**TWO BIRDS WITH ONE STONE : TWO FACTOR**

**AUTHENTICATION WITH SECURITY BEYOND**

**CONVENTIONAL BOUND**

A Project report Submitted to

##### PACE INSTITUTE OF TECHNOLOGY AND SCIENCES (AUTONOMOUS)

*in partial fulfillment of the requirement For the award of Degree of*

##### BACHELOR OF TECHNOLOGY

**IN INFORMATION TECHNOLOGY**

##### DURING 2019-2023

**SUBMITTED BY**

|  |  |  |
| --- | --- | --- |
| **(19KQ1A1211)** |  | **KANAGALA VENKATA SIVA BHARGAVI** |
| **(19KQ1A1227)**  **(19KQ1A1241)** |  | **BHASKARUNI SAISAKETH**  **MEDA VENKATA DHANUSH** |
| **(19KQ1A1249)** |  | **PATAPANTHULA VAMSI** |
| **(20KQ5A1204)** |  | **VARAKALA SRIHARI** |
|  | **Submitted to** |  |

***Under the Esteemed Guidance of***

**Mr. K. MANOHARA RAO**

**Assistant Professor, M. Tech Department of IT.**



**DEPARTMENT OF INFORMATION TECHNOLOGY**

PACE INSTITUTE OF TECHNOLOGY & SCIENCES

(ApprovedbyA.I.C.T.E.,NewDelhi & Govt of Andhra Pradesh,Affiliated to JNTU Kakinada)

**ACCREDITED BY NAAC WITH ‘A’ GRADE & NBA(An ISO 9001:2008 Certified Institution)**

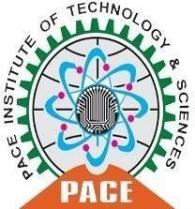
NH-5, Near Valluramma Temple,ONGOLE-523272,Contact No:08592- **2019-2023**

##### PACE INSTITUTE OF TECHNOLOGY AND SCIENCES

(Approved by A.I.C.T.E., New Delhi & Govt of Andhra Pradesh, Affiliated to JNTU Kakinada)

**ACCREDITED BY NAAC WITH ‘A’ GRADE & NBA (An ISO 9001:2008 Certified Institution)**

NH-5,Near Valluramma Temple,ONGOLE-523272,Contact No:08592201167[,www.pace.ac.in](http://www.pace.ac.in/)



**CERTIFICATE**

**This is to certify that the project entitled “Two Birds with one Stone :Two-factor authentication with security beyond conventional bound” is a bonafide work done by**

**19KQ1A1211 KANAGALA VENKATA SIVA BHARGAVI**

**19KQ1A1227 BHASKARUNI SAKETH**

**19KQ1A1241 MEDA VENKATA DHANUSH**

**19KQ1A1249 PATAPANTHULA VAMSI**

**20KQ5A1204 VARAKALA SRIHARI**

##### PACE INSTITUTE OF TECHNOLOGY AND SCIENCES (AUTONOMOUS)

*in partial fulfillment of the requirement For the award of Degree of*

##### BACHELOR OF TECHNOLOGY

**IN INFORMATION TECHNOLOGY**

##### DURING 2019-2023

**Internal Guide Head of Department**

**Mr. K. Manohara Rao M.Tech. Dr A.Seshagiri Rao M.Tech.,Ph.D**

Assistant Professor Associate Professor & HOD

Department of IT. Department of IT.

**External Examiner**

**DECLARATION**

We are here to declare that the project report titled **“ Two Birds with one Stone :Two-factor authentication with security beyond conventional bound ”** is a genuine work carried out by us under the guidance **of Mr. K. Manohara Rao M.Tech.,** Assistant Professor of Information Technology, is submitted in partial fulfillment of the requirements for the award of the Degree of Bachelor of Technology in Information Technology. This is a record of bonafied work carried out by us and the results embodied in this project report have not been reproduced or copied from any source. The results embodied in this project report have not been submitted to any other University or Institute for the award of any other degree or diploma.

**KANAGALA VENKATA SIVA BHARGAVI (19KQ1A1211)**

**BHASKARUNI SAI SAKETH (19KQ1A1227)**

**MEDA VENKATA DHANUSH (19KQ1A1241)**

**PATAPANTHULA VAMSI (19KQ1A1249)**

**VARAKALA SRIHARI (20KQ5A1204)**

### ACKNOWLEDGEMENT

We are ineffably indebted to our honorable Secretary & Correspondent **Er.M.Sridhar, B.E., M.Tech** and our beloved Principal **Dr.G.V.K Murthy , M.S, Ph.D,** for their kind support and help.

We take this opportunity to express our deep respect and profound gratitude to our esteemed guide **Mr.K.Manohara Rao, M.Tech, Assistant Professor, Information Technology , PACE INSTITUTE OF TECHNOLOGY & SCIENCES Vallur,** for his exemplary guidance, monitoring and constant encouragement throughout the course of this project.

We would like to extend our sincere and heartfelt thanks to **Dr.A.Seshagiri Rao, M.Tech., Ph.D. Head of Information Technology Department,** who has given valuable advice and suggestions and has encouraged us by providing all the facilities necessary to complete this project successfully.

We thankful to all the **Professors and Faculty Members** in the department for their teachings and academic support and thanks to **Technical Staff and Non-Teaching Staff** in the department for their support.

We also acknowledge with a deep sense of reverence, our gratitude towards our parents, members of our family and friends for their constant support and encouragement.

**KANAGALA VENKATA SIVA BHARGAVI (19KQ1A1211)**

**BHASKARUNI SAISAKETH (19KQ1A1227)**

**MEDA VENKATA DHANUSH (19KQ1A1241)**

**PATAPANTHULA VAMSI (19KQ1A1249)**

**VARAKALA SRIHARI (20KQ5A1204)**

### CONTENTS

|  |  |
| --- | --- |
| **ABSTRACT LITERATURESURVEY ACRONYMS AND ABBREVATION**  **1.INTRODUCTION** | **I**  **II III**  **IV** |
| 1.1 Introduction | 01 |
| 1.2 Objective | 03 |
| 1.3 Our contributions | 03 |
| 1.4 Existing system | 04 |
| 1.5 Proposed system | 05 |
|  |  |
| **2. LIERATURE SURVEY** | **07** |
| 2.1 A Strong and Efficient Certificate less Digital Signature Scheme | 08 |
| 2.2 Authenticating Secret Key-Exchange Between Two Parties. | 08 |
| 2.3 Payments for Outsourced Computation. | 09 |
| 2.4 Identity-Based Encryption from the Weil Pairing | 09 |
| 2.5 Authenticating Secret Key-Exchange Between Two Parties | 10 |
| **3.SOFTWARE ENVIRONMENT** | **11** |
| 3.1 Use case diagram | 12 |
| 3.2 Class diagram | 13 |
| 3.3 sequence diagram | 14 |
| 3.4 Collaboration diagram | 15 |
| 3.5 Activity diagram | 16 |
| 3.6 Data flow diagram | 17 |

1. [SOFTWARE DESCRIPTION 19](#_bookmark0)

4.1 [Java Technology 20](#_bookmark1)

4.2 [The java programming language](#_bookmark2) 21

4.3 [The java platform 21](#_bookmark3)

4.4 [Networking 26](#_bookmark4)

4.5 TCS/ IP stack. 27

* 1. [JDBC Goals. 30](#_TOC_250007)

5.[DETAILED DESIGN 33](#_bookmark5)

5.1 Pseudo code description. 34

5.2 [Login 34](#_TOC_250006)

5.3 [File transmission. 34](#_TOC_250005)

5.4 [Session key generation 35](#_TOC_250004)

5.5 [Hacking zone Monitoring. 36](#_TOC_250003)

5.6 [Authentication 36](#_TOC_250002)

5.7 Architecture 37

* 1. [Modules 38](#_TOC_250001)

6.CODING AND TESTING 40

6.1Testing 41

6.2[Process 41](#_bookmark6)

6.3Types of testing 41

Unit Testing 41

Integration Testing 42

[Functional Testing 42](#_bookmark8)

[System Testing 43](#_bookmark9)

[White box testing 43](#_bookmark10)

[Black box testing 43](#_bookmark11)

[Unit testing 43](#_bookmark12)

[Test Strategy and approach 44](#_TOC_250000)

[System Implementation 45](#_bookmark13)

[Feasibility study 46](#_bookmark14)

[Economical Feasibility. 46](#_bookmark15)

7.[JAVA CODES 47](#_bookmark16)

7.1 KGC 47

7.2 Broadcast Key 59

7.3 File protection 60

7.4 Hash Algorithm 63

**8.PROGRAM EXECUTION DIAGRAMS 64**

**9.BIBILIOGRAPHY** 75

### ABSTRACT

Key-exchange, in particular Diffie–Hellman key exchange (DHKE), is among the core cryptographic mechanisms for ensuring network security. For key-exchange over the Internet, both security and privacy are desired. In this paper, we develop a family of privacy-preserving authenticated DHKE protocols named deniable Internet key-exchange (DIKE), both in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity and practical (online) efficiency. They provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard. To the best of our knowledge, our protocols are the first provably secure DHKE protocols that additionally enjoy all the following privacy protection advantages: 1) forward deniability, actually concurrent non- malleable statistical zero-knowledge, for both protocol participants simultaneously; 2) the session transcript and session-key can be generated merely from DH-exponents (together with some public values), which thus cannot be traced to the pair of protocol participants; and 3) exchanged messages do not bear peer’s identity, and do not explicitly bear player role information.

### LIST OF FIGURES

|  |  |  |
| --- | --- | --- |
| **Fig No** | **Figure Name** | **Page No** |
| Fig:3.2 | Architecture | 11 |
| Fig:3.2.1 | DFD | 13 |
| Fig:5.1 | ER | 14 |
| Fig:5.2.1 | Use Case Diagram | 23 |
| Fig:5.2.2 | Class Diagram | 28 |
| Fig:5.2.3 | Sequences Diagram | 29 |
| Fig:5.2.4 | Collaboration Diagram | 31 |
| Fig:5.3 | Activity Diagram | 32 |
| Fig:8.1.1 | Data flow Diagram | 34 |
| Fig:8.1.2 | Code compilation | 35 |
| Fig:8.1.3 | JDBC | 38 |
| Fig:8.1.4 | TCP/IP Stack | 40 |
| Fig:8.1.5 | User Login form | 70 |
| Fig:8.1.6 | User New Register | 70 |
| Fig:8.1.7 | User sign up | 71 |
| Fig:8.1.8 | User Nodes | 72 |
| Fig:8.1.9 | File transfer | 73 |
| Fig:8.1.10 | Authentication Form | 74 |
| Fig:8.1.10 | Transmission Form | 75 |
| Fig:8.1.11 | Receiver Form | 76 |

**ACRONYMS AND ABBREVIATIONS**

|  |  |
| --- | --- |
| DHKE | Diffie Hellman key exchange |
| DIKE | Deniable Internet key exchange |
| IKE | Internet key exchange |
| IPE | Internet Protocol exchange |
| IP Sec | Internet Protocol Security |
| IPEv1 | Internet Protocol Exchange v1 |
| IPEv2 | Internet Protocol Exchange v2 |
| KGC | Key generation compiler |
| PKI | Public Key Infrastructure |
| RO | Random Oracle |
| DFD | Data Flow Diagram |
| ER | Entity Relationship |
| TCP/IP | Transport Control Protocol/Internet Protocol |

**CHAPTER - 1**

# INTRODUCTION

##### INTRODUCTION

**T**HE Internet Key-Exchange (IKE) protocols are the *core* cryptographic protocols to ensure Internet security, which specify key exchange mechanisms used to establish shared keys for use in the Internet Protocol Security (IPsec) standards. The IPsec and IKE are intended to protect messages communicated in the IP layer, i.e., “layer 3” of ISO-OSI, which process the transmission of messages using the network addresses *possibly without knowing end-user peers’ identities*. The IKE and IPsec can in turn be used to offer confidentiality, authentication and privacy for communication protocols in the higher layers of ISO-OSI. The standard of IKE has gone through two generations. The first generation IKEv1 uses public-key encryption as the authentication mechanism. The second generation IKEv2 uses signatures as the authentication mechanism, with the SIGMA protocol serving as the basis. The IKEv2 protocol is based on DHKE, and works in the “post-specified peer” setting, where the information of who the other party is does not necessarily exist at the session initiation stage and is learnt by the party only after the protocol run evolves (even just in the last round). Actually, this is quite a common case for KE protocols in practice, particularly for the purpose of preserving players’ privacy. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation; Or, a party may respond to a request (for a KE session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself .

**Objective:**

Key-exchange, in particular Diffie–Hellman key exchange (DHKE), is among the core cryptographic mechanisms for ensuring network security. For key-exchange over the Internet Key, both security and privacy are desired. In this project, we develop a family of privacy-preserving authenticated DHKE protocols named deniable Internet key-exchange (DIKE), both in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity and practical (online) efficiency. They provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard.

##### Our Contributions:

In this work, develop a family of privacy-preserving (particularly, *deniable*) authenticated DHKE protocols, named deniable Internet key-exchange (DIKE), in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity, practical (online) efficiency, provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard and the SIGMA protocol. The security of DIKE is analyzed in accordance with the Canetti-Krawczyk framework (CK-framework) *with post-specified peers* in the random oracle (RO) model. We also make discussions on a list of concrete yet essential security properties of DIKE, *most of which are beyond the CK-framework*. We then define CNMSZK for DHKE , along with detailed clarifications and justifications. To our knowledge, our formulation of CNMSZK for DHKE stand for the strongest definition of deniability, to date, for key-exchange protocols. The CNMSZK property of our protocols is analyzed in the restricted random oracle model, under an extension of the knowledge- of-exponent assumption named concurrent knowledge-of-exponent (CKEA) that might be of independent interest.

##### Existing System:

The standard of IKE has gone through two generations. The first generation IKEv1 uses public-key encryption as the authentication mechanism. The second generation IKEv2 uses signatures as the authentication mechanism, with the SIGMA protocol serving as the basis. The IKEv2 protocol is based on DHKE [20], and works in the “post-specified peer” setting, where the information of who the other party is does not necessarily exist at the session initiation stage and is learnt by the party only after the protocol run evolves (even just in the last round). Actually, this is quite a common case for KE protocols in practice, particularly for the purpose of preserving players’ privacy. For example, the key-exchange session may take place with any one of a set of servers sitting behind a (url/ip) address specified in the session activation; Or, a party may respond to a request (for a KE session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself (e.g., a roaming mobile user connecting from a temporary address, or a smart-card that authenticates the legitimacy of the card reader before disclosing its own identity). For key- exchange protocols, both security and privacy are desired. Actually, providing a certain level of privacy protection serves as one of the major criteria underlying the evolution of a list of important industrial standards of KE protocols, which is particularly witnessed by the evolution of IKE that is based on the SIGMA protocol.

##### Disadvantages:

A party may respond to a request (for a KE session) coming from a peer that is not willing to reveal its identity over the network and, sometimes, even not to the responder before the latter has authenticated itself.

* A session transcript, the malicious verifier cannot prove that the honest prover was ever involved in the conversation.
* we would like to happen is that if the prover acts honestly during the protocol, it also should not be able at a later stage to claim the messages are authentic in order to violate the privacy of the verifier.

##### Proposed System:

In this work, develop a family of privacy-preserving (particularly, deniable) authenticated DHKE protocols, named deniable Internet key-exchange (DIKE), in the traditional PKI setting and in the identity-based setting. The newly developed DIKE protocols are of conceptual clarity, practical (online) efficiency, provide useful privacy protection to both protocol participants, and add novelty and new value to the IKE standard and the SIGMA protocol. The security of DIKE is analyzed in accordance with the Canetti-Krawczyk framework (CK-framework) with post-specified peers in the random oracle (RO) model. We also make discussions on a list of concrete yet essential security properties of DIKE, most of which are beyond the CK-framework. We then define CNMSZK for DHKE , along with detailed clarifications and justifications. To our knowledge, our formulation of CNMSZK for DHKE stand for the strongest definition of deniability, to date, for key- exchange protocols.

##### Advantages:

* 1. Forward deniability, actually concurrent non-malleable statistical zero-knowledge, for both protocol participants simultaneously.
  2. The session transcript and session-key can be generated merely from DH-exponents (together with some public values), which thus cannot be traced to the pair of protocol participants.
  3. Exchanged messages do not bear peer’s identity, and do not explicitly bear player role information.

##### System Requirements:

**Hardware Requirements:**

* Processor : Any Processor above 1.2 GHz
* Ram : 1Gb.
* Hard Disk : 40 GB.
* Compact Disk : 650 Mb.
* Input device : Standard Keyboard and Mouse.
* Output device : VGA and High Resolution Monitor.

##### Software Requirements:

* Operating System : Windows Family.
* Language : JAVA.
* Front End : Swing Framework.
* Database : My Sql.

CHAPTER - 2

# LITERATURE SURVEY

#### 2.LITERATURE SURVEY

##### 2.1 A Strong and Efficient Certificate less Digital Signature Scheme

This paper extends the certificateless public key infrastructure model that was proposed by Hassouna et al by proposing new digital signature scheme to provide true non- repudiation, the proposed signature scheme is short and efficient, it is also has strength point that the KGC has no contribution in signature generation/verification process, therefore any compromise of the KGC does not affect the non-repudiation service of the system. Furthermore, even the KGC cannot do signature forgery by (temporary) replacing the user’s public key

##### 2.2 Authenticating Secret Key-Exchange Between Two Parties.

Key-exchange, in particular Diffie–Hellman key exchange (DHKE), is among the core cryptographic mechanisms for ensuring network security. For key-exchange, both security and privacy are desired. The proposed scheme is used to exchange the secret key by implementing the Shamir secret sharing scheme. The secret can be splitted into shares based on the threshold values, it refers to way for distributing a secret amongst a discussion of sender and receiver and the other parties include in transaction. The secret key will be reconstructed solely through equal number of shares. Individual shares are of no use on their own. Using Shamir Secret sharing, we have been enhance the approach for this present work, that it will reconstruct the secret key through the procedure of dividing into parts that is given into the each specified participants on own unique part, and some of the parts are need that to be reconstruct the secret key only the receiver verified based on their unique identity. Then the receiver receives the key of all the parts and decryption is made to message. In this work, introduce an new approach of sharing message in the way of authentication, it will produce more authentication than other secret

sharing methods, because the reconstruction of key is determined and the message authentication is undetermined to hackers or illegal users.

##### Payments for Outsourced Computation

With the recent advent of cloud computing, the concept of outsourcing computations, initiated by volunteer computing efforts, is being revamped. While the two paradigms differ in several dimensions, they also share challenges, stemming from the lack of trust between outsourcers and workers. In this work, we propose a unifying trust framework, where correct participation is financially rewarded: neither participant is trusted, yet outsourced computations are efficiently verified {and} validly remunerated. We propose three solutions for this problem, relying on an offline bank to generate and redeem payments; the bank is oblivious to interactions between outsourcers and workers. We propose several attacks that can be launched against our framework and study the effectiveness of our solutions. We implemented our most secure solution and our experiments show that it is efficient: the bank can perform hundreds of payment transactions per second and the overheads imposed on outsourcers and workers are negligible.

##### Identity-Based Encryption from the Weil Pairing

We propose a fully functional identity-based encryption scheme (IBE). The scheme has chosen ciphertext security in the random oracle model assuming a variant of the computational Diffie-Hellman problem. Our system is based on bilinear maps between groups. The Weil pairing on elliptic curves is an example of such a map. We give precise definitions for secure identity based encryption schemes and give several applications for such systems.

##### Authenticating Secret Key-Exchange Between Two Parties

Key-exchange, in particular Diffie–Hellman key exchange (DHKE), is among the core cryptographic mechanisms for ensuring network security. For key-exchange, both security and privacy are desired. The proposed scheme is used to exchange the secret key by implementing the Shamir secret sharing scheme. The secret can be splitted into shares based on the threshold values, it refers to way for distributing a secret amongst a discussion of sender and receiver and the other parties include in transaction. The secret key will be reconstructed solely through equal number of shares. Individual shares are of no use on their own. Using Shamir Secret sharing, we have been enhance the approach for this present work, that it will reconstruct the secret key through the procedure of dividing into parts that is given into the each specified participants on own unique part, and some of the parts are need that to be reconstruct the secret key only the receiver verified based on their unique identity. Then the receiver receives the key of all the parts and decryption is made to message. In this work, introduce an new approach of sharing message in the way of authentication, it will produce more authentication than other secret sharing methods, because the reconstruction of key is determined and the message authentication is undetermined to hackers or illegal users.

CHAPTER - 3

**SOFTWARE ENVIRONMENT**

##### 3.SOFTWARE ENVIRORMENT DIAGRAMS

**Use case Diagram:**

A use case is a set of scenarios that describing an interaction between a user and a system. A use case diagram displays the relationship among actors and use cases. The two main components of a use case diagram are use cases and actors. An actor is represents a user or another system that will interact with the system modeled. A use case is an external view of the system that represents some action the user might perform in order to complete a task.



request

Security Key

user

KGC

Session Key

Encryption & Decryption

##### Class Diagram:

Class diagrams are the mainstay of object-oriented analysis and design. Class diagrams show the classes of the system, their interrelationships (including inheritance, aggregation, and association), and the operations and attributes of the classes. Class diagrams are used for a wide variety of purposes, including both conceptual/domain modeling and detailed design modeling.



requestfromUser() responsetoReceiver() getSecret() selectSession() generationFunction() broadcast()

getUserLift() createChannel() broadcast()

String session int sessionkey int hashValue

String sender String receiver int secretkey int sessionkey int hashvalue

KeyDistribution

KeyGeneration

checkUserDetails() genKey() forwardUser()

get prims() genKey()

String uname String ipadd String port int secretkey

int p int q

int secretKey

UserRegistration

Initialization

Sequence Diagram:

User1 KGC

User1

KGC

User2

User3

User3

User2

intialization

get Secret Key

request to session key

Response

Response

Give Secret Key

Give Secret Key

Select Session Key

Generate hash Function

Encryption Broadcast Key

Broadcast Key

Broadcast Key

Decryption

Decryption

Decryption

##### Collaboration Diagram:

16: Encryption

Intialization 1:

3: Request

8:

9:

10:

Select Session Key

Hash function

KGC

5: Response

Get Secret Key

2:

12:

Give Secret Key 7:

Response

Give Secret Key

6:

15: Encryption

4:

11:

User2

User1

14:

User3

13:

Encryption

##### Activity Diagram:

Activity diagrams are typically used for business process modeling, for modeling the logic captured by a single use case or usage scenario, or for modeling the detailed logic of a business rule. Although UML activity diagrams could potentially model the internal logic of a complex operation it would be far better to simply rewrite the operation so that it is simple enough that you don’t require an activity diagram. In many ways UML activity diagrams are the object-oriented equivalent of flow charts and data flow diagrams (DFDs) from structured development.



Member

Register new Members with KGC

KGC

*Check Avaailable*

Generate Secret Key

Secret Key given

Request to KGC

Randomly Generate Session Key

If Authorised User

Retreive Session Key

Decrypt the key and msg

##### Data Flow Diagram:

The Data Flow diagram is a graphic tool used for expressing system requirements in a graphical form. The DFD also known as the “bubble chart” has the purpose of clarifying system requirements and identifying major transformations that to become program in system design.

Thus DFD can be stated as the starting point of the design phase that functionally decomposes the requirements specifications down to the lowest level of detail. The DFD consists of series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system.

A DFD describes what data flow is rather than how they are processed, so it does not depend on hardware, software, data structure or file organization. The DFD consists of series of bubbles joined by lines. The bubbles represent data transformations and the lines represent data flows in the system. A DFD describes what data flow is rather than how they are processed, so it does not depend on hardware, software, data structure or file organization.

Session Checking

User Verifying

DHKE

Session Management

Share Key

DIKE



Request to KGC

Give Secret Key to user

User request to KGC with Members

KGC

response to member

Randomly Select a Session Key

KGC Generate Hash Function and Encrypt Sesson key

Broadcast key to member

Retrive Session Key

Decrypt the key

User

Session Creation

Session Authentication

Deriving

Session Users

Random Oracle

Internet

Valid user

CHAPTER - 4

## SOFTWARE DESCRIPTION

##### 4.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 thatcan be characterized by all of the following buzzwords:

Simple

Architecture neutral

Object oriented

Portable

Distributed

High performance

Interpreted

Multithreaded

Robust

Dynamic

Secure

With most programming languages, you either compile or interpret a program so that you can run it on your computer. The Java programming language is unusual in that a program is both compiled and interpreted. With the compiler, first you translate a program into an intermediate language called Java byte codes —the platform-independent codes interpreted by the interpreter on the Java platform. The interpreter parses and runs each Java byte code instruction on the computer.

Compilation happens just once; interpretation occurs each time the program is executed. The following figure illustrates how this works.

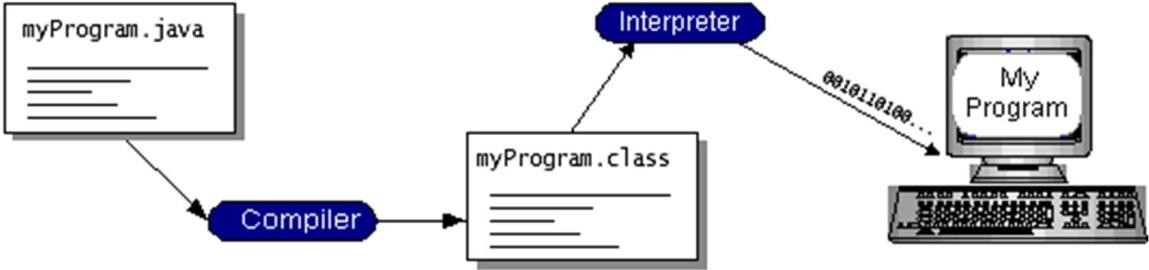


FIGURE 3.1- WORKING OF JAVA

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 is

grouped into libraries of related classes and interfaces; these libraries are known as packages

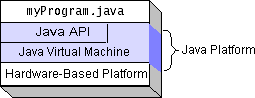
The following figure depicts a program that’s running on the Java platform. As the figure shows, the Java API and the virtual machine insulate the program from the hardware.

FIGURE 3.2- 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.

##### What Can Java Technology Do?

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

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

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

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

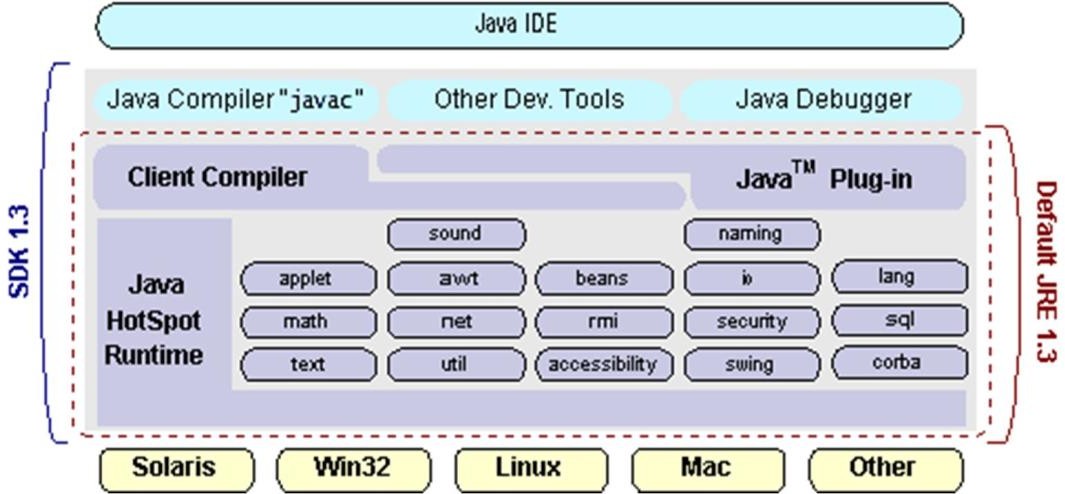


FIGURE 3.3 – JAVA 2 SDK

##### 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, the 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.

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.

##### 4.4 Networking

**TCP/IP stack**

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

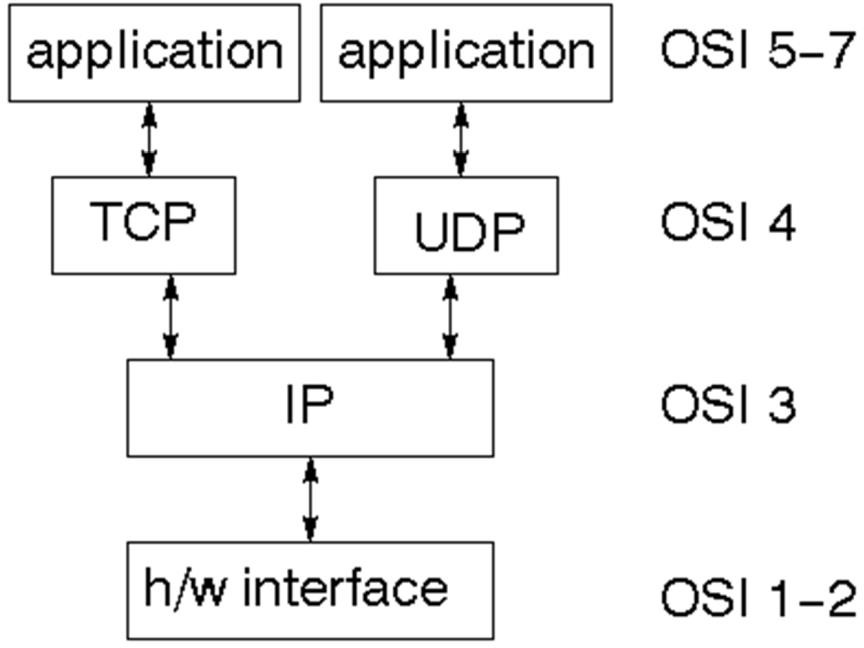


FIGURE 3.4 – TCP/IP STACK

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

##### IP datagram’s

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

##### 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

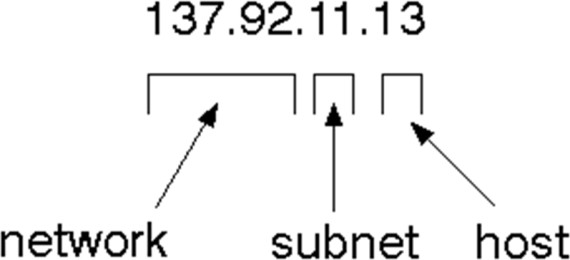


FIGURE 3.5- IP ADDRESSING

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

##### Port addresses

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

##### Sockets

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

#include <sys/types.h> #include <sys/socket.h>

int socket(int family, int type, int protocol);

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

**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 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:

###### 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.

###### 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.

###### 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.

###### 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.

###### 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.

###### 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.

###### 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.

CHAPTER - 5

## DETAILED DESIGN

5. **DETAILED DESIGN**

##### 5.3.1 Pseudo Code Description

##### LOGIN

Input: Username and password GET Username

GET Password

IF (Username == New Username && Password ==New Password) THEN

Add Username ELSE

Login Successfully ENDIF

##### FILE TRANSMISSION

Input: File , IP address GET Sender IP address Browse the file name.

IF (Select Option==Public) THEN

Send file to all users who currently available in network ENDIF

IF (Select Option==Private) THEN Send file to particular user.

ENDIF.

##### SESSION(KEY) GENERATION

Input: Access Specifier module selection. GET the key .

Key may be Public (or) Private.

Using the SHA-1+HMAC method, key is generated.

**function**hmac (key, message) **if** (length(key) >blocksize) **then** key = hash(key)

##### end if

**if** (length(key) <blocksize) **then**

key = key ∥ [0x00 \* (blocksize - length(key))]

##### end if

o\_key\_pad = [0x5c \* blocksize] ⊕ key i\_key\_pad = [0x36 \* blocksize] ⊕ key

**return** hash(o\_key\_pad∥ hash(i\_key\_pad∥ message))

##### end function

**SESSION(KEY) MANAGEMENT**

Input: Key GET the key

INITIALIZE the connection IF (new\_key==existing\_key)

UPDATE the key using the pre-distribution key THEN

AUTHENTICATE using centralized distribution key ELSE

UPDATE new\_key in the routing table

##### HACKING ZONE MONITORING

Input: Data

GET the node details.

IF (node in network==node in router) THEN User is authorized can send data through keys ELSE

Detect the false name hacked by hackers stored in database.

##### AUTHENTICATION

Input: Message

GET Sender IP Address

GET Receiver IP Address

IF (Select Receiver Option==Public) THEN Text will be displayed to all the receivers.

ENDIF

IF (Select Receiver Option==Private) THEN

Text will be displayed only to the particular receiver ENDIF

**ARCHITECTURE:**



##### Modules:

1. Create Session and Authentication
2. Secure Privacy
3. Key Exchange
4. Restricted Random Oracle
   1. Create Session and Authentication

Each and every user can create our session account based in session management.

Session management collects user information for processing the session

accessing .User after create a session to authenticate into session management and accessing Diffie Helman Key Exchange protocol to participate in the session.



* 1. Secure Privacy

Every user have own privacy information it’s useful for session maintenance. Session validating the user information for access the key and data. It having some security validation for verifies the user.



* 1. Key Exchange

The DHKE can create a session key for transmit and accessing the data. The session key to maintain in the session management. The session key for share over internet for using DIKE protocol. After verifying the session user can retrieve the data and key accessing.



* 1. Restricted Random Oracle

In cryptography, a random oracle is an oracle (a theoretical black box) that responds to every unique query with a (truly) random response chosen uniformly from its output domain. If a query is repeated it responds the same way every time that query is Submitted.



**DFD:**



## CHAPTER - 6 CODING AND TESTING

* + 1. **CODING AND TESTING**

##### TESTING

* + 1. **PROCESS**

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

##### TYPES OF TESTS UNIT TESTING

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

##### INTEGRATION TESTING

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

##### FUNCTIONAL TESTING

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

##### SYSTEM TESTING

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

##### WHITE BOX TESTING

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

##### BLACK BOX TESTING

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

##### UNIT TESTING

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

##### TEST STRATEGY AND APPROACH

Field testing will be performed manually and functional tests will be written in detail.

##### Test objectives

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.
* Features to be tested
* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

##### Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications,

e.g. components in a software system or – one step up – software applications at the company level – interact without error.

##### Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

##### Test Results

All the test cases mentioned above passed successfully. No defects encountered.

##### SYSTEM 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.

##### FEASIBILITY STUDY:

Depending on the results of the initial investigation the survey is now expended to a more detailed feasibility study. “FEASIBILTY STUDY” is a test of system proposal according to its workability, impact of the organization, ability to meet needs and effective use of the resources.

Steps in feasibility analysis:

Eight steps involved in the feasibility analysis are:

* Form a project team and appoint a project leader.
* Enumerate potential proposed system.
* Define and identify characteristics of proposed system.
* Determine and evaluate performance and cost effective of each proposed system.
* Weight system performance and cost data.
* Select the best proposed system.
* Prepare and report final project directive to management.

Three key considerations involved in the feasibility analysis are

* + ECONOMICAL FEASIBILITY
  + TECHNICAL FEASIBILITY
  + BEHAVIORAL FEASIBILITY

##### 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.

##### 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.

##### BEHAVIORAL FEASIBILITY

The staff of the organization was open-minded towards the acceptance of this new concept. No specialized training would-be needed, only a few hours. Of instructed demo to needs to be given to the user .hence the software exhibits behavioral feasibility

##### 7.JAVA CODES

**CODE:**

**KGC.java** import java.io.\*; import java.net.\*; import java.util.\*; import java.sql.\*; class KGC

{

Socket soc; ServerSocket ss; DataInputStream dis; DataOutputStream dos; ObjectInputStream ois;

ObjectOutputStream oos;

Connection con;

Statement st;

ResultSet rs;

String content,keyinfo;

Vector nodena=new Vector();

//Vector nodeList=new Vector();

Vector<String> ipAddress=new Vector<String>(); Vector portNum=new Vector();

Vector v=new Vector(); KGC()

{

try

{

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

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/key\_exchange", "root","root123");

st=con.createStatement();

// int del=st.executeUpdate("delete from tempnodeinfo"); ss=new ServerSocket(7373);

System.out.println("MainServer Starting ");

while(true)

{

soc=ss.accept();

oos=new ObjectOutputStream(soc.getOutputStream()); ois=new ObjectInputStream(soc.getInputStream());

String msg = (String)ois.readObject(); System.out.println("NodeForm msg is :" +msg);

if(msg.equals("logincheck"))

{

System.out.println("===Serverf Respond "+msg); String user=(String)ois.readObject();

String pass=(String)ois.readObject(); String nodeDetails = "";

rs=st.executeQuery("select NodeName,IpAddress,PortNumber from nodeinfo where NodeName='"+user+"' and PortNumber='"+pass+"'");

if(rs.next())

{

nodeDetails += rs.getString(1)+"#"; nodeDetails += rs.getString(2)+"#";

nodeDetails += rs.getString(3)+"#"; oos.writeObject("success"); oos.writeObject( nodeDetails );

}

else

{

dos.writeUTF("failed");

}

}

else if( msg.equals("getOthers")){ Vector otherNodes = new Vector();

String nodeName = (String)ois.readObject(); System.out.println("===>"+nodeName);

rs = st.executeQuery("select NodeName from node info where NodeName != '"+nodeName+"'");

while( rs.next()){

otherNodes.addElement(rs.getString(1)); System.out.println("===>"+otherNodes);

}

oos.writeObject( otherNodes );

}

else if( msg.equals("forsessionkey")){

String nodeName = (String)ois.readObject();

Object[] selectedNodes = (Object[])ois.readObject(); System.out.println("node list value-

->"+nodeName+"\t"+selectedNodes.toString());

Vector nodeList = new Vector();

nodeList.add(nodeName);

for(int i=0;i<selectedNodes.length;i++){ nodeList.add(selectedNodes[i]);

}

System.out.println("nodeList-->"+nodeList);

String[] keys = new String[ selectedNodes.length ]; String inputForHash = "";

for( int i = 0; i < selectedNodes.length; i++ ){

rs = st.executeQuery("select keyinfo from keyinfo2 where nodename = '"+selectedNodes[i].toString()+"'");

if( rs.next()){

keys[i] = rs.getString(1);

}

}

System.out.println("The key is "+keys.length); Random random = new Random();

int rNum = random.nextInt(keys.length ); System.out.println("The random is "+rNum);

String sessionKey = keys[rNum]; System.out.println("The Session Key "+sessionKey); inputForHash = nodeName;

for( Object s : selectedNodes ){

inputForHash += (String)s;

}

System.out.println( inputForHash ); st.executeUpdate("delete from hashinfo");

int insert=st.executeUpdate("insert into hashinfo values('"+inputForHash+"','"+sessionKey+"')");

long hashValue = HashAlgorithm.cbuHash( inputForHash ); System.out.println( hashValue );

String encryptedSessionKey =

FileProtection.encrypt( sessionKey,String.valueOf( hashValue ));

System.out.println("The value is"+encryptedSessionKey ); ArrayList nodeInfo = new ArrayList();

for(int j=0;j<nodeList.size();j++)

{

ResultSet rs=st.executeQuery("select \* from nodeinfo where NodeName='"+nodeList.get(j).toString()+"'");

while(rs.next()){

nodeInfo.add( rs.getString(2)); nodeInfo.add( rs.getString(3));

}

}

System.out.println(nodeInfo);

BroadCastKey.secureChannel( nodeInfo,encryptedSessionKey);

}

else if(msg.equals("retrievSession")){

String nodeName=(String)ois.readObject(); String nodeN=null;

String sessionKeyIs="";

rs=st.executeQuery("select \* from hashinfo"); if(rs.next()){

nodeN=rs.getString(1);

}

if((nodeN.contains(nodeName))){ oos.writeObject("Authorised");

}

else{

oos. write

Object(nodeN);

System.out.println("Not a Authorised User"); oos.writeObject("NotAuthorised");

}

else if( msg.equals("getnodeinfo")){

Object[] nodeList = (Object[])ois.readObject(); System.out.println(""+nodeList);

ArrayList nodeInfo = new ArrayList(); for( int i =0; i < nodeList.length; i++ ){

rs = st.executeQuery("select \* from NodeInfo where NodeName = '"+nodeList[i].toString()+"'");

while( rs.next() ){

nodeInfo.add( rs.getString( 2 ));

nodeInfo.add( rs.getString( 3 ));

}

}

}

else

{

oos.writeObject( nodeInfo );

String nodename=(String)ois.readObject(); String ip=(String)ois.readObject();

String port=(String)ois.readObject(); System.out.println("NAME" +nodename+ "ip" +ip);

int reg=st.executeUpdate("insert into nodeinfo values('"+nodename+"','"+ip+"','"+port+"')");

GenerateKey gk=new GenerateKey(); gk.groupKey();

rs=st.executeQuery("select \* from keyinfo"); if(rs.next())

DB "+keyinfo);

{

keyinfo=rs.getString(1); System.out.println("Key info from

int t=st.executeUpdate("insert into keyinfo2 values('"+nodename+"','"+keyinfo+"')");

}

oos.writeObject("registered"); oos.writeObject(keyinfo);

}

}

}

catch (Exception c)

{

c.printStackTrace();

}

}

public static void main(String[] args) throws Exception

{

new KGC();

}

}

**Broadcast.java** import java.util.\*; import java.io.\*; import java.net.\*;

class BroadCastKey

{

public static void secureChannel( ArrayList nodeInfo,String sessionKeyIs ){ try{

for( int i = 0; i < nodeInfo.size(); i = i+2 ){

Socket soc =newSocket( nodeInfo.get(i).toString(),Integer.parseInt((String)nodeInfo.get(i+1)));

ObjectOutputStream oos = newObjectOutputStream( soc.getOutputStream());

oos.writeObject("sessionKey"); oos.writeObject( sessionKeyIs );

}

}

catch(Exception e){

e.printStackTrace();

}

}

}

##### FileProtection.java

import java.security.\*;

import java.security.spec.InvalidKeySpecException; import javax.crypto.Cipher;

import javax.crypto.spec.SecretKeySpec; import sun.misc.\*;

import javax.crypto.\*;

class FileProtection

{

private static final String ALGORITHM = "AES"; private static final int ITERATIONS =2;

private static final byte[] keyValue =new byte[] {'T', 'h', 'i', 's', 'I', 's', 'A', 'S', 'e', 'c', 'r', 'e', 't', 'K', 'e', 'y'};

public static String encrypt(String value,String salt) throws Exception

{

Key key = generateKey();

Cipher c = Cipher.getInstance(ALGORITHM); c.init(Cipher.ENCRYPT\_MODE, key);

String valueToEnc = null; String eValue = value;

for (int i = 0; i < ITERATIONS; i++)

{

valueToEnc = salt + eValue;

byte[] encValue = c.doFinal(valueToEnc.getBytes()); eValue = new BASE64Encoder().encode(encValue);

}

return eValue;

}

public static String decrypt(String value,String salt) throws Exception

{

Key key = generateKey();

Cipher c = Cipher.getInstance(ALGORITHM); c.init(Cipher.DECRYPT\_MODE, key);

String dValue = null;

String valueToDecrypt = value;

for (int i = 0; i < ITERATIONS; i++)

{

byte[] decordedValue = new

BASE64Decoder().decodeBuffer(valueToDecrypt); byte[] decValue = c.doFinal(decordedValue);

dValue = new String(decValue).substring(salt.length()); valueToDecrypt = dValue;

}

return dValue;

}

public static Key generateKey() throws Exception

{

Key key = new SecretKeySpec(keyValue, ALGORITHM); String k=key.toString();

String[] k1=k.split("@");

//SecretKeyFactory keyFactory = SecretKeyFactory.getInstance(ALGORITHM);

// key = keyFactory.generateSecret(new DESKeySpec(keyValue));

//System.out.println("Key value is:"+key);

//System.out.println("Key is:"+k1[1]);

//XML.Encryptkey=k1[1]; return key;

}

public static String secretKey() throws Exception

{

KeyGenerator keyGen = KeyGenerator.getInstance("AES"); SecretKey key = keyGen.generateKey();

String sec = key.toString(); return sec;

}

}

##### HashAlgorithm.java

public class HashAlgorithm

{

public HashAlgorithm()

{

}

public static long cbuHash( String MSG )

{

long h = 0;

byte[] str = MSG.getBytes();

for ( int i= 0; i < str.length ; i++ )

{

h = ( h << 2 ) + str[i];

}

return h;

}

}

//ABC = msg

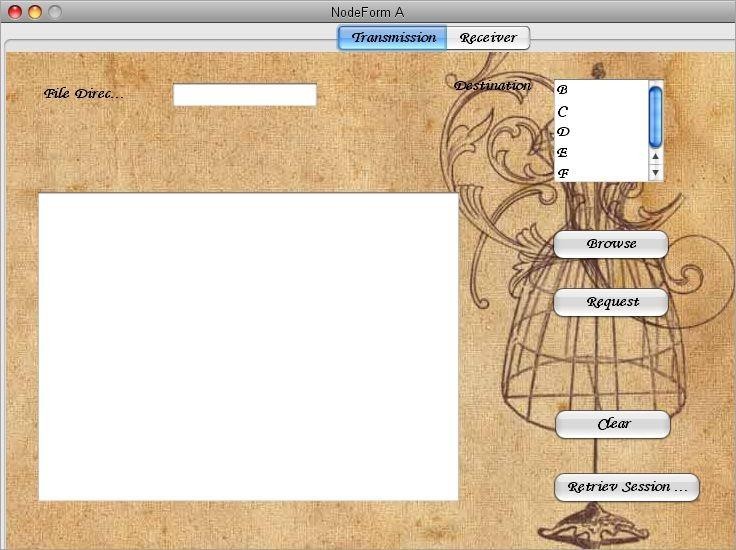
**CHAPTER – 8**

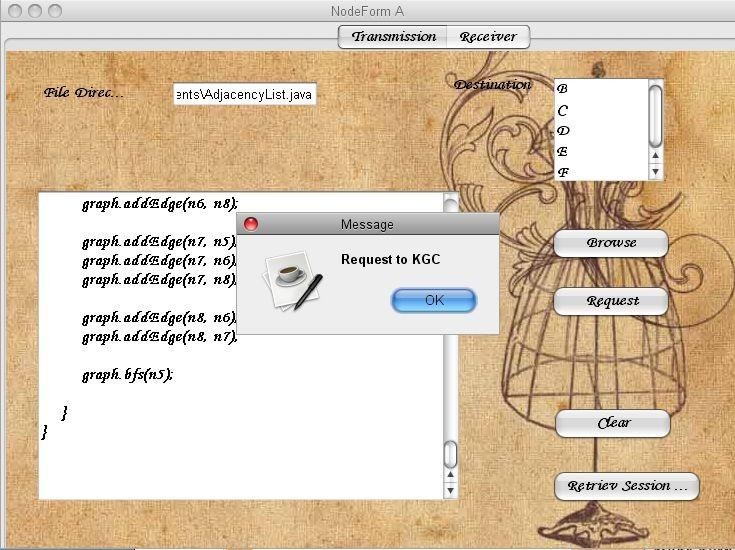
**OUTPUT SCREENSHOTS**

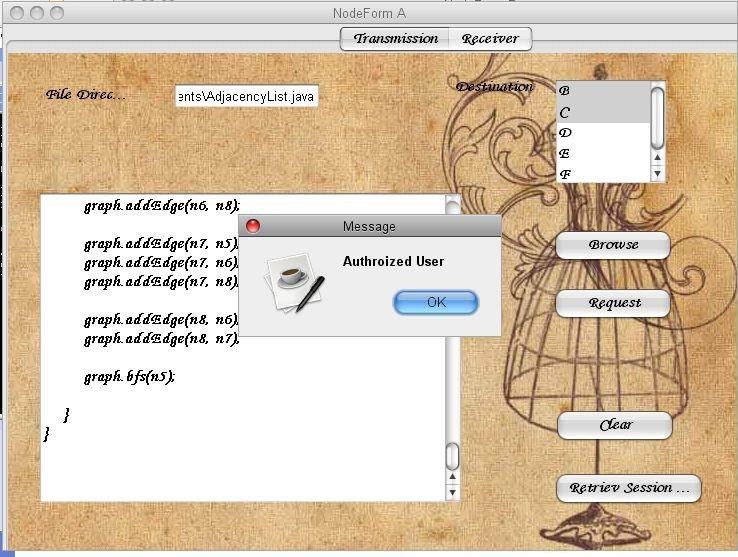
**8.OUTPUT SCREENSHOTS**



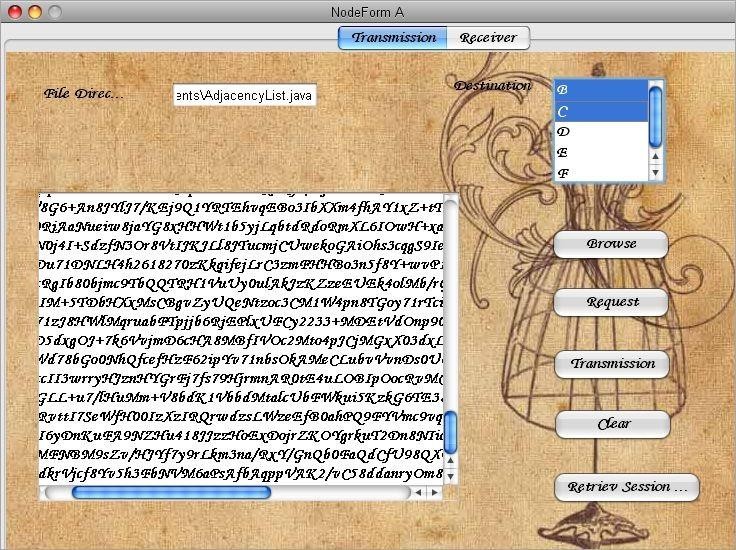




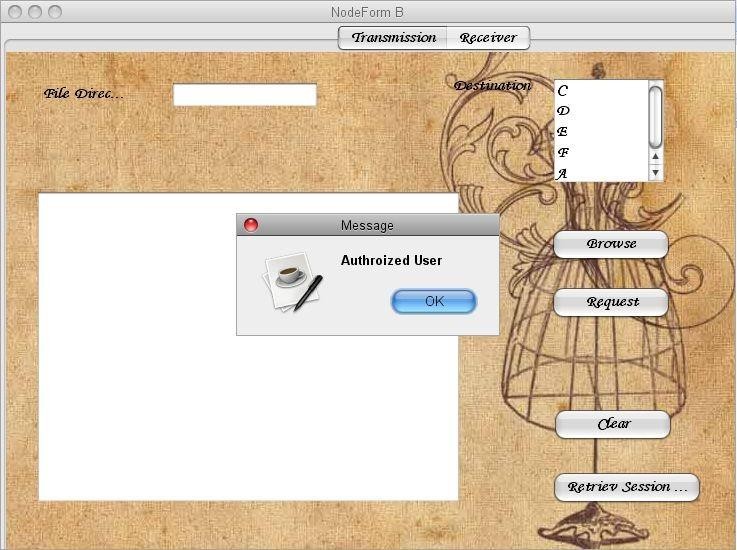












**CHAPTER - 9**

# BIBILIOGRAPHY

**REFERENCE**

**9.BIBILIOGRAPHY**

* + - 1. Williamson, Malcolm J. (January 21, 1974). Non-secret encryption using a finite field (PDF) (Technical report). Communications Electronics Security Group. Archived (PDF) from the original on 2017-03-23.
      2. B. Carbunar and M. Tripunitara, "Fair Payments for Outsourced Computations", Proc. Seventh IEEE Int'l Conf. Sensor Ad Hoc and Mesh Comm. and Networks (SECON), 2010.
      3. M. Belenkiy, M. Chase, C.C. Erway, J. Jannotti, A. Kp and A. Lysyanskaya, "Incentivizing Outsourced Computation", NetEcon '08: Proc. Third Int'l Workshop Economics of Networked Systems, 2008.
      4. B. Carbunar and R. Sion, "Uncheatable Reputation for Distributed Computation Markets", Proc. Int'l Conf. Financial Cryptography (FC) and Data Security, 2006.
      5. Williamson, Malcolm J. (August 10, 1976). Thoughts on Cheaper Non- Secret Encryption (PDF) (Technical report). Communications Electronics Security Group. Archived (PDF) from the original on 2004-07-19.
      6. P. Barreto, H. Kim, B. Lynn, M. Scott, “Efficient Algorithms for Pairing- based Cryptosystems”, in Advances in Cryptology – Crypto 2002, Lecture Notes in Computer Science, Springer-Verlag, 2002.
      7. M. Bellare, A. Desai, D. Pointcheval, P. Rogaway, “Relations among notions of security for public-key encryption schemes”, in Advances in Cryptology – Crypto ’98, Lecture Notes in Computer Science, Vol. 1462, Springer-Verlag, pp. 26–45, 1998.
      8. M. Bellare and P. Rogaway, "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols", Proc. First ACM Conf. Computer and Comm. (CCS '93) Security, pp. 62-73, 1993.