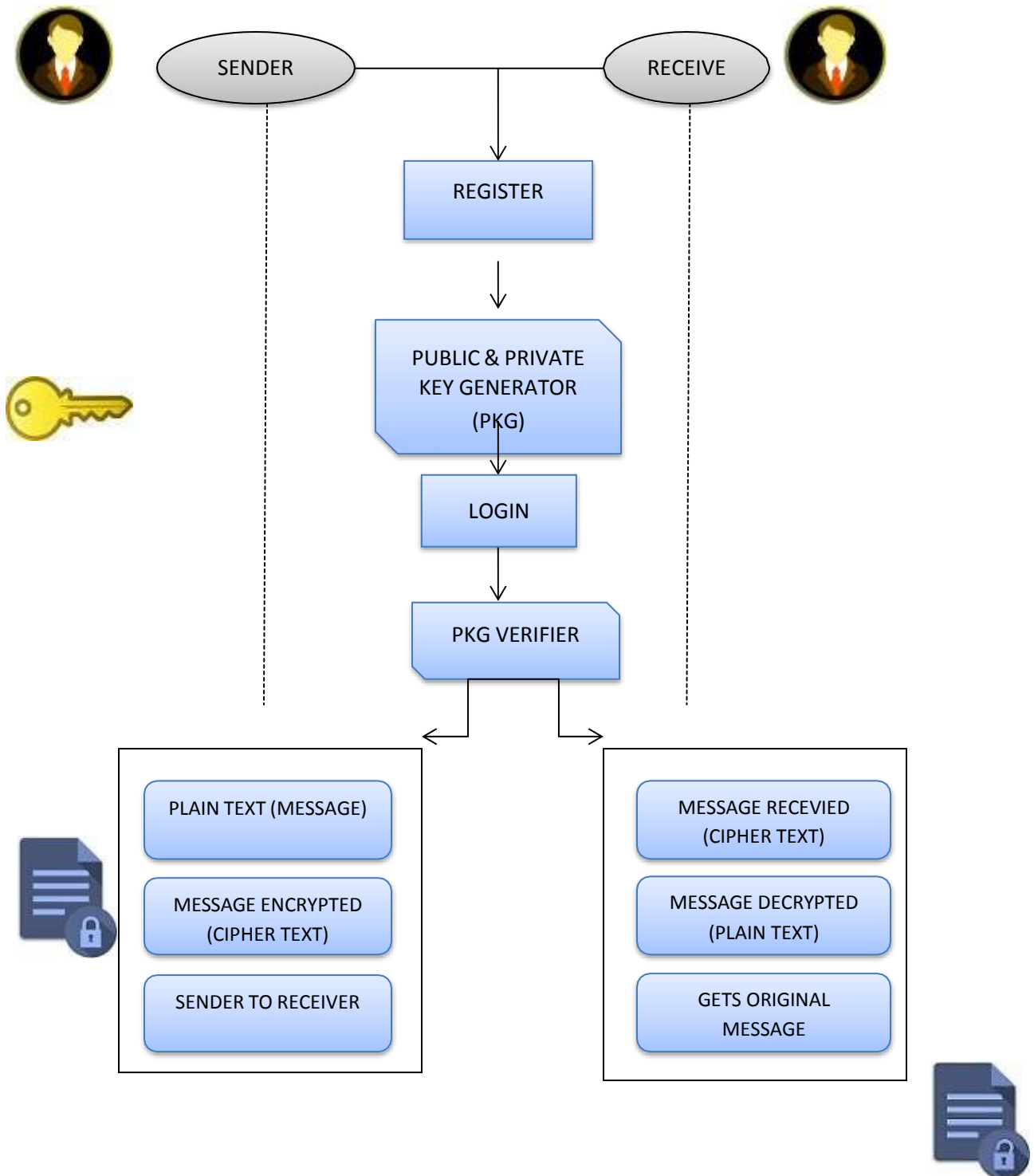
1.3.1. HARDWARE REQUIREMENTS:

- Processor: Intel Core i3 Processor
- Speed: 2.5 GHz
- RAM : 2GB(min)
- Hard Disk: 500MB
- Key Board: Standard Windows Keyboard
- Mouse: Two or Three Button Mouse
- Monitor : LCD

1.3.2. SOFTWARE REQUIREMENTS:

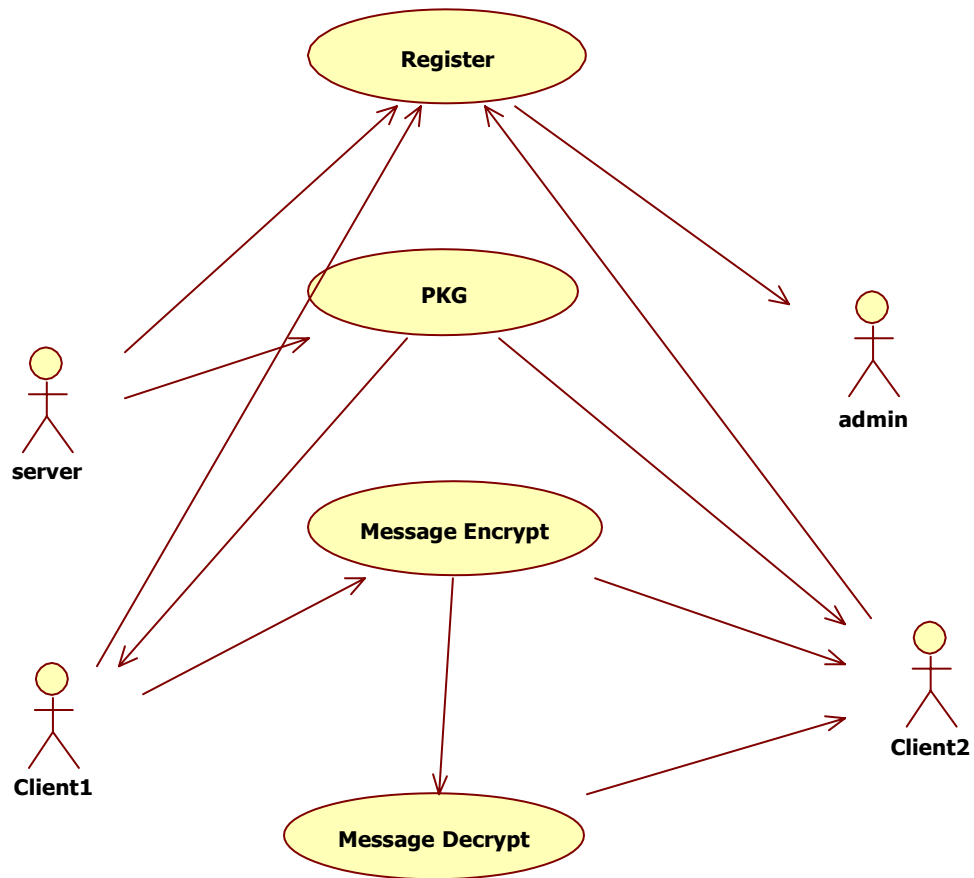
- Operating System : Windows7/10.
- Application Server : Tomcat6.0/7/8.X.
- Front End : Java , HTML,CSS
- Scripts : JavaScript.
- Server side Script : Java Server Pages.
- IDE : Net beans
- Back End : MYSQL 5.0/ Heidi SQL 8.1
- Database Connectivity : JDBC

1.1 ARCHITECTURE DIAGRAM:

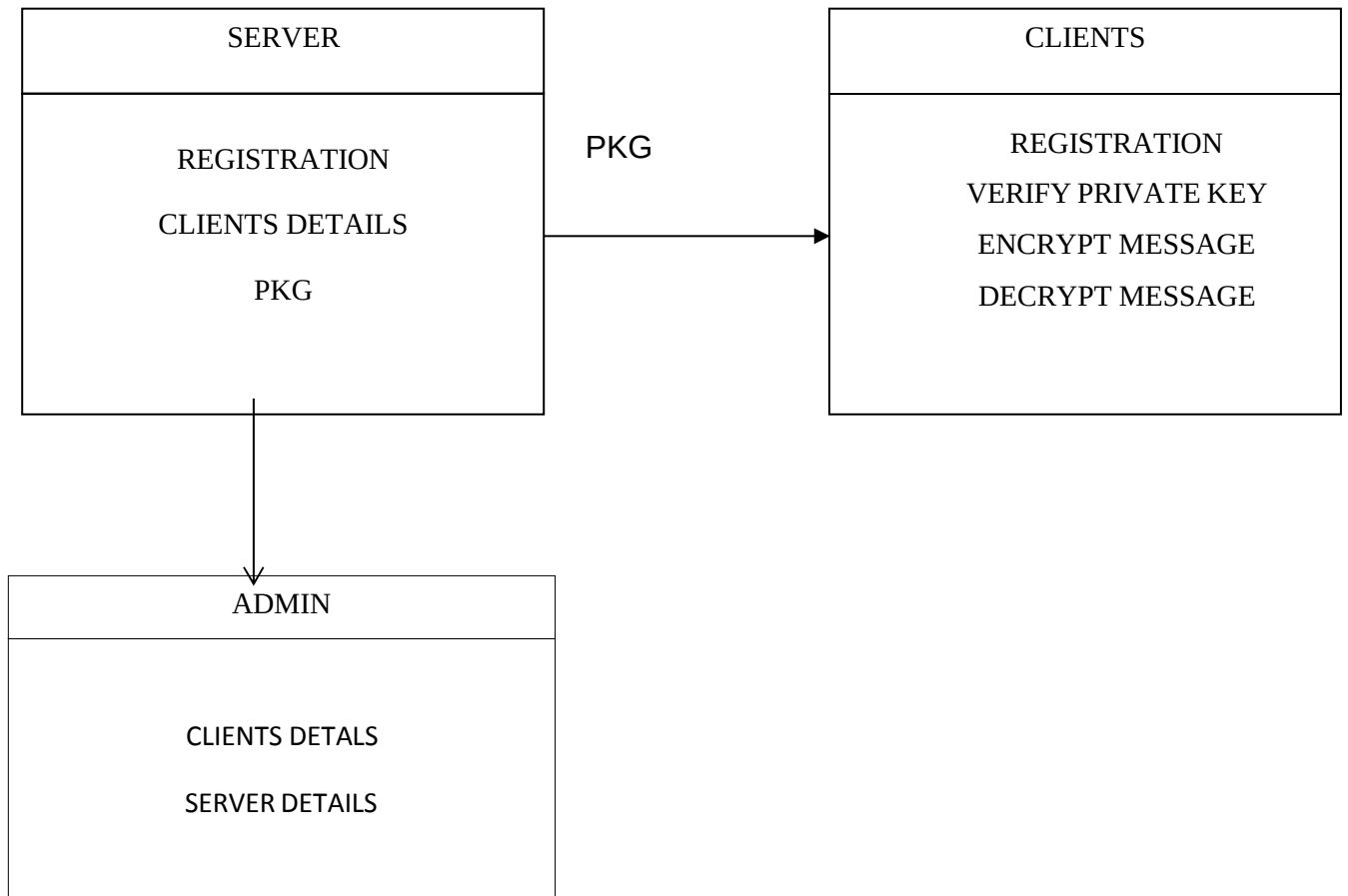


1.2 UML DIAGRAMS:

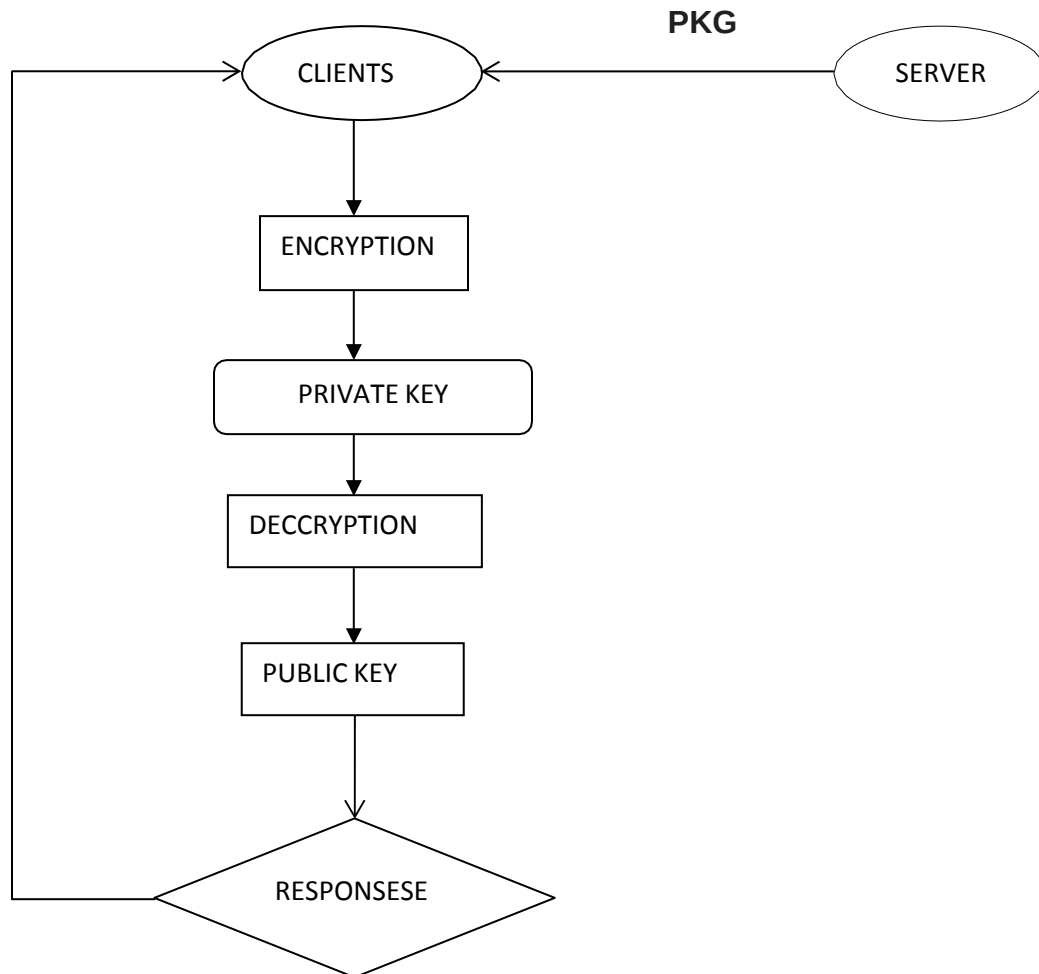
1.2.1 USECASE DIAGRAM:



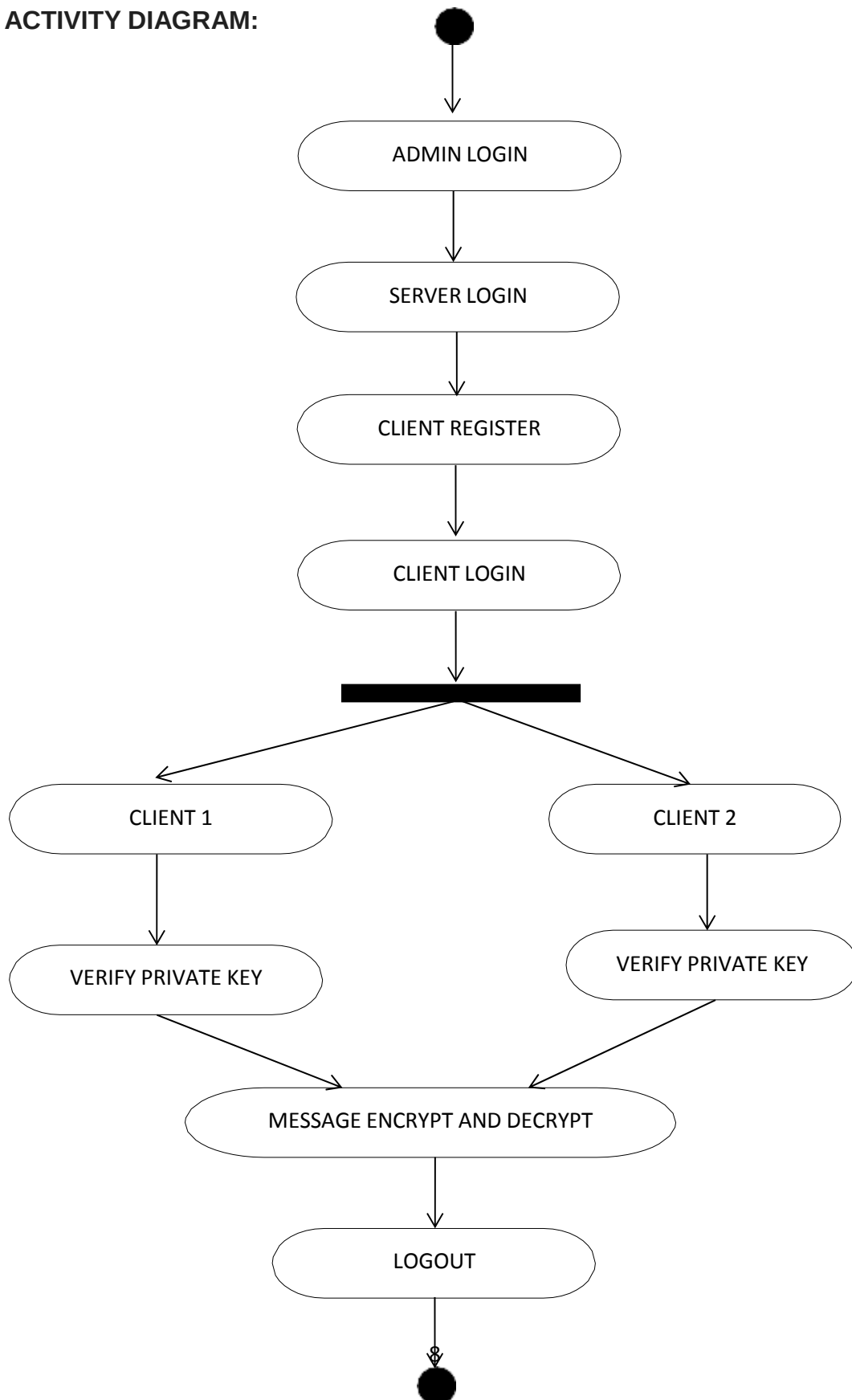
1.2.2 CLASS DIAGRAM:-



1.2.3 DATA FLOW DIAGRAM:-



1.2.4 ACTIVITY DIAGRAM:



CHAPTER-2

LITERATURE SURVEY

TITLE: Secure Key-Reduplication with Identity-Based Broadcast Encryption.

YEAR OF PUBLISHING: 2021

AUTHOR NAME: Ling Liu, Yuqing Zhang.

ABSTRACT:

For the purpose of ensuring data confidentiality, they are usually encrypted before being outsourced. Traditional encryption will inevitably result in multiple different ciphertexts produced from the same plaintext by different users' secret keys, which hinders data deduplication. Convergent encryption makes deduplication possible since it naturally encrypts the same plaintexts into the same ciphertexts. One attendant problem is how to reliably and effectively manage a huge number of convergent keys. Several deduplication schemes have been proposed to deal with the convergent key management problem. However, they either need to introduce key management servers or require interaction between data owners. In this paper, we design a novel client-side deduplication protocol named KeyD without such an independent key management server by utilizing the identity-based broadcast encryption (IBBE) technique. Users only interact with the cloud service provider (CSP) during the process of data upload and download. Security analysis demonstrates that KeyD ensures data confidentiality and convergent key security, and well protects the ownership privacy simultaneously.

TITLE: Identity-Based Hybrid Format-Preserving Encryption Scheme.

YEAR OF PUBLISHING: 2020

AUTHOR NAME: Shijin Li, Yijia GAO.

ABSTRACT:

The format-preserving encryption has the characteristics of the encrypted data format and the same data length, and does not break the data format constraints, thereby reducing the cost of modifying the data format. The analysis of existing sensitive information format-preserving encryption schemes is based on a symmetric encryption system, which has problems such as low key transmission security and high key management cost. This paper proposes an identity-based format-preserving encryption scheme. Compared with the existing format-preserving encryption scheme, the communication parties do not need to transmit a key, and the encryption key and the decryption key It is proved that the scheme satisfies the identity-based pseudo-random permutation security, and at the same time, the scheme satisfies the adaptive selection of ciphertext indistinguishability under plaintext attack.

TITLE: Efficient Identity-based Encryption with Revocation for Data Privacy in Internet of Things.

YEAR OF PUBLISHING: 2021.

AUTHOR NAME: Pushpita Chatterjee, Yi Chen.

ABSTRACT:

The Internet of Things (IoT) is making the world around us smarter and more convenient. However, its extensive application has rendered security problems, such as the privacy of sensitive data, increasingly serious. Encryption provides an effective and important means of protecting data privacy in the IoT. Because of resource limitations, to achieve high efficiency IoT devices require an encryption scheme that ensures that the encryption phase does not incur a heavy data transmission overhead. By virtue of its many advantages, the use of public key encryption in current applications is widespread. Identity-based public key encryption (IBE) removes the obstacle raised by the sophisticated certificate management required by other schemes, and its efficiency renders it more suitable for application in the IoT. However, a problem that needs to be solved in IBE is the revocation of a user whose private key may have been exposed. In

this paper, we present an efficient and practical IBE scheme having revocation functionality to preserve data privacy in IoT applications. Elements in the system, such as sensors and actuators, can exchange encrypted data directly or via a cloud server. If a private key is compromised, the private key generator can revoke its user. The security of our proposed scheme can be proved based on SM9 encryption and the bilinear Diffie–Hellman problem.

TITLE: Adaptively Secure Puncturable Identity-Based Proxy Re-Encryption Scheme for Securing Group Message

YEAR OF PUBLISHING: 2021.

AUTHOR NAME: Hu Xiong, Lili Wang.

ABSTRACT:

Puncturable proxy re-encryption (PPRE) is envisioned to provide secure access control delegation and fine-grained forward security for asynchronous group messaging systems. Nevertheless, the existing PPRE scheme not only suffers from the burden of certificate management, but also merely achieves selective security based on the non-standard assumption. In this paper, a Puncturable identity-based proxy re-encryption (P-IB-PRE) scheme is proposed to efficiently protect the security and privacy of the group message. The proposed scheme introduces a message server as the proxy to transform ciphertext for each participant in the group, thus the heavy computation overhead is delegated to the message server with abundant resources. Most importantly, our scheme enables the recipient to revoke its private key's decryption capability of the specific messages without affecting other messages. Moreover, the identity-based mechanism eliminates the burden of certificate management as well as improves efficiency. The proposed scheme achieves adaptive security under the standard Decisional Bilinear Diffie-Hellman (DBDH) assumption. Eventually, theoretical and experimental analysis demonstrates that the proposed scheme has an excellent performance in efficiency and practicality.

TITLE: Identity-Based Encryption Transformation for Flexible Sharing of Encrypted Data in Public Cloud.

YEAR OF PUBLISHING: 2020.

AUTHOR NAME: Hua Deng, Zheng Qin.

ABSTRACT:

With the rapid development of cloud computing, an increasing number of individuals and organizations are sharing data in the public cloud. To protect the privacy of data stored in the cloud, a data owner usually encrypts his data in such a way that certain designated data users can decrypt the data. This raises a serious problem when the encrypted data needs to be shared to more people beyond those initially designated by the data owner. To address this problem, we introduce and formalize an identity-based encryption transformation (IBET) model by seamlessly integrating two well-established encryption mechanisms, namely identity-based encryption (IBE) and identity-based broadcast encryption (IBBE). In IBET, data users are identified and authorized for data access based on their recognizable identities, which avoids complicated certificate management in usual secure distributed systems. More importantly, IBET provides a transformation mechanism that converts an IBE ciphertext into an IBBE ciphertext so that a new group of users not specified during the IBE encryption can access the underlying data. We design a concrete IBET scheme based on bilinear groups and prove its security against powerful attacks. Thorough theoretical and experimental analyses demonstrate the high efficiency and practicability of the proposed scheme.

CHAPTER-3

MODULES

- 1. SERVER MODULE**
- 2. CLIENT MODULE**
- 3. ENCRYPTION**
- 4. DECRYPTION**

3.1. MODULES DESCRIPTION:

3.1.1. SERVER MODULE:-

In this module server register the account and create a master key for access the server module data. The details uploaded in the server are in the encrypted format. To access the data stored in the server the user have to forward a request to the server. The server generates a public key and forwarded to the request user. The client module will decrypt the data and read content.

3.1.2. CLIENT MODULE:-

In this module the user will forward a request the server to decrypt the detail available in the server. The site will transfer the request and receives the key from the server. The data in the decrypted format were visible to the requested user.

3.1.3. ENCRYPTION:

Encryption converts data into scrambled text. The unreadable text can only be decoded with a key. By using message encryption, a message sender can be sure that the content of the message has not been modified before reaching the recipient. The message is then addressed to each potential recipient specified in that policy, using the users' public keys. Message encryption ensures that the sender and the intended recipient are the only parties that can read a message's content. Outsiders cannot decrypt the message, as they do not possess the encryption key.

3.1.4. DECRYPTION:

The conversion of encrypted data into its original form is called Decryption. It is generally a reverse process of encryption. It decodes the encrypted information so that an authorized user can only decrypt the data because decryption requires a secret key or password.

CHAPTER-4

LANGUAGE SPECIFICATIONS:

Java Technology

Java technology is both a programming language and a platform.

The Java Programming Language

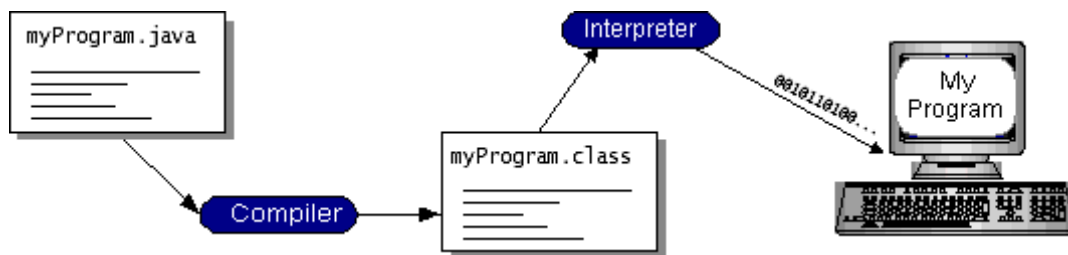
The Java programming language is a high-level language that can be characterized by all of the following buzzwords:

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

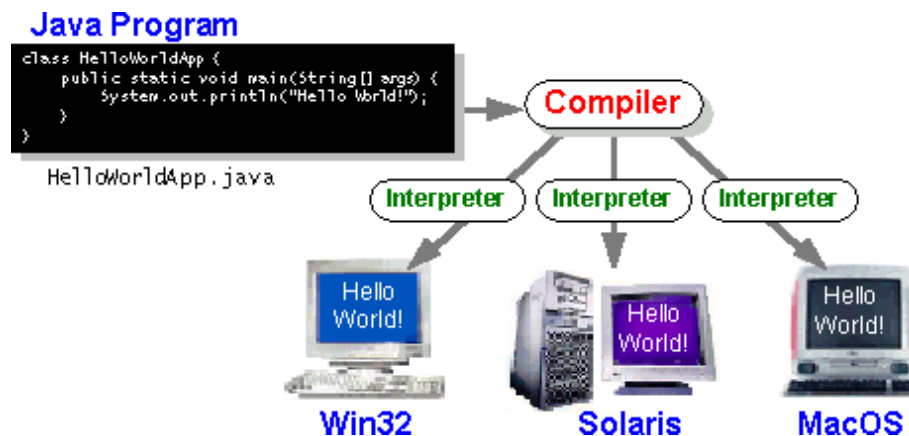
With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called *Java 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.



You can think of Java bytecodes 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 bytecodes help make “write once, run anywhere” possible. You can compile your program into bytecodes on any platform that has a Java compiler. The bytecodes 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. 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.

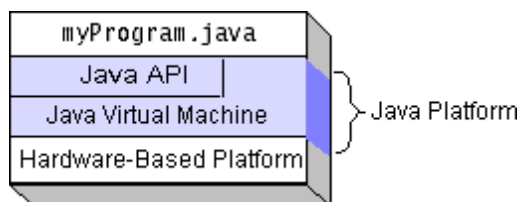
The Java platform has two components:

- *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 bytecode compilers can bring performance close to that of native code without threatening portability.

What Can Java Technology Do?

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

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

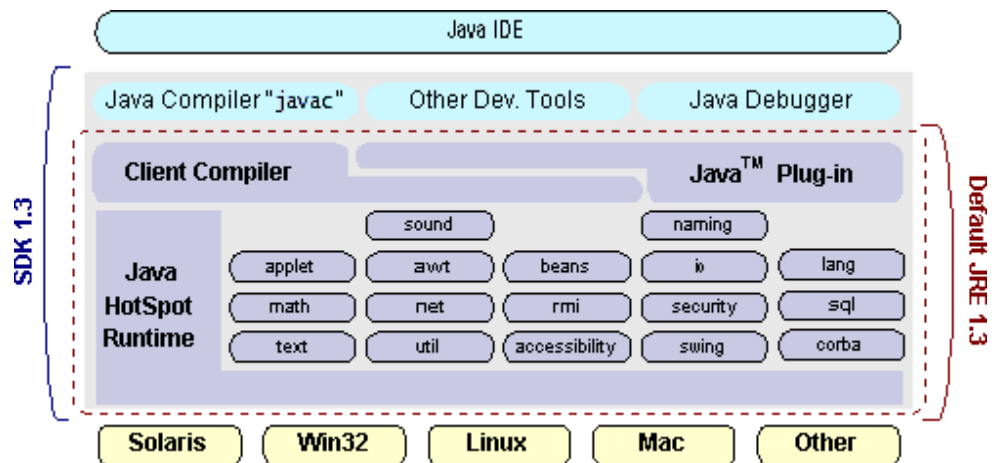
An application is a standalone program that runs directly on the Java platform. A special kind of application known as a *server* serves and supports clients on a network. Examples of servers are Web servers, proxy servers, mail servers, and print servers. Another specialized program is a *servlet*.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

How does the API support all these kinds of programs? It does so with packages of software components that provide 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.



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 as 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.

4.1 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 an MYSQL 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 MYSQL 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 MYSQL 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 MYSQL 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.

4.2 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 MYSQL database access mechanism that provides a consistent interface to a variety of RDBMS. 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 design goals for JDBC are as follows:

1. MYSQL Level API

The designers felt that their main goal was to define a MYSQL 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.

2. MYSQL Conformance

MYSQL 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 MYSQL API must “sit” on top of other common MYSQL 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

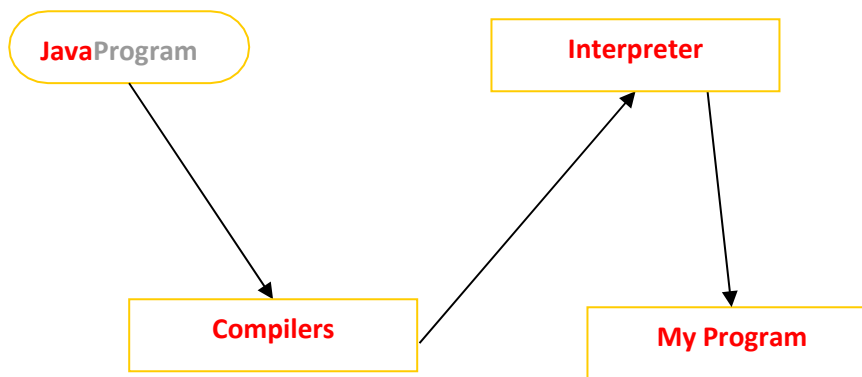
Strong typing allows for more error checking to be done at compile-time; also, fewer errors appear at runtime.

7. Keep the common cases simple

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

Finally we decided to proceed the implementation using Java Networking. And for dynamically updating the cache table we go for MS Access database.

Java is also unusual in that each java program is both compiled and interpreted. with a compile you translate a java program into an intermediate language called java byte codes the platform-independent code instruction is passed and run on the computer. compilation happens just once; interpretation occurs each time the program is executed. the figure illustrates how this works.



4.3.1 JDBC ARCHITECTURE

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 java development tool or a web browser that can run Java applets, is an implementation of the java VM. the java VM can also be implemented in hardware. Java byte codes help make “write once, run anywhere” possible. You can compile your java program into byte codes on my platform that has a java compiler. the byte codes can then be run any implementation of the java vm. for example, the same java program can run windows nt, Solaris, and Macintosh.

4.3 Netbeans:

NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5. It is also an application platform framework for Java desktop applications and others. The NetBeans IDE is written in Java and can run on Windows, OS X, Linux, Solaris and other platforms supporting a compatible JVM.

The NetBeans Platform allows applications to be developed from a set of modular software components called modules. Applications based on the NetBeans Platform (including the NetBeans IDE itself) can be extended by third-party developers.

NetBeans Platform Features:

The main reusable features and components comprising the NetBeans Platform are outlined below.

Module System:

The modular nature of a NetBeans Platform application gives you the power to meet complex requirements by combining several small, simple, and easily tested modules encapsulating coarsely-grained application features. Powerful versioning support helps give you confidence that your modules will work together, while strict control over the public APIs your modules expose will help you create a more flexible application that's easier to maintain. Since your application can use either standard NetBeans Platform modules or OSGi bundles, you'll be able to integrate third-party modules or develop your own.

Lifecycle Management:

Just as application servers, such as GlassFish or WebLogic, provide lifecycle services to web applications, the NetBeans runtime container provides lifecycle services to Java desktop applications. Application servers understand how to compose web modules, EJB modules, and related artefacts, into a single web application.

In a comparable manner, the NetBeans runtime container understands how to compose NetBeans modules into a single Java desktop application.

There is no need to write the main method for your application because the NetBeans Platform already contains one. Also, support is provided for persisting user settings across restart of the application, such as, by default, the size and positions of the windows in the application.

Pluggability, Service Infrastructure, and File System:

End users of the application benefit from pluggable applications because these enable them to install modules into their running applications. NetBeans modules can be installed, uninstalled, activated, and deactivated at runtime, thanks to the runtime

container. The NetBeans Platform provides an infrastructure for registering and retrieving service implementations, enabling you to minimize direct dependencies between individual modules and enabling a loosely coupled architecture (high cohesion and low coupling).

The NetBeans Platform provides a virtual file system, which is a hierarchical registry for storing user settings, comparable to the Windows Registry on Microsoft Windows systems. It also includes a unified API providing stream-oriented access to flat and hierarchical structures, such as disk-based files on local or remote servers, memory-based files, and even XML documents.

Window System, Standardized UI Toolkit, and Advanced Data-Oriented Components:

Most serious applications need more than one window. Coding good interaction between multiple windows is not a trivial task. The NetBeans window system lets you maximize/minimize, dock/undock, and drag-and-drop windows, without you providing any code at all. Swing and JavaFX are the standard UI toolkits on the Java desktop and can be used throughout the NetBeans Platform. Related benefits include the ability to change the look and feel easily via "Look and Feel" support in Swing and CSS integration in JavaFX, as well as the portability of GUI components across all operating systems and the easy incorporation of many free and commercial third-party Swing and JavaFX components.

With the NetBeans Platform, you're not constrained by one of the typical pain points in Swing: the JTree model is completely different to the JList model, even though they present the same data. The NetBeans Nodes API provides a generic model for presenting your data. The NetBeans Explorer & Property Sheet API provides several advanced Swing components for displaying nodes. In addition to a window system, the NetBeans Platform provides many other UI-related components, such as a property sheet, a palette, complex Swing components for presenting data, a Plugin Manager, and an Output window.

Miscellaneous Features, Documentation, and Tooling Support:

The NetBeans IDE, which is the software development kit (SDK) of the Netbeans Platform, provides many templates and tools, such as the award winning Matisse GUI Builder that enables you to very easily design your application's layout. The NetBeans Platform exposes a rich set of APIs, which are tried, tested, and continually being improved. The community is helpful and diverse, while a vast library of blogs, books, tutorials, and training materials are continually being developed and updated in multiple languages by many different people around the world.

Apache Tomcat:

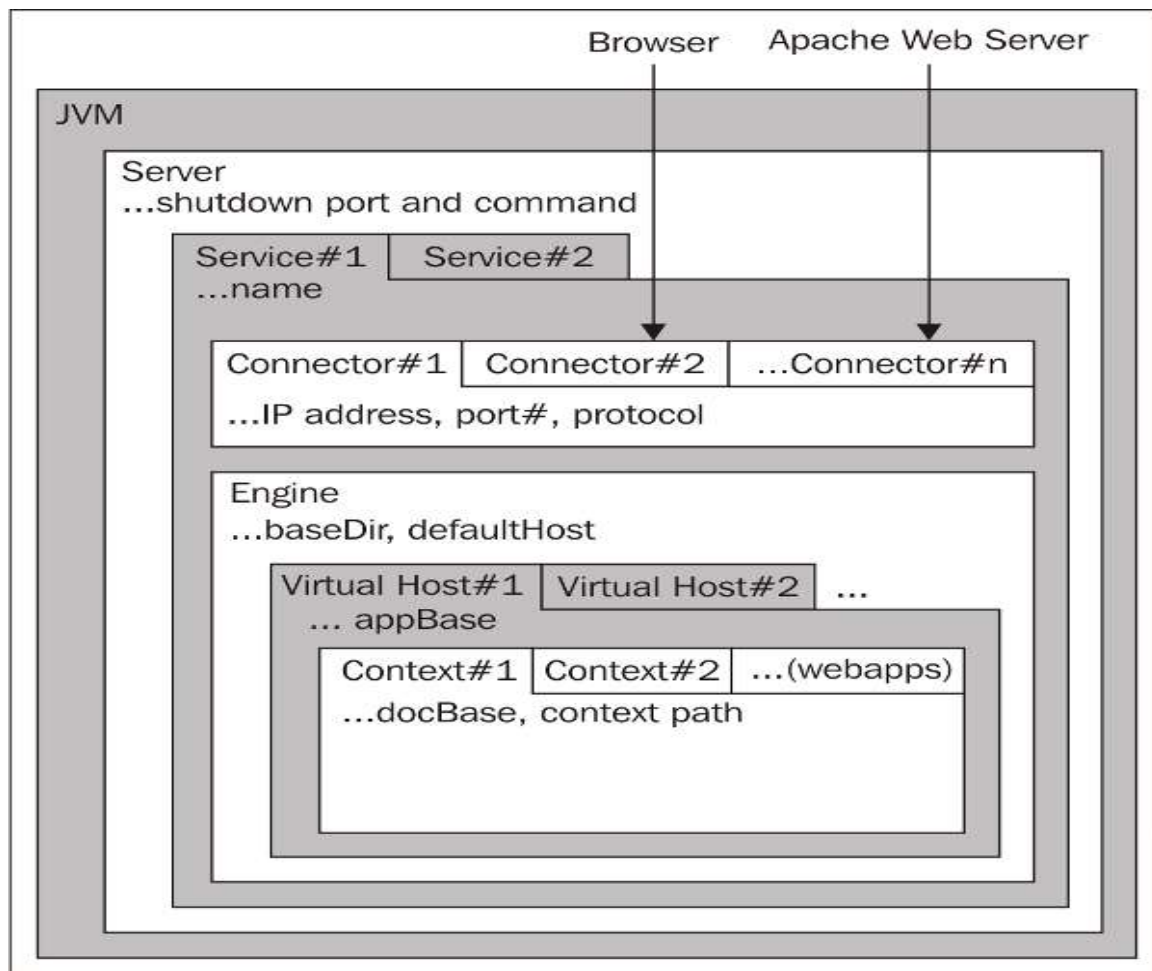
Apache Tomcat (or simply Tomcat, formerly also Jakarta Tomcat) is an open source web server and servlet container developed by the Apache Software Foundation (ASF). Tomcat implements the Java Servlet and the JavaServer Pages (JSP) specifications from Oracle Corporation, and provides a "pure Java" HTTP web server environment for Java code to run.

Component taxonomy:

Tomcat's architecture follows the construction of a Matrushka doll from Russia. In other words, it is all about containment where one entity contains another, and that entity in turn contains yet another. In Tomcat, a 'container' is a generic term that refers to any component that can contain another, such as a Server, Service, Engine, Host, or Context. Of these, the Server and Service components are special containers, designated as Top Level Elements as they represent aspects of the running Tomcat instance. All the other Tomcat components are subordinate to these top level elements. The Engine, Host, and Context components are officially termed Containers, and refer to components that process incoming requests and generate an appropriate outgoing response.

Nested Components can be thought of as sub-elements that can be nested inside either Top Level Elements or other Containers to configure how they function. Examples of nested components include the Valve, which represents a reusable unit of work; the

Pipeline, which represents a chain of Valves strung together; and a Realm which helps set up container-managed security for a particular container. Other nested components include the Loader which is used to enforce the specification's guidelines for servlet class loading; the Manager that supports session management for each web application; the Resources component that represents the web application's static resources and a mechanism to access these resources; and the Listener that allows you to insert custom processing at important points in a container's life cycle, such as when a component is being started or stopped. Not all nested components can be nested within every container. A final major component, which falls into its own category, is the Connector. It represents the connection end point that an external client (such as a web browser) can use to connect to the Tomcat container.



4.4.1 Architecture of Apache Tomcat

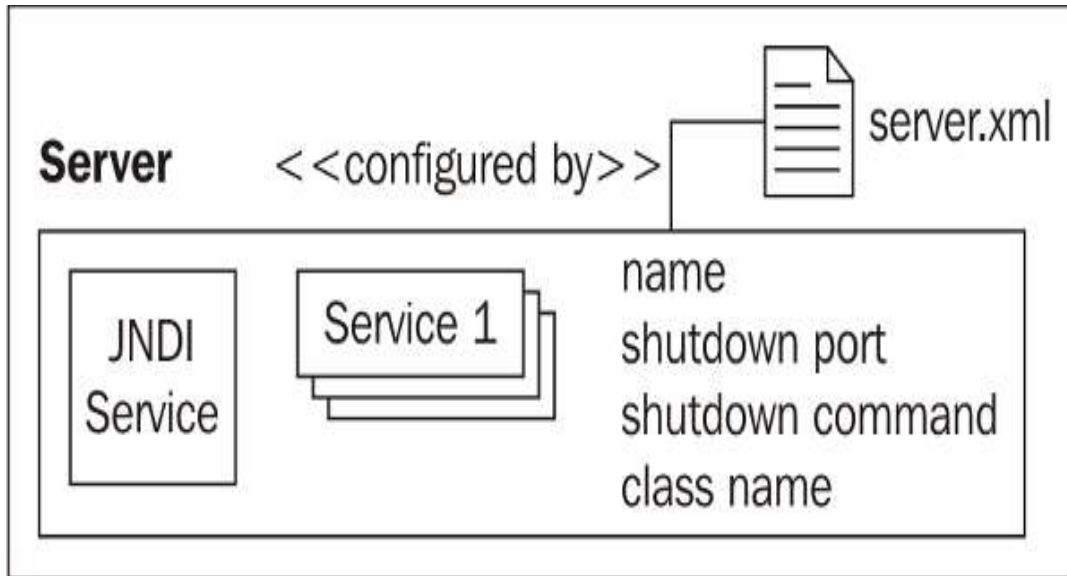
Architectural benefits:

This architecture has a couple of useful features. It not only makes it easy to manage component life cycles (each component manages the life cycle notifications for its children), but also to dynamically assemble a running Tomcat server instance that is based on the information that has been read from configuration files at startup. In particular, the server.xml file is parsed at startup, and its contents are used to instantiate and configure the defined elements, which are then assembled into a running Tomcat instance. The server.xml file is read only once, and edits to it will not be picked up until Tomcat is restarted. This architecture also eases the configuration burden by allowing child containers to inherit the configuration of their parent containers. For instance, a Realm defines a data store that can be used for authentication and authorization of users who are attempting to access protected resources within a web application. For ease of configuration, a realm that is defined for an engine applies to all its children hosts and contexts. At the same time, a particular child, such as a given context, may override its inherited realm by specifying its own realm to be used in place of its parent's realm.

Top Level Components:

The Server and Service container components exist largely as structural conveniences. A Server represents the running instance of Tomcat and contains one or more Service children, each of which represents a collection of request processing components.

Server: A Server represents the entire Tomcat instance and is a singleton within a Java Virtual Machine, and is responsible for managing the life cycle of its contained services. The following image depicts the key aspects of the Server component. As shown, a Server instance is configured using the server.xml configuration file. The root element of this file is <Server> and represents the Tomcat instance. Its default implementation is provided using org.apache.catalina.core.StandardServer, but you can specify your own custom implementation through the class Name attribute of the <Server> element.



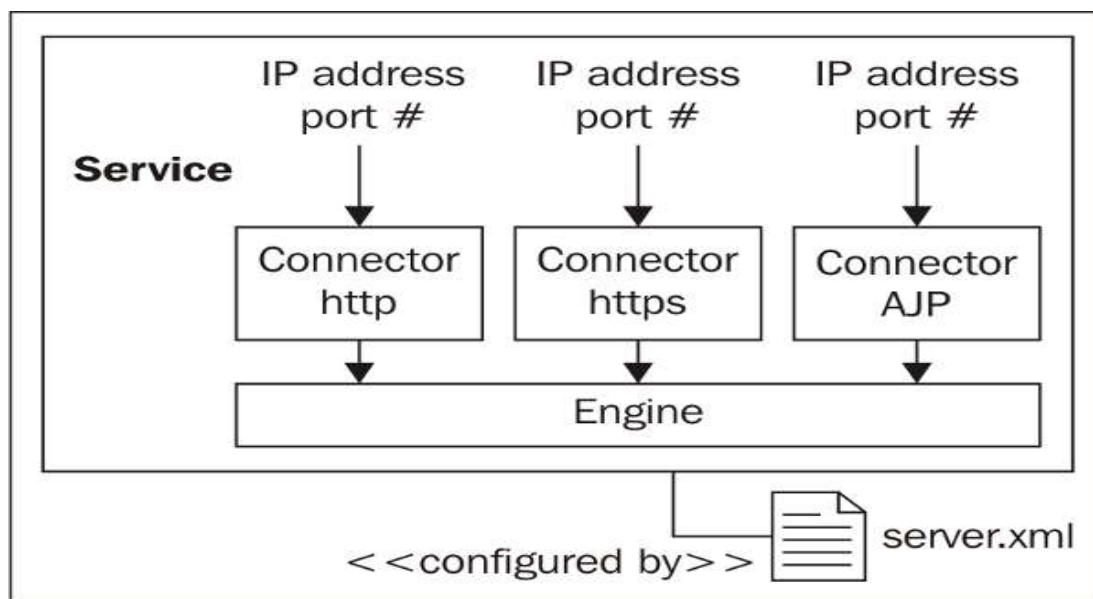
4.4.2 Working of Server

A key aspect of the **Server** is that it opens a server socket on port 8005 (the default) to listen a shutdown command (by default, this command is the text string **SHUTDOWN**). When this shutdown command is received, the server gracefully shuts itself down. For security reasons, the connection requesting the shutdown must be initiated from the same machine that is running this instance of Tomcat. A **Server** also provides an implementation of the Java Naming and Directory Interface (JNDI) service, allowing you to register arbitrary objects (such as data sources) or environment variables, by name. At runtime, individual components (such as servlets) can retrieve this information by looking up the desired object name in the server's JNDI bindings. While a JNDI implementation is not integral to the functioning of a servlet container, it is part of the Java EE specification and is a service that servlets have a right to expect from their application servers or servlet containers. Implementing this service makes for easy portability of web applications across containers. While there is always just one server instance within a JVM, it is entirely possible to have multiple server instances running on a single physical machine, each encased in its own JVM. Doing so insulates web applications that are running on one VM from errors in applications that are running on others, and simplifies maintenance by allowing a JVM to be restarted independently of the others. This is one of the mechanisms used in a shared hosting environment (the other is virtual hosting, which we

will see shortly) where you need isolation from other web applications that are running on the same physical server.

Service:

While the Server represents the Tomcat instance itself, a Service represents the set of request processing components within Tomcat. A Server can contain more than one Service, where each service associates a group of Connector components with a single Engine. Requests from clients are received on a connector, which in turn funnels them through into the engine, which is the key request processing component within Tomcat. The image shows connectors for HTTP, HTTPS, and the Apache JServ Protocol (AJP). There is very little reason to modify this element, and the default Service instance is usually sufficient.



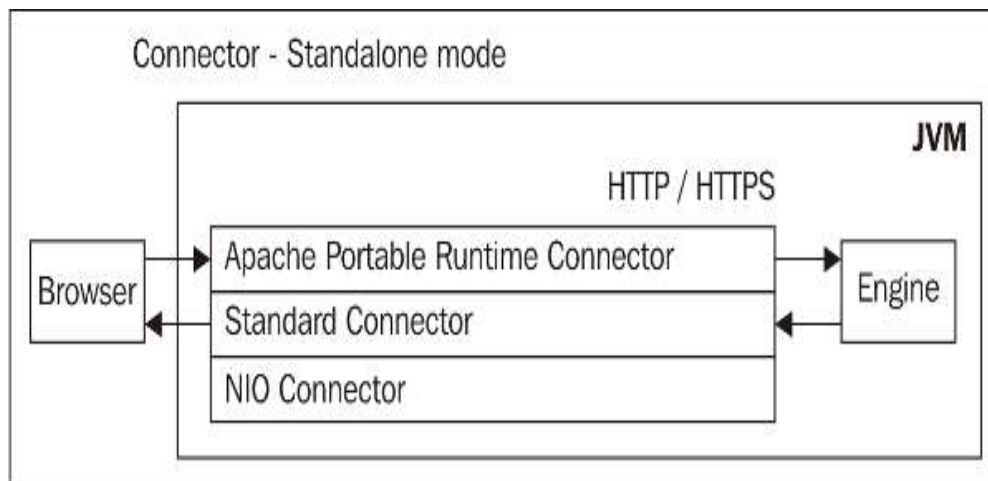
A hint as to when you might need more than one Service instance can be found in the above image. As shown, a service aggregates connectors, each of which monitors a given IP address and port, and responds in a given protocol. An example use case for having multiple services, therefore, is when you want to partition your services (and their contained engines, hosts, and web applications) by IP address and/or port number.

For instance, you might configure your firewall to expose the connectors for one service to an external audience, while restricting your other service to hosting intranet

applications that are visible only to internal users. This would ensure that an external user could never access your Intranet application, as that access would be blocked by the firewall. The Service, therefore, is nothing more than a grouping construct. It does not currently add any other value to the proceedings.

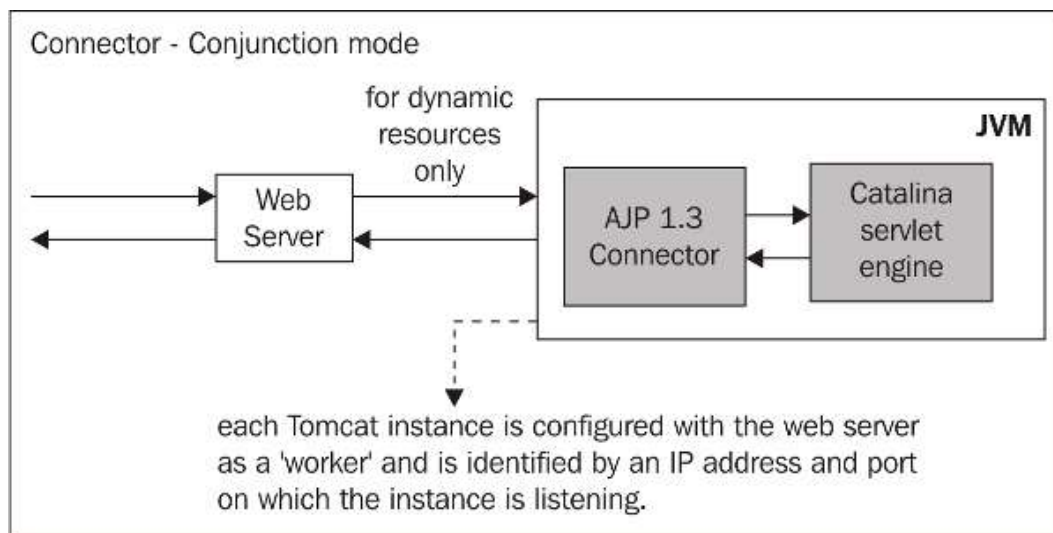
Connectors:

A Connector is a service endpoint on which a client connects to the Tomcat container. It serves to insulate the engine from the various communication protocols that are used by clients, such as HTTP, HTTPS, or the Apache JServ Protocol (AJP). Tomcat can be configured to work in two modes—Standalone or in Conjunction with a separate web server. In standalone mode,



Tomcat is configured with HTTP and HTTPS connectors, which make it act like a full-fledged web server by serving up static content when requested, as well as by delegating to the Catalina engine for dynamic content. Out of the box, Tomcat provides three possible implementations of the HTTP/1.1 and HTTPS connectors for this mode of operation. The most common are the standard connectors, known as Coyote which are implemented using standard Java I/O mechanisms. You may also make use of a couple of newer implementations, one which uses the non-blocking NIO features of Java 1.4, and the other which takes advantage of native code that is optimized for a particular operating system through the Apache Portable Runtime (APR). Note that both the Connector and the Engine run in the same JVM.

In fact, they run within the same Server instance. In conjunction mode, Tomcat plays a supporting role to a web server, such as Apache httpd or Microsoft's IIS. The client here is the web server, communicating with Tomcat either through an Apache module or an ISAPI DLL. When this module determines that a request must be routed to Tomcat for processing, it will communicate this request to Tomcat using AJP, a binary protocol that is designed to be more efficient than the text based HTTP when communicating between a web server and Tomcat. On the Tomcat side, an AJP connector accepts this communication and translates it into a form that the Catalina engine can process.



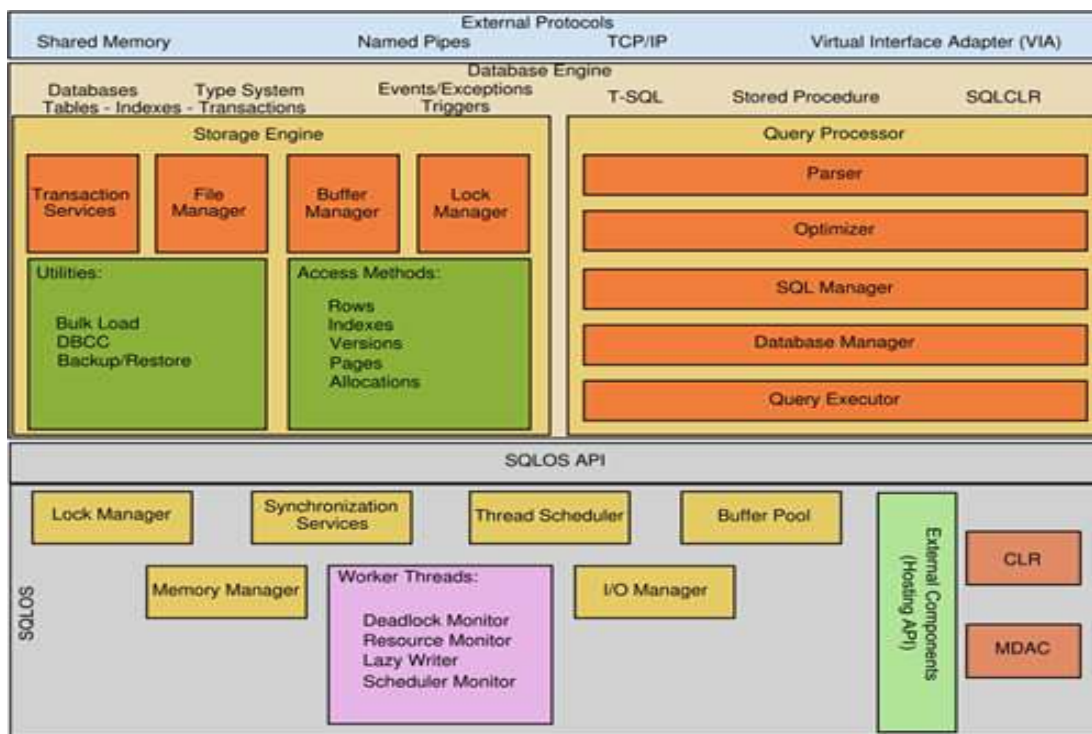
In this mode, Tomcat is running in its own JVM as a separate process from the web server. In either mode, the primary attributes of a Connector are the IP address and port on which it will listen for incoming requests, and the protocol that it supports. Another key attribute is the maximum number of request processing threads that can be created to concurrently handle incoming requests. Once all these threads are busy, any incoming request will be ignored until a thread becomes available. By default, a connector listens on all the IP addresses for the given physical machine (its address attribute defaults to 0.0.0.0). However, a connector can be configured to listen on just one of the IP addresses for a machine. This will constrain it to accept connections from only that specified IP address. Any request that is received by any one of a service's connectors is passed on to the service's single engine. This engine, known as Catalina, is responsible for the

processing of the request, and the generation of the response. The engine returns the response to the connector, which then transmits it back to the client using the appropriate communication protocol.

4.4 MYSQL Server

Microsoft MYSQL Server is a relational database management system developed by Microsoft. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). There are at least a dozen different editions of Microsoft MYSQL Server aimed at different audiences and for different workloads (ranging from small applications that store and retrieve data on the same computer, to millions of users and computers that access huge amounts of data from the Internet at the same time). True to its name, Microsoft MYSQL Server's primary query languages are T-MYSQL and ANSI MYSQL.

4.4.1 MYSQL Server Architecture Diagram:



For this purpose, it defined an xml data type that could be used either as a data type in database columns or as literals in queries.

XML columns can be associated with XSD schemas; XML data being stored is verified against the schema. XML is converted to an internal binary data type before being stored in the database. Specialized indexing methods were made available for XML data. XML data is queried using XQuery; MYSQL Server 2005 added some extensions to the T-MYSQL language to allow embedding XQuery queries in T-MYSQL. In addition, it also defines a new extension to XQuery, called XML DML, that allows query-based modifications to XML data. MYSQL Server 2005 also allows a database server to be exposed over web services using Tabular Data Stream (TDS) packets encapsulated within SOAP (protocol) requests. When the data is accessed over web services, results are returned as XML.

Common Language Runtime (CLR) integration was introduced with this version, enabling one to write MYSQL code as Managed Code by the CLR. For relational data, T-MYSQL has been augmented with error handling features (try/catch) and support for recursive queries with CTEs (Common Table Expressions). MYSQL Server 2005 has also been enhanced with new indexing algorithms, syntax and better error recovery systems. Data pages are check summed for better error resiliency, and optimistic concurrency support has been added for better performance. Permissions and access control have been made more granular and the query processor handles concurrent execution of queries in a more efficient way. Partitions on tables and indexes are supported natively, so scaling out a database onto a cluster is easier.

MYSQL CLR was introduced with MYSQL Server 2005 to let it integrate with the .NET Framework. MYSQL Server 2005 introduced "MARS" (Multiple Active Results Sets), a method of allowing usage of database connections for multiple purposes. MYSQL Server 2005 introduced DMVs (Dynamic Management Views), which are specialized views and functions that return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. Service Pack 1 (SP1) of MYSQL Server 2005 introduced Database Mirroring, a high availability option

that provides redundancy and failover capabilities at the database level. Failover can be performed manually or can be configured for automatic failover.

CHAPTER-5

SYSTEM TESTING

5.1. FEASIBILITY STUDY

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

Three key considerations involved in the feasibility analysis are

- ◆ ECONOMICAL FEASIBILITY
- ◆ TECHNICAL FEASIBILITY
- ◆ SOCIAL FEASIBILITY

5.1.1. ECONOMICAL FEASIBILITY

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

5.1.2. TECHNICAL FEASIBILITY

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

5.1.3. SOCIAL FEASIBILITY

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

5.2. SYSTEM TESTING AND MAINTENANCE

Testing is vital to the success of the system. System testing makes a logical assumption that if all parts of the system are correct, the goal will be successfully achieved. In the testing process we test the actual system in an organization and gather errors from the new system operates in full efficiency as stated. System testing is the

stage of implementation, which is aimed to ensuring that the system works accurately and efficiently.

In the testing process we test the actual system in an organization and gather errors from the new system and take initiatives to correct the same. All the front-end and back-end connectivity are tested to be sure that the new system operates in full efficiency as stated. System testing is the stage of implementation, which is aimed at ensuring that the system works accurately and efficiently.

The main objective of testing is to uncover errors from the system. For the uncovering process we have to give proper input data to the system. So we should have more conscious to give input data. It is important to give correct inputs to efficient testing.

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

This will create two problems, Time delay between the cause and appearance of the problem. The purpose of the system testing is to consider all the likely variations to which it will be suggested and push the system to its limits

The testing process focuses on logical intervals of the software ensuring that all the statements have been tested and on the function intervals (i.e.,) conducting tests to uncover errors and ensure that defined inputs will produce actual results that agree with the required results. Testing has to be done using the two common steps Unit testing and Integration testing. In the project system testing is made as follows:

The procedure level testing is made first. By giving improper inputs, the errors that occurred are noted and eliminated. This is the final step in the system life cycle. Here we implement the tested error-free system into a real-life environment and make necessary changes, which runs in an online fashion. Here system maintenance is done every month or year based on company policies and is checked for errors like runtime errors, long-run errors and other maintenances like table verification and reports.

5.3. UNIT TESTING

Unit testing verification efforts on the smallest unit of software design, module. This is known as “Module Testing”. The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps, each module is found to be working satisfactorily as regard to the expected output from the module.

5.4. INTEGRATION TESTING

Integration testing is a systematic technique for constructing tests to uncover error associated within the interface. In the project, all the modules are combined and then the entire programmer is tested as a whole. In the integration-testing step, all the error uncovered is corrected for the next testing steps.

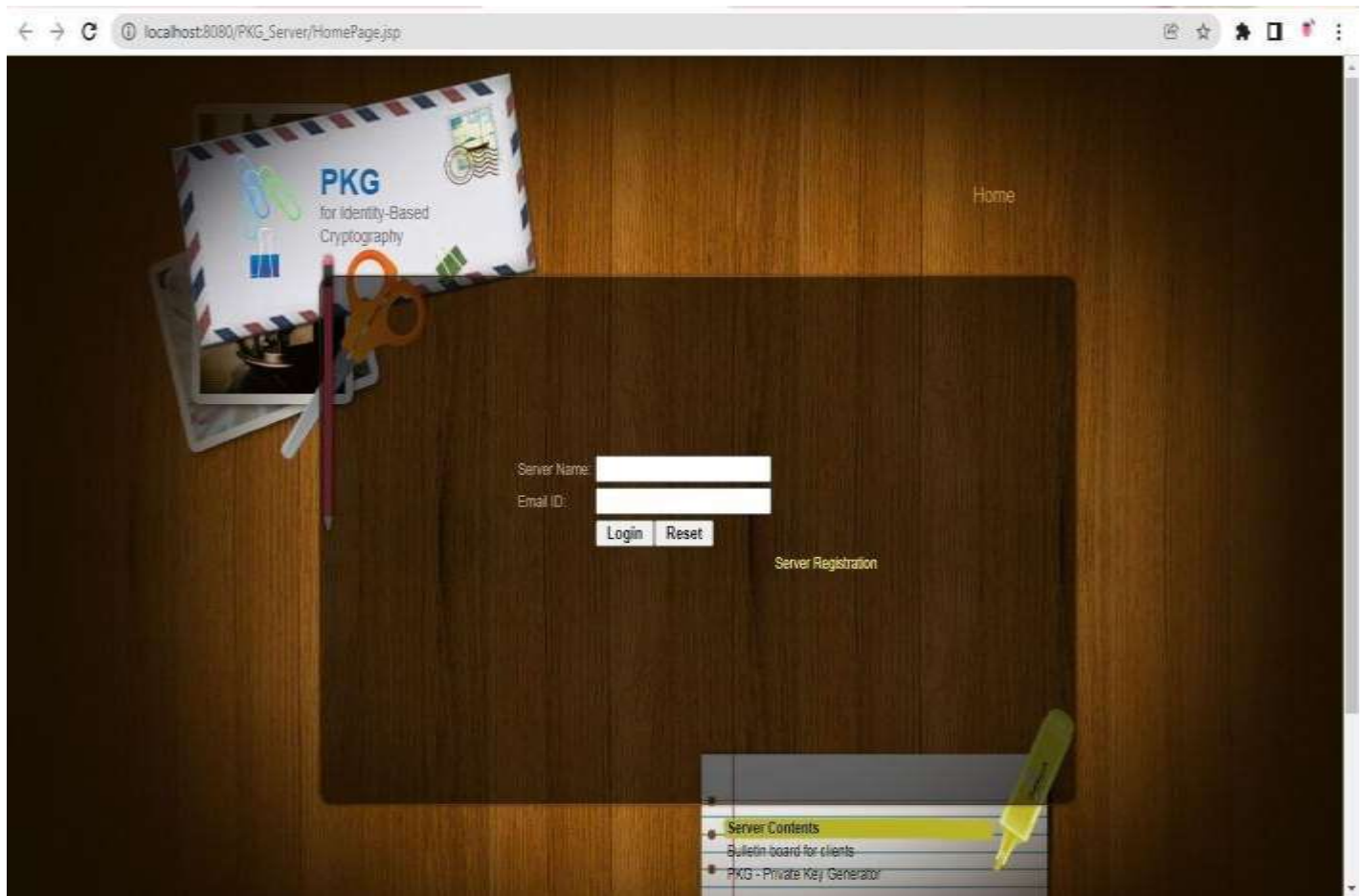
CHAPTER-6

CONCLUSION:

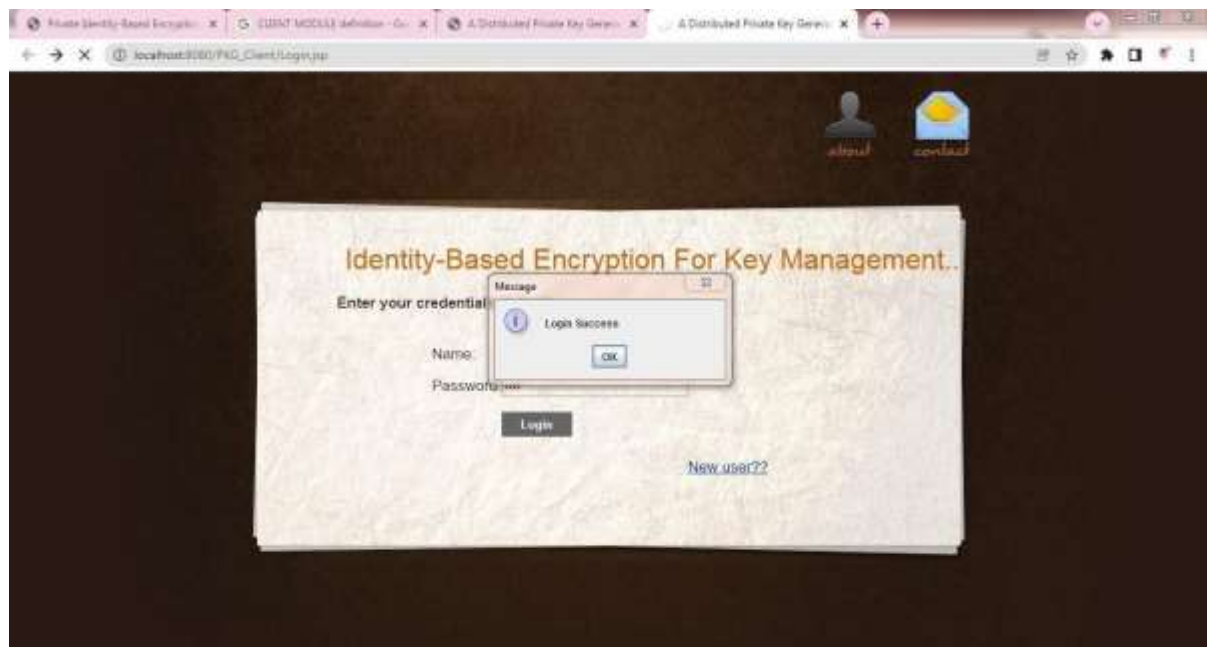
By combining the answers of a person to a questionnaire and XOR system of equations, we have shown how to build a safe and efficient generating identify key protocol. Based on this protocol, we also built secret storage and recovery application based on a decentralized storage system. Moreover, in this work, we present security scenarios to evaluate the security of the schemes we build based on the questionnaire. However, when the questions contain the person's personal information, such as "When was the last time you had cancer screening," or "How often do you do a lung exam a year", we need to hide go to each person's P . And in our protocol, by saving the hint set H_u , only users' u who know the correct answers to their individual questions P_u can rebuild the complete P_u set.

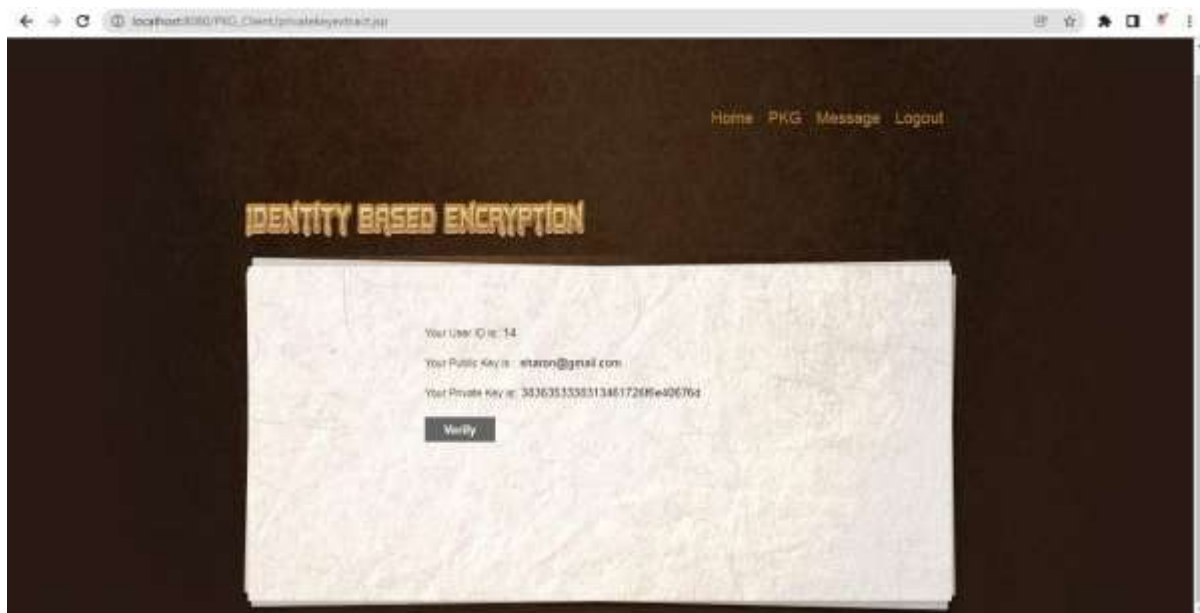
CHAPTER 7 : APPENDICES

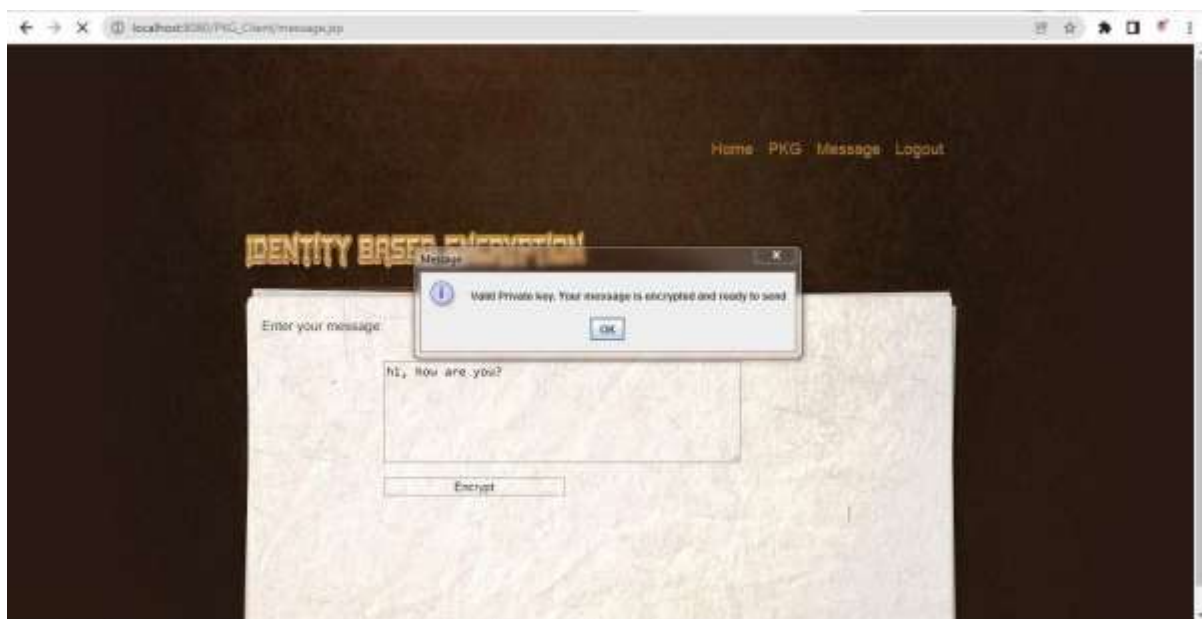
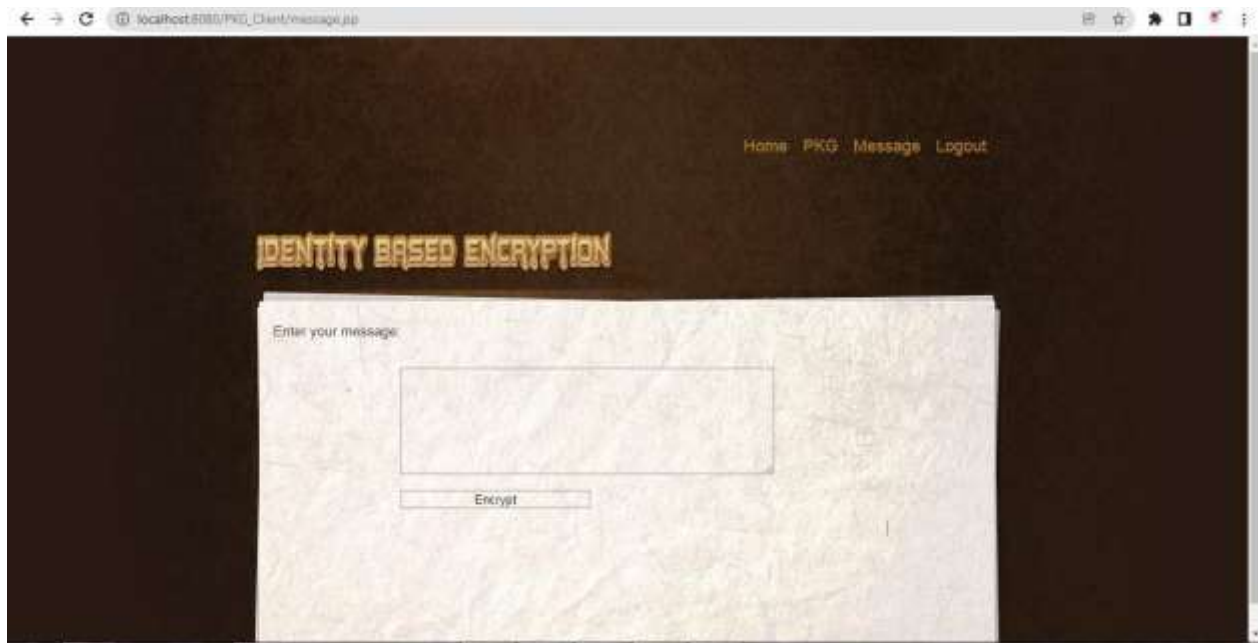
7.1 SCREENSHOTS:

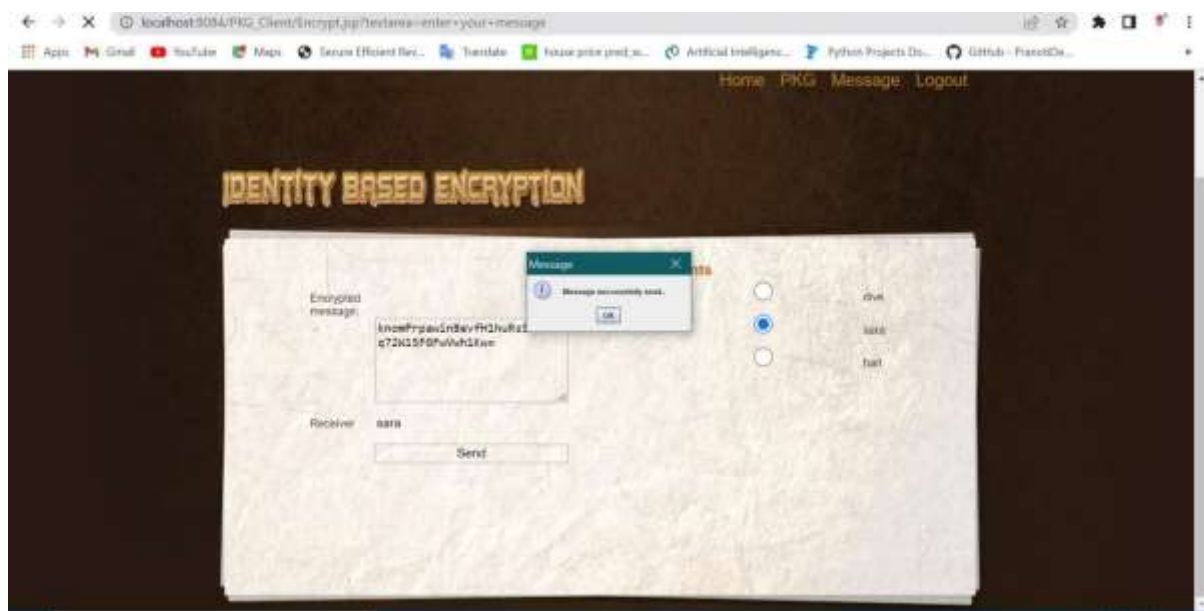
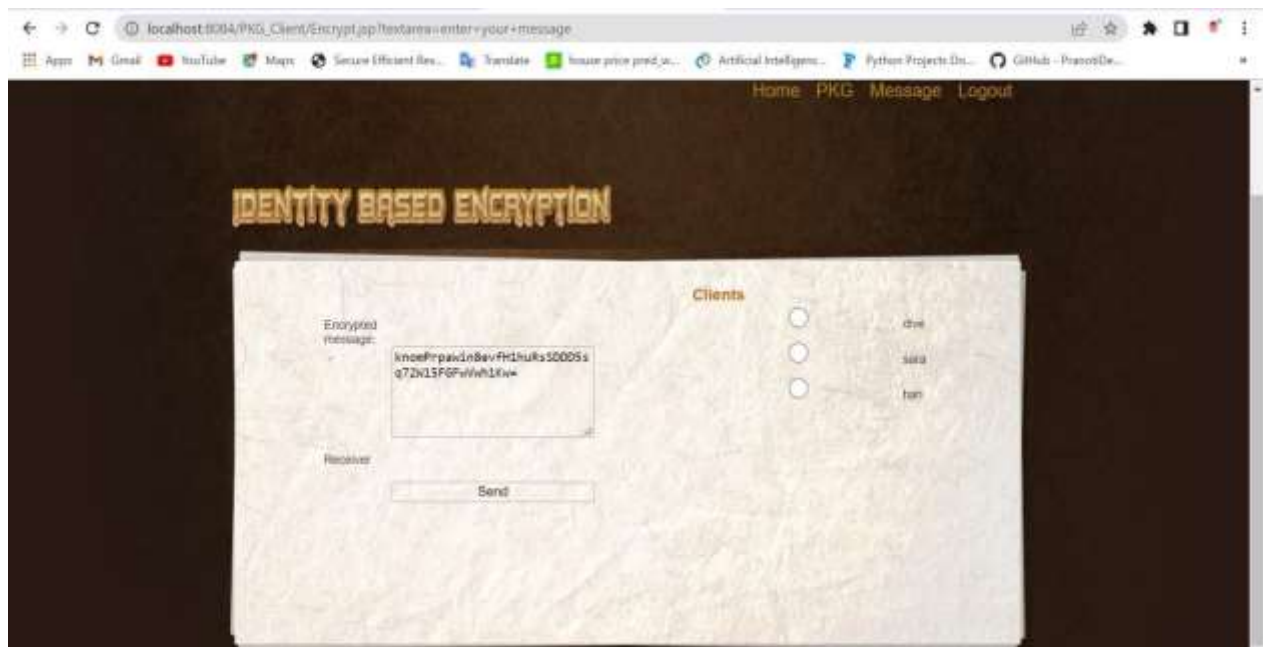


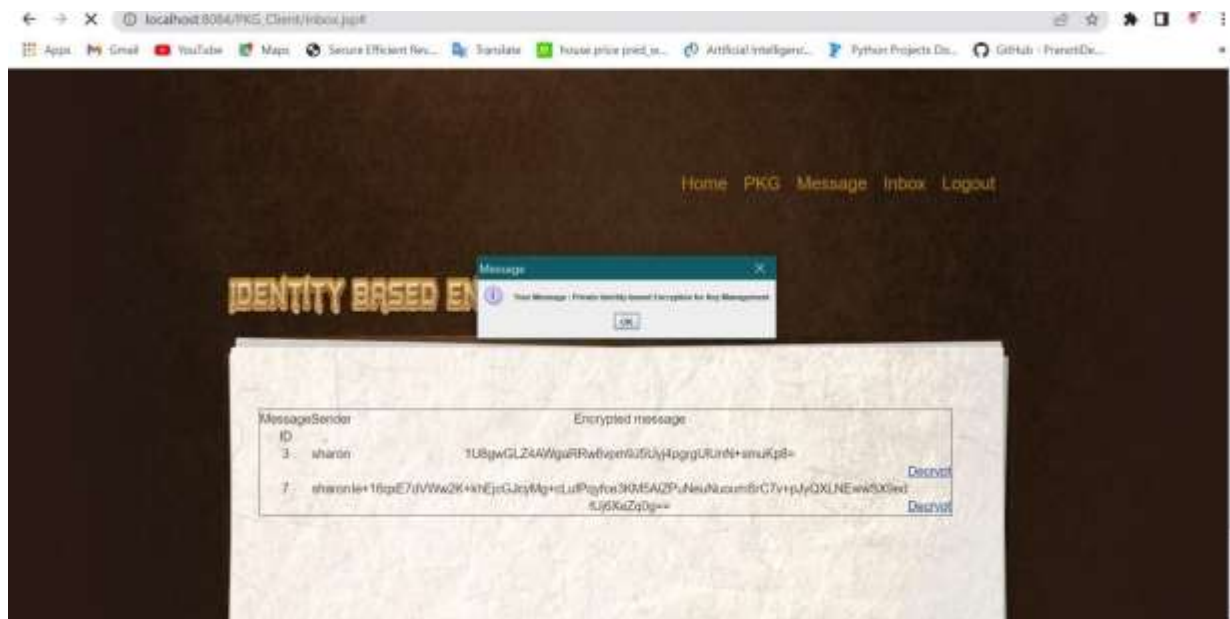












7.2 CODING:

```
String name=(String)session.getAttribute("name1");

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

        Connection                                dbConnection                                =
DriverManager.getConnection("jdbc:mysql://localhost:3306/pkg", "root", "admin");

        Statement statement = dbConnection.createStatement();

        String  query="Select  userid  from  userdetails  where
username=? ";

        PreparedStatement
check=dbConnection.prepareStatement(query);

        check.setString(1, name);

ResultSet rs=check.executeQuery();if(rs.next())

        {

        int id=rs.getInt(1);

JFrame frame = new JFrame();

        String result = JOptionPane.showInputDialog(frame, "Please enter your Private
key:");

        Connection                                connection                                =
DriverManager.getConnection("jdbc:mysql://localhost:3306/pkg", "root", "admin");

        Statement statement1 = connection.createStatement();

        PreparedStatement check2 = connection.prepareStatement("Select privatekey
from keydetails where privatekey=? and userid=? ");
check2.setString(1, result);
```

```

        check2.setInt(2,id);

        ResultSet rs2 = check2.executeQuery();

        if (rs2.next())
        {
            Cipher cipher;

            //retrieving content from text area

            String ques = request.getParameter("textarea");

            StringBuffer sb = new StringBuffer();

            sb.append(ques)

String ss = sb.toString(); Random rand
= new Random();String r = "";

        for (int j = 0; j < 1; j++) {

            long random = rand.nextLong();

            //System.out.println(random);

            long a=random%16;

            if(a==0)

            {

                r = Long.toString(random);

            }

            else

```

```

{
    random=random+a;

    r = Long.toString(random);

}

System.out.println("Random Value:"+r.substring(2, 18))
}

```

```

        AlgorithmParameterSpec      paramSpec      =      new
        IvParameterSpec(publickey.getBytes());

```

```

        cipher = Cipher.getInstance("AES/CBC/PKCS5Padding");
        cipher.init(Cipher.ENCRYPT_MODE, key, paramSpec);
        byte[] encrypted = cipher.doFinal(ss.getBytes());
        String output = new BASE64Encoder().encode(encrypted);
        System.out.println("Orginal String: " + ss);
        System.out.println("Encrypted string: " + output);

```

```

        JOptionPane.showMessageDialog(null,"Valid Private key. Your message is
        encrypted and ready to send");

```

```

%>

```

```

<!DOCTYPE html>

```

```

<html>

    <head>

        <meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">

        <title>A Distributed Private Key Generator for Identity Based
Cryptography</title>

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

        <link rel="stylesheet" type="text/css" href="lightbox.css" media="screen">

    <script>

        function dolt(obj)

        {

            if (obj.checked){

                obj.form.txt1.value += ((obj.form.txt1.value!="") ? '\r\n:') + obj.value;

            }

            else {

                obj.form.txt1.value = obj.form.txt1.value.replace(new RegExp("\n*" + obj.value,
"g"), "").replace(/^\s/, "");

            }

        }

    }

```

```

</script></head><body>

    <div id="pagewrap">

        <!-- Site header and navigation -->

            <div id="header">

<ul id="navigation">

<a href="HomePage.jsp" style="color: darkgoldenrod; text-decoration: none; font-size:
large">Home</a>



<a href="privatekeyextract.jsp" style="color: darkgoldenrod; text-decoration: none; font-
size: large">PKG</a>



<a href="logout.jsp" style="color: darkgoldenrod; text-decoration: none; font-size:
large">Logout</a>

</div>

<div id="content">

<div class="right">

<br></br>

<form action="sendingmessage.jsp">

```

```

<table>

<tr><td style="font-size: 20px">Encrypted message:</td><td></td></tr>

<tr><td><textarea cols="40" rows="10" id="getrow" name="textarea" style=" margin-
top:      30px;height:      110px;width:      400px;font-size:      15px"><%
out.println(output.trim());%></textarea></td></tr>

</table>



</td>

<td align="right" style="text-align: start">

<%

}

    else

    {

        JOptionPane.showMessageDialog(null,"Invalid Private key");

        response.sendRedirect("HomePage.jsp");

    }

%>

</table>

</form>

</body>

</

```

Chapter -8

REFERENCES:

- [1] M. Abe and S. Fehr. Adaptively Secure Feldman VSS and Applications to Universally-Composable Threshold Cryptography. In *Advances in Cryptology—CRYPTO'04*, pages 317–334, 2004.
- [2] S. S. Al-Riyami and K. G. Paterson. Certificateless Public Key Cryptography. In *Advances in Cryptology—ASIACRYPT'03*, pages 452–473, 2003.
- [3] Y. Amir, C. Nita-Rotaru, J. R. Stanton, and G. Tsudik. Secure Spread: An Integrated Architecture for Secure Group Communication. *IEEE Transactions on Dependable and Secure Computing*, 2(3):248–261, 2005.
- [4] G. R. Blakley. Safeguarding cryptographic keys. In *the National Computer Conference*, pages 313–317, 1979.
- [5] D. Boneh, X. Boyen, and E. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Advances in Cryptology—EUROCRYPT'05*, pages 440–456, 2005.
- [6] D. Boneh and M. K. Franklin. Identity-Based Encryption from the Weil Pairing. In *Advances in Cryptology—CRYPTO'01*, pages 213–229, 2001.
- [7] C. Cachin, K. Kursawe, A. Lysyanskaya, and R. Stroh. Asynchronous verifiable secret sharing and proactive cryptosystems. In *ACM Conference on Computer and Communications Security*, pages 88–97, 2002.
- [8] C. Cachin and J. A. Poritz. Secure intrusion-tolerant replication on the internet. In *DSN*, pages 167–176, 2002.
- [9] R. Canetti. *Studies in Secure Multiparty Computation and Applications*. PhD thesis, The Weizmann Institute of Science, 1996.

[10] R. Canetti, R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Adaptive Security for Threshold Cryptosystems. In *Advances in Cryptology—CRYPTO'99*, pages 98–115, 1999.