A PRACTICAL PASSWORD-BASED TWO-SERVER AUTHENTICATION AND KEY EXCHANGE SYSTEM

A PROJECT REPORT

Submitted by

LAVANYAA.M	71204205017
PRIYA.A	71203205029
SARANYA.V	71204205043

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

in

INFORMATION TECHNOLOGY

KUMARAGURU COLLEGE OF TECHNOLOGY, COIMBATORE

ANNA UNIVERSITY: CHENNAI 600 025

APRIL 2008

ANNA UNIVERSITY: CHENNAI 600 025

BONAFIDE CERTIFICATE

Certified that this project report "“**A PRACTICAL PASSWORD-BASED TWO-SERVER AUTHENTICATION AND KEY EXCHANGE SYSTEM**” is the bonafide work of **“LAVANYAA.M, PRIYA.A** and **SARANYA.V”** who carried out the project work under my supervision.

SIGNATURE

Dr.Thangasamy, B.E(Hons)., Ph.D.
Dean Of The Department
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

SIGNATURE

Mr.E.A.Vimal, M.E.
Lecturer.
Dept of Information Technology,
Kumaraguru College of Technology,
Coimbatore – 641 006.

The candidates with University Register No 71204205017, 71204205029 and 71204205043 were examined by us in the project viva-voice examination held on.23.04.08....

INTERNAL EXAMINER

23/4/08

EXTERNAL EXAMINER

DECLARATION

We,

LAVANYAA.M	71204205017
PRIYA.A	71203205029
SARANYA.V	71204205043

hereby declare that the project entitled **"A PRACTICAL PASSWORD-BASED TWO-SERVER AUTHENTICATION AND KEY EXCHANGE SYSTEM "**, submitted in partial fulfillment to Anna University as the project work of Bachelor of Technology (Information Technology) Degree, is a record of original work done by us under the supervision and guidance of Department of Information Technology, Kumaraguru college of Technology, Coimbatore.

Place: Coimbatore

Date: 23.04.08

M.Lavanya

[Lavanyaa.M]

A. Priya

[Priya.A]

V. Saranya.

[Saranya.V]

Project Guided by

[Mr.E.A.Vimal M.E.]

ACKNOWLEDGEMENT

We express our sincere thanks to our chairman **Padmabhushan Arutselvar Dr. N. Mahalingam B.Sc, F.I.E** and correspondent **Prof. Balasubramanian Ph.D.** for all their support and ray of strengthening hope extended. We are immensely grateful to our principal **Dr. Joseph V. Thanikal M.E., Ph.D., PDF., CEPIT.,** for his invaluable support to the outcome of this project.

We also thank **Dr.Thangasamy B.E(Hons)., Ph.D.,** Dean of the Department of Computer science and engineering for his valuable advice and useful suggestions throughout this project.

We also extend our heartfelt thanks to our project guide **Mr. E.A. Vimal M.E.,** lecturer, Department of Information Technology who rendered his valuable guidance and support to perform our project work extremely well.

We express our heartiest thanks to our project coordinator **Mr.K.R.Baskaran M.E.,** Assistant Professor, Department of Information Technology who has helped us to perform our project work extremely well.

We thank the teaching and non-teaching staffs of our Department for providing us the technical support in the duration of our project.

We express our humble gratitude and thanks to our beloved parents and family members who have supported and helped us to complete the project and our friends, for lending us valuable tips, support and co-operation throughout the project work.

ABSTRACT

Most password-based user authentication systems place total trust on the authentication server where clear text passwords or easily derived password verification data are stored in a central database. Compromise of the authentication server by either outsiders or insiders subjects all user passwords to exposure and may have serious consequences. There are many security concerns associated with password authentication, due mainly to the fact that most users' passwords are drawn from a relatively small and easily generated dictionary. If information sufficient to verify a password guess is leaked, the password may be found. Such an attack is called as offline dictionary attack.

In this project, we present a practical password-based user authentication and key exchange system employing novel two-server architecture. Our system has a number of appealing features. In our system, only a front-end service server engages directly with users while a control server stays behind the scene; therefore, it can be directly applied to strengthen single-server password systems. In addition, the system is secure against offline dictionary attacks mounted by either of the two servers.

CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVATIONS

ASF	:	Apache Software Foundation
CS	:	Control Server
HTML	:	Hypertext Transfer Protocol
J2EE	:	Java 2 Enterprise Edition
JSP	:	Java Server Pages
LAN	:	Local Area Network
MAC	:	Message authentication code
RPC	:	Remote Procedure Calls
SQL	:	Standard Query Language
SS	:	Service Server

INTRODUCTION

1. INTRODUCTION

1.1 GENERAL:

A password-based user authentication system involves two servers. Here user's password is split into two parts during user registration and stores each part in each server. The front-end service server engages directly with users while a control server stays behind the scene and it helps service server in authentication. Each server uses its password share to authenticate the user. This authentication system is low cost and easy to use. A user only needs to memorize a short password and can be authenticated anywhere, anytime regardless of the types of access devices he/she employs.

It is important to note the essential differences between the two-server model and the earlier multiserver models: In the two-server model, a user ends up establishing a session key only with the public server, and the role of the back-end server is merely to assist the public server in user authentication, while in the multiserver models, a user establishes a session key with each of the servers. For exactly this reason, we view the two-server system in as a special case of the gateway augmented multiserver model of two servers. From a security point of view, servers in the multiserver models are equally exposed to outside attackers, while in the two-server model, only the public server faces such a problem. This clearly improves the server side security and in turns the overall system security in the two-server model.

1.2 PROBLEM DEFINITION:

The main objective of our project is to design a system that prevents offline dictionary attacks mounted by either of the two servers. However, the use of passwords has intrinsic weaknesses. It is a well-known problem that human-user-chosen passwords are inherently weak since most users choose short passwords that are easy to remember. In particular, passwords are normally drawn from a relatively small dictionary, so it allows for brute-force dictionary attacks, where an attacker enumerates every possible password in the dictionary to determine the actual password.

Dictionary attacks can be mounted online or offline. In an online dictionary attack, attackers attempt to log in to a server by trying all possible passwords from the dictionary until they find a correct one. In an offline dictionary attack, attackers record a past successful login session between a user and a server and then check all the passwords in the dictionary against the login transcript. Online dictionary attacks can be easily thwarted at the system level by limiting the number of unsuccessful login attempts made by a user. In contrast, offline dictionary attacks are notoriously harder to deal with. As a result, tremendous effort has been dedicated to countering offline dictionary attacks in password systems. The present single server authentication architecture was analyzed and used as a foundation for the proposed system

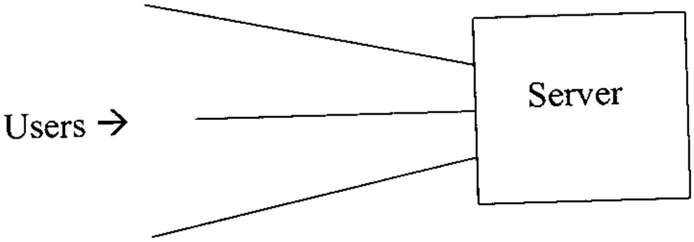
1.3 EXISTING SYSTEM:

Single-Server System:

Most of the existing password systems were designed over a single server, where each user shares a password with a single authenticate server. In this single server model user passwords are kept in databases. This type of single server system is given in figure 1.

Limitations:

The single server results in a single point of vulnerability in terms of offline dictionary attacks against the user password database.



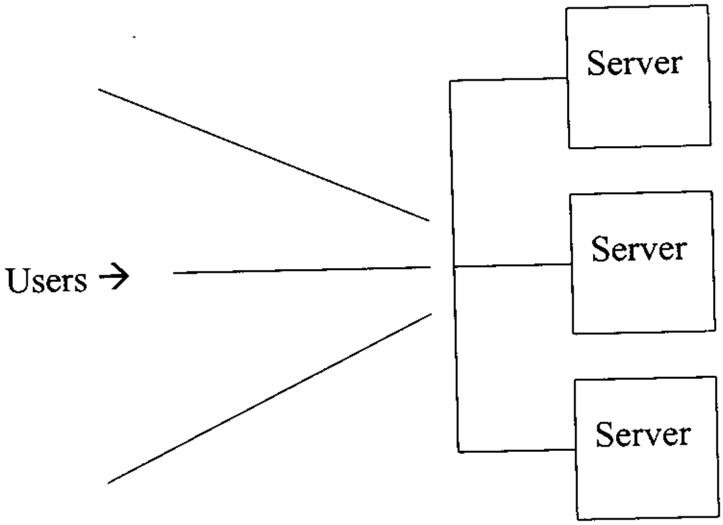
(Fig.1) SINGLE SERVER MODEL

Plain Multiserver System:

The second type is the plain multiserver model, in which the server side comprises multiple servers for the purpose of removing the single point of vulnerability; the servers are equally exposed to users and a user has to communicate in parallel with several or all servers for authentication. This type of plain multiserver system is given in figure 2.

Limitations:

The main problem with the plain multiserver model is the demand on communication bandwidth. Also simultaneous communication with multiple servers results in need for synchronization at the user side.



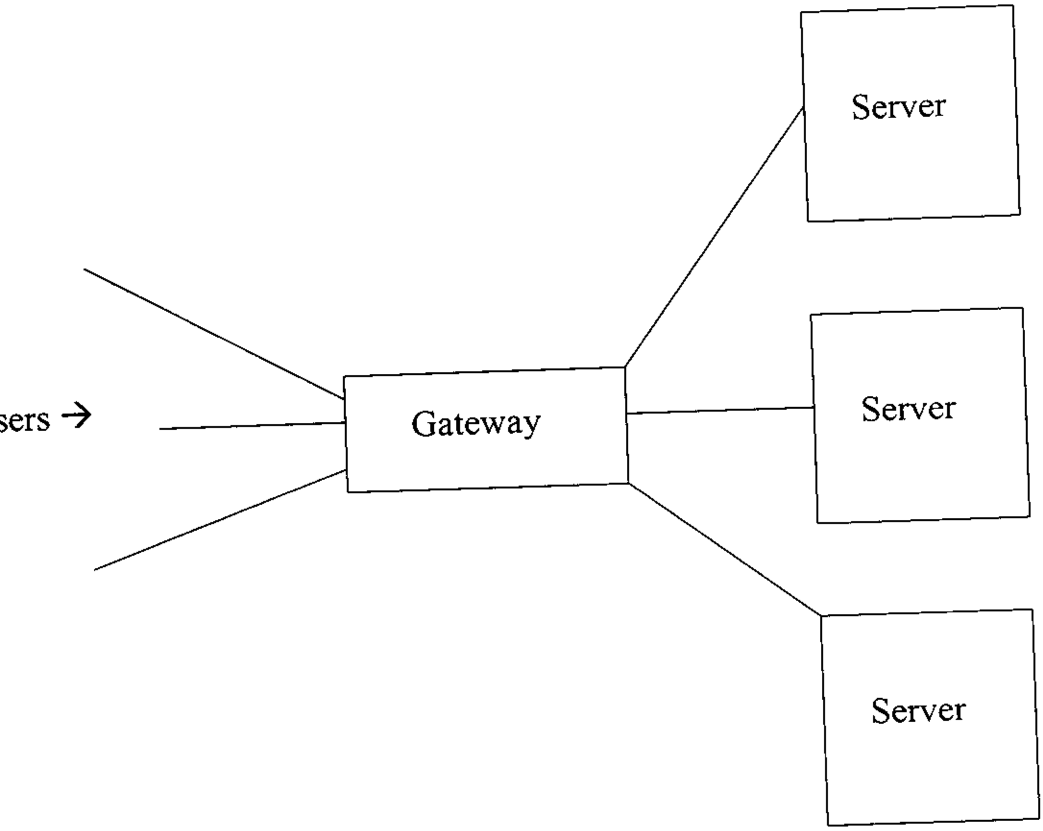
(Fig.2) PLAIN MULTISERVER MODEL

Gateway Augmented Multiserver:

The third type is the gateway augmented multiserver model, where a gateway is positioned as a relaying point between users and servers and a user only needs to contact the gateway. This type removes the demand of simultaneous communication by a user with multiple servers. This type of gateway augmented multiserver is given in figure 3.

Limitations:

Since gateway does not involve in service provision, authentication and other security enforcements, it generally imply more points of vulnerabilities



(Fig.3) GATEWAY AUGMENTED MULTISERVER MODEL

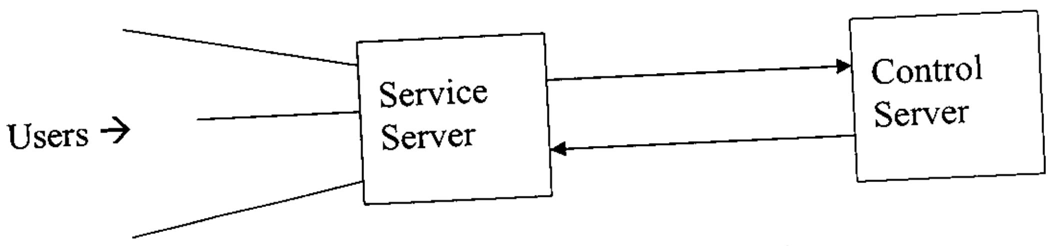
1.4 PROPOSED SYSTEM:

Two-Server System:

The proposed system is based on two-servers called as service server and control server. The service server is public while the control server is the back-end server and it is invisible to the users. The users password is transformed into two shares, which are held by service server and control server, respectively. Based on their respective shares, control server and service server together validate users during user login. This type of two-server system is given in figure 4.

Advantages of the proposed system:

- Secure against offline dictionary attack.
- Eliminating of a single point of vulnerability.
- No need of high bandwidth for communication.
- It is of low cost and easy to use.



(Fig.4) **TWO SERVER MODEL**

LITERATURE REVIEW

2. LITERATURE REVIEW

2.1 CLIENT/SERVER ARCHITECTURE:

The client/server software architecture is a versatile, message-based and modular infrastructure that is intended to improve usability, flexibility, interoperability, and scalability as compared to centralized, mainframe, time sharing computing. A client is defined as a requester of services and a server is defined as the provider of services. A single machine can be both a client and a server depending on the software configuration. The client/server architecture reduced network traffic by providing a query response system. In client/server architectures, Remote Procedure Calls (RPCs) or standard query language (SQL) statements are typically used to communicate between the client and server. One type of Client Server Architecture is Two Tier Architectures.

Two Tier Architectures:

With two tier client/server architectures the user system interface is usually located in the user's desktop environment and the various application services are usually in a server that is a more powerful machine that services many clients. Processing management is split between the user system interface environment and the application server environment. There are a number of software vendors that provide tools to simplify development of applications for the two tier client/server architecture. The two tier client/server architecture is a good a dozen to 100 people interacting on a LAN simultaneously.

2.2 J2EE:

Java Platform, Enterprise Edition or **JavaEE** is a widely used platform for server side programming in Java. The JavaEE Platform differs from the Standard Edition (SE) of Java in that it adds additional libraries which provide functionality to deploy fault-tolerant, distributed, multi-tier Java software, based largely on modular components running on an application server.

JavaEE includes several API specifications, such as JDBC, RMI, e-mail, JMS, web services, XML, etc, and defines how to coordinate them. Java EE also features some specifications unique to JavaEE for components. These include Enterprise JavaBeans, servlets, portlets (following the Java Portlet specification), JavaServer Pages and several web service technologies. This allows developers to create enterprise applications that are portable and scalable, and that integrate with legacy technologies. A JavaEE "application server" can handle the transactions, security, arity, scalability, concurrency and management of the components that are deployed to it, meaning that the developers can concentrate more on the business logic of the components rather than on infrastructure and integration tasks.

2.3. JAVA SERVER PAGES:

Java Server Pages (JSP) technology enables Web developers and designers to rapidly develop and easily maintain, information-rich, dynamic Web pages that leverage existing business systems.

JSP technology uses XML-like tags that encapsulate the logic that generates the content for the page. The application logic can reside in server-based resources (such as JavaBeans component architecture) that the page accesses with these tags. Any and all formatting (HTML or XML) tags are passed directly back to the response page. By separating the page logic from its design and display and supporting a reusable component-based design, JSP technology makes it faster and easier than ever to build Web-based applications.

2.4 APACHE TOMCAT APPLICATION SERVER:

Apache Tomcat is a web container developed at the Apache Software Foundation (ASF). Tomcat is a web server that supports servlets and JSPs. It providing an environment for Java code to run in cooperation with a web server. Tomcat can also function as an independent web server It adds tools for configuration and management but can also be configured by editing configuration files that are normally XML-formatted. Tomcat includes its own HTTP server internally. Tomcat is cross-platform, running on any operating system that has a Java Runtime Environment.

2.5 DATABASE DESIGN:

The database has become an integral part of almost every human's life. Without it, many things become very tedious, perhaps impossible tasks, Banks, university, and libraries are three examples of organizations that depend heavily on some sort of database system, On the Internet, Search engines, inline shopping, and even the website naming convention would be impossible without the use of a database.

Microsoft Access is a powerful program to create and manage the databases. It has many built in features to assist in constructing and viewing the information. Access is much more involved and is a more genuine database application than other programs.

Some keywords involved in this process are:

- Database File.
- Table.
- Field.
- Data-types.

Database File is the main file that encompasses the entire database and that is saved to hard-drive or floppy disk. A table is a collection of data about a specific topic. There can be multiple tables in a database. Fields are the different categories within a Table. Tables usually contain multiple fields. A field only has 1 data type.

2.6 HARDWARE SPECIFICATIONS (MINIMUM REQUIREMENTS):

Processor	:	Intel Pentium IV 2.2 GHZ
Hard Disk Drive	:	20 GB.
RAM	:	256 MB RAM minimum.

2.7 SOFTWARE SPECIFICATIONS (MINIMUM REQUIREMENTS):

Operating System:		Windows 2000/XP
Software	:	Apache Tomcat Application Server 4.1, MS-Access

DETAILS OF METHODOLOGY EMPLOYED

3. DETAILS OF METHODOLOGY EMPLOYED

3.1 TWO-SERVER PASSWORD AUTHENTICATION PROTOCOL:

In this section, we elaborate on our proposed password authentication and key exchange protocols upon the two server model. Here, we completely avoid the use of a public key cryptosystem at the server side. Our protocols are quite efficient in terms of both communication and computation. For ease of reference, notations that are used below are listed in Table 1.

Q, p, q	three large primes such that $Q = 2p + 1$ and $p = 2q + 1$.
g_1, g_2	$g_1, g_2 \in QR_p$ are of order q and the discrete logarithms to each other are not known, where QR_p is the group of quadratic residues modulo p.
g_3	$g_3 \in QR_Q$ is of order p.
π	a user's password.
$h(.)$	a cryptographic hash function modelled as the random oracle [8].
$\mathcal{U}$, $\mathcal{SS}$, $\mathcal{CS}$	identities of user, service sever and control server, respectively.

Table 1: Notations

3.1.1 SYSTEM MODEL:

Three types of entities are involved in our system, i.e., users, a service server (SS) that is the public server in the two server model, and a control server (CS) that is the back-end server. In this setting, users only communicate with SS and do not necessarily know CS. For the purpose of user authentication, a user U has a password which is transformed into two long secrets, which are held by SS and CS, respectively. Based on their respective shares, SS and CS together validate users during user login. We assume the following security model: SS is controlled by a passive adversary and SS is controlled by an active adversary in terms of offline dictionary attacks to user passwords.

By definition a passive adversary follows honest-but-curious behavior, that is, it honestly executes the protocol according to the protocol specification and does not modify data, but it eavesdrops on communication channels, collects protocol transcripts and tries to derive user passwords from the transcripts; moreover, when an passive adversary controls a server, it knows all internal states of knowledge known to the server, including its private key and the shares of user passwords. In contrast, an active adversary can act arbitrarily in order to uncover user passwords. This authentication protocol is given in the figure no 2.

We stress that this security model exploits the different levels of trust upon the two servers. As discussed earlier, this clearly holds with respect to outside attackers. As far as inside attackers are concerned, justifications come from our application and generalization of the system to the architecture of a single control server supporting multiple service servers, where the control server affords and deserves enforcing more stringent security measurements against inside attackers.

3.1.2 HIGH LEVEL DESCRIPTION:

Central to our protocol design is the defense against offline dictionary attacks by the servers when they are controlled by adversaries. The intuition is to "harden" a user's short password into two long shares 1 and 2 in such a way that they are no longer subject to offline dictionary attacks, and then distribute them to the two servers. As a result, an attacker cannot succeed in offline dictionary attacks without grabbing both shares by compromising both servers. During user login, the control server CS using its share 2 assists the service server SS using 1 in user authentication. More specifically, in an out-of-band user registration phase, user U splits his password into two long random secrets 1 and 2 and registers them to CS and SS , respectively, where During authentication, U using and SS using 1 authenticate each other and negotiate a secret session key, with the help of SS using 2 .Where the back-end server is strictly passive and is not allowed to eavesdrop on communication channels, while SS in our setting is allowed for eavesdropping. We believe this weakening is important and more realistic since eavesdropping is practically easy and insidious.

3.2 USER REGISTRATION:

The users should register with the control server and service server in order to login authentication. The user should split his password (X) into two random numbers X1 and X2 where X1, X2 belongs to Z_q and such that

$$X1 + X2 = X \pmod{q}$$

Where,

q : a prime number.

X : a user's password.

X1,X2 : password share's

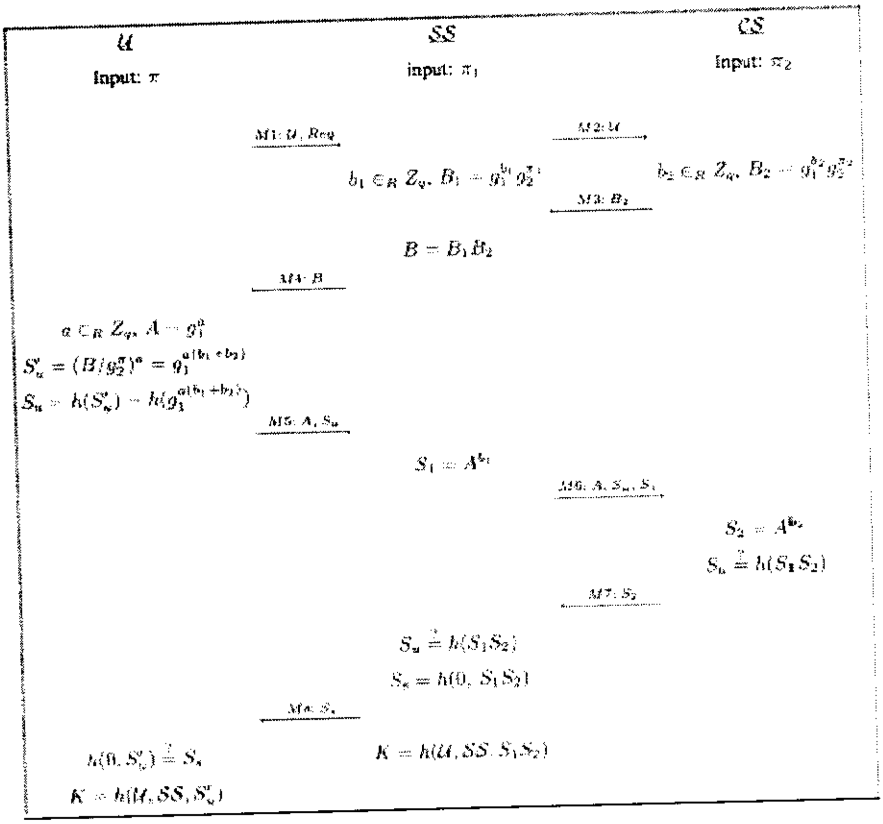
User then registers the password shares in a secure manner to SS and CS. SS stores its account information (user Id, X1) to its secret database, and CS stores (user Id, X2) to its secret database. In case CS supports multiple servers, it stores (user Id, X2) to distinguish users associated with different servers. One may wonder how U registers X2 to CS as CS is supposed hidden from U . This actually is not a problem in practice: U can reach CS through out-of-band channels, such as postal mail. Indeed, imagine that a user enrolls in a bank; it is not strange at all that the user still needs to submit a secret to a higher authority of the bank so as to activate his account.

3.3 A BASIC PASSWORD AUTHENTICATION PROTOCOL:

Let p , q , g 1 , g 2 , and h(.) be defined in Table 1. We outline the basic password authentication protocol in Fig. 5, which enables mutual authentication and key exchange between U and SS . In the figure, we have omitted the modulo p notation for arithmetic operations, as this should be clear from the context. To initiate a request for service, U sends his identity together with a service request Req to SS in M 1 . SS first relays the request to CS by sending the user ID in M 2 , and then selects a random number b 1 2 R Zq and computes B 1 =g1g2 mod p using his password share 1 . Upon receiving M 2 , CS chooses a random number b 2 from Zq and computes B 2 =g1g2 mod p using his password share 2 . CS then sends B 2 in M 3 to SS . Upon reception of B 2 , CS computes and sends B =B1B2 mod p in M4 . After receiving M4 , U selects a from Zq , and computes A =a 1mod p . U then sends A and Su to CS in M5 . Getting the message, CS computes S1 =A b1 (mod p) and sends S 1 , A and Su to SS in M 6 . Upon receipt of M6 , CS computes S2= A b2 (mod p) and checks whether Su =h (S1 S2), If it holds, SS is assured of the authenticity of U , and continues the protocol by sending S2 to CS in M 7 ; otherwise, SS aborts the protocol.

Fig: 5

Basic Password Authentication And Key Exchange Protocol


$\mathcal{U}$
Input: π
$\mathcal{SS}$
input: π_1
$\mathcal{CS}$
Input: π_2
$M1: \mathcal{U}, Req$
$M2: \mathcal{U}$
$b_1 \in_R Z_q, B_1 = g_1^{b_1} g_2^{\pi_1}$
$b_2 \in_R Z_q, B_2 = g_1^{b_2} g_2^{\pi_2}$
$M3: B_2$
$B = B_1 B_2$
$M4: B$
$a \in_R Z_q, A = g_1^a$
$S'_u = (B/g_2^{\pi})^a = g_1^{a(b_1+b_2)}$
$S_u = h(S'_u) = h(g_1^{a(b_1+b_2)})$
$M5: A, S_u$
$S_1 = A^{b_1}$
$M6: A, S_u, S_1$
$S_2 = A^{b_2}$
$S_u \overset{?}{=} h(S_1 S_2)$
$M7: S_2$
$S_u \overset{?}{=} h(S_1 S_2)$
$S_s = h(0, S_1 S_2)$
$M8: S_s$
$h(0, S'_u) \overset{?}{=} S_s$
$K = h(\mathcal{U}, \mathcal{SS}, S_1 S_2)$
$K = h(\mathcal{U}, \mathcal{SS}, S'_u)$

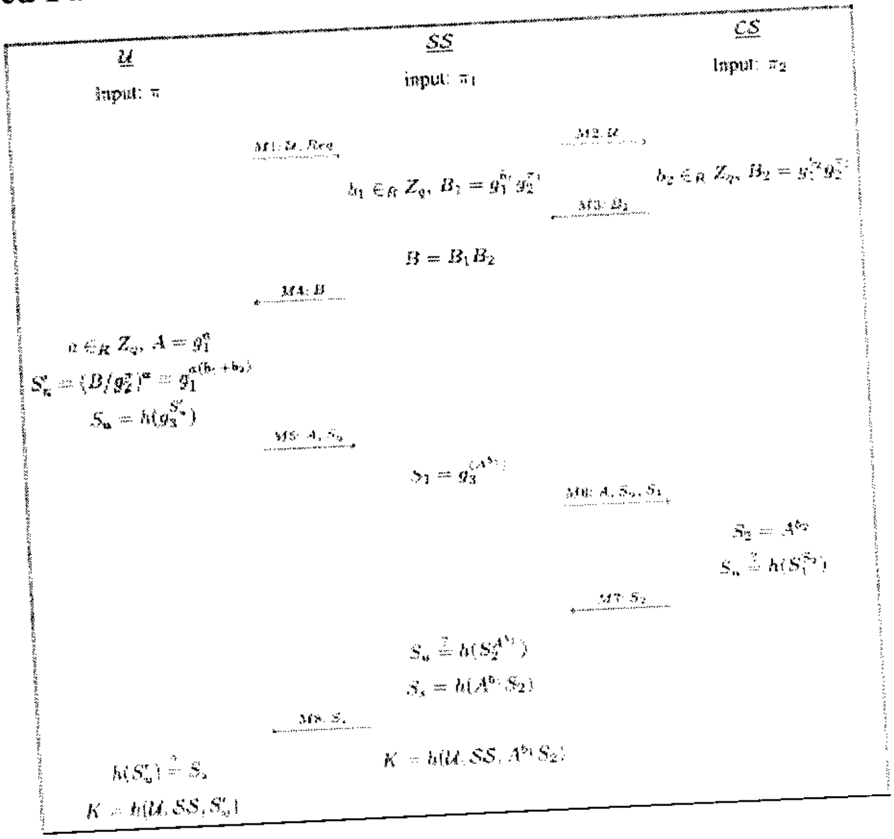
u Þ , respectively. U then sends A and Su to SS in M 5 . Getting the message, SS computes S1= A b1 (mod p) and sends S 1 , A and Su to SS in M 6 . Upon receipt of M 6 , SS computes S 2 =A b2 (mod p) and checks whether Su = h (S1 S2), If it holds, SS is assured of the authenticity of U , and continues the protocol by sending S 2 to SS in M 7 ; otherwise, SS aborts the protocol. Assuming SS receives S 2 in M 7 , it checks whether Su = h (S1 S2) . If it holds, SS is convinced of the authenticity of U . At this stage, both servers have authenticated U . SS then computes and sends Ss=h(S1 S2)to U in M 8 and afterward computes a session key K. otherwise, SS aborts the protocol. Upon receiving M 8.

3.4 AN IMPROVED PASSWORD AUTHENTICATION PROTOCOL:

There are two weaknesses in the above basic protocol. The first one is made obvious by recalling that we assumed a secret channel between SS and CS in the system model. The second one is that CS can compute the session key established between U and SS, so CS gets to know the data exchanged between them. While CS is passive, this clearly affects the principles of "need to know" and "separation of duty." To address these weaknesses, one-way secrecy of the channel between U and SS actually suffices in the above basic protocol.

Fig: 6

Improved Password Authentication And Key Exchange Protocol



$\mathcal{U}$
Input: π
SS
input: π_1
CS
Input: π_2
$M1: \mathcal{U}, Req$
$M2: \mathcal{U}$
$b_1 \in_R Z_q,\ B_1 = g_1^{b_1} g_2^{\pi_1}$
$b_2 \in_R Z_q,\ B_2 = g_1^{b_2} g_2^{\pi_2}$
$M3: B_2$
$B = B_1 B_2$
$M4: B$
$a \in_R Z_q,\ A = g_1^a$
$S_u' = (B/g_2^{\pi})^a = g_1^{a(b_1+b_2)}$
$S_u = h(g_3^{S_u'})$
$M5: A, S_u$
$S_1 = g_3^{(A^{b_1})}$
$M6: A, S_u, S_1$
$S_2 = A^{b_2}$
$S_u \overset{?}{=} h(S_1^{S_2})$
$M7: S_2$
$S_u \overset{?}{=} h(S_2^{A^{b_1}})$
$S_s = h(A^{b_1} S_2)$
$M8: S_s$
$K = h(\mathcal{U}, SS, A^{b_1} S_2)$
$h(S_u') \overset{?}{=} S_s$
$K = h(\mathcal{U}, SS, S_u')$

Our solution, indeed, takes advantage of this observation by having SS concealing A b1 (mod p) while still enabling CS for user authentication. The system setting and the security model are the same as in the basic protocol, except that no secret communication channel between SS and CS is assumed. Supposing U has already registered _1 to SS and _2 to CS as in the basic protocol, we present an improved password authentication protocol in Fig 6, where the system parameters are defined in Table 1, arithmetic operations associating with g1 and g2 are modulo p, and operations associating with g3 are modulo Q. This improved protocol is very similar to the basic protocol in Fig 5, with the only exception that we introduce the arithmetic operations associating with g3, which are represented by Su, S1 and the checks performed by SS and CS. By checking it against the basic protocol, we believe it is not hard to understand this improved protocol. So, we do not repeat the process of the protocol execution here. Next, we first check correctness of the protocol.

3.4.1 CORRECTNESS:

For the purpose of verifying U, CS needs to check $S_u \overset{?}{=} h(S_1^{a_2} \pmod Q)$, and SS needs to check $S_u \overset{?}{=} h(S_2^{a_1} \pmod Q)$ To make the checks work, it must hold that

$$g_3^{(g_1^{a_1 a_2} \bmod p)} \pmod Q = g_3^{(g_1^{a_1} \bmod p)(g_1^{a_2} \bmod p)} \pmod Q.$$

Notice that if the order of g3 is $\bar{p} \neq p$, then the exponentiation parts of g3 at both sides of the equation are not necessarily equal, that is, $g_1^{(a_1 a_2)} \pmod p \pmod{\bar{p}}$ is of high possibility to not equal $(g_1^{a_1} \bmod p)(g_1^{a_2} \bmod p) \pmod{\bar{p}}$, which, in turn, suggests the equation does not hold. Conversely, if the order of g3 is p as in the protocol, the equation is bound to hold as

$$g_3^{a(b_1+b_2)} (\bmod\ p)(\bmod\ p) = (g_3^{ab_1} \bmod\ p)(g_3^{ab_2} \bmod\ p)(\bmod\ p).$$

3.4.2 SECURITY:

We next examine security of this improved protocol. As we stated, this protocol is quite similar to the basic protocol, except for the introduction of computations associating with g3. It should be clear that this change makes it no easier to CS and SS for the purpose of offline dictionary attacks, as direct computation or guessing of x from $g_3^{x} (\bmod\ Q)$ is clearly no easier than from $g_1^{x} (\bmod\ p)$. As a matter of fact, nor does this change make it harder for CS and SS with respect to offline dictionary attacks, since it is of equal possibility to guess x from $g_3^{x} (\bmod\ Q)$ and from $g_1^{x} (\bmod\ p)$. We thus focus on the effect of removal of the secret channel between SS and CS, and whether CS can compute the session key between U and SS. Clearly, the removal of the secret channel would, in principle, facilitate outside adversaries who do not control any server to derive the session key between U and SS.

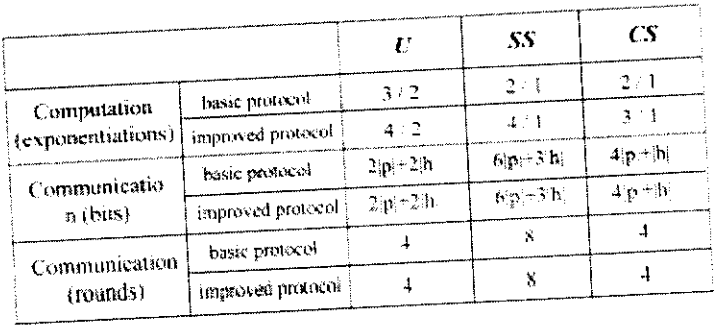
Compared to the basic protocol, an outside adversary additionally gleans $S_1 = g_3^{(A^{b_1})} (\bmod\ Q)$ and $S_2 = A^{b_2} (\bmod\ p)$. The adversary needs to know $A^{b_1} (\bmod\ p)$ in order to derive the session key. However, the additional $S_1 = g_3^{(A^{b_1})} (\bmod\ Q)$ does not help the adversary in computing $A^{b_1} (\bmod\ p)$, which is equivalent to computing the discrete log of S1. This suggests the removal of the secret channel between SS and CS does not, in fact, facilitate the outside adversary. For exactly the same reason, CS cannot compute the session key either with the knowledge of S1. As a result, we have managed to remove the weaknesses contained in the basic protocol.

3.5 COMPARISON OF BASIC AND IMPROVED PROTOCOL:

In this section, we examine performance of our proposed two protocols. Let $|p|$ and $|h|$ denote the bit length of p and the hash function $h(\cdot)$ respectively. We outline the performance results in Table. 2 We have three aspects to evaluate:

Table 2:

Comparison Of Basic And Improved Protocol



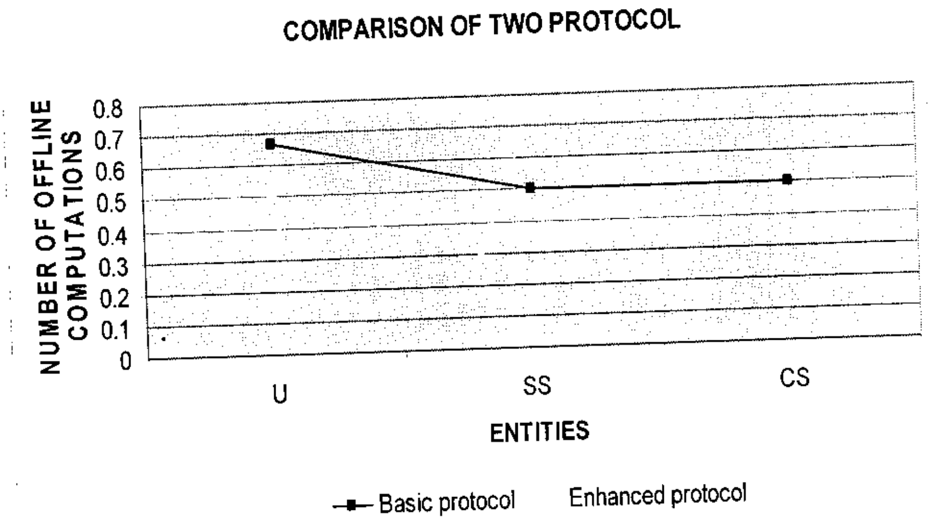
		U	*SS*	*CS*
Computation (exponentiations)	basic protocol	3 / 2	2 / 1	2 / 1
	improved protocol	4 / 2	4 / 1	3 / 1
Communication (bits)	basic protocol	2|p|+2|h	6|p|+3|h|	4|p|+|h|
	improved protocol	2|p|+2|h.	6|p|+3|h|	4|p|+|h|
Communication (rounds)	basic protocol	4	8	4
	improved protocol	4	8	4

3.5.1 COMPUTATION PERFORMANCE:

Since exponentiations dominate each party's computation overhead, we only count the number of exponentiations as the computation performance. The digits before "/" denote the total number of exponentiations performed by each party, and the digits following "/" denote the number of exponentiations that can be computed offline. Note that by leveraging on the techniques in [9], each of $g_1^{b_1}g_2^{a_1} \pmod{p}$ and $g_1^{b_2}g_2^{a_2} \pmod{p}$ can be computed by a single exponentiation.

Fig: 7

Comparison Of Two Protocol



3.5.2. COMMUNICATION PERFORMANCE IN TERMS OF BITS:

As $|Q|$ is only 1 bit longer than $|p|$. we do not distinguish between $|p|$. and $|Q|$ for ease of comparison. In addition, we have neglected including the bandwidth of M1 and M2 in this aspect of calculation.

3.5.3 COMMUNICATION PERFORMANCE IN TERMS OF ROUNDS:

One round is a one-way transmission of messages. Table 2 shows that the proposed two protocols demonstrate similar performance and are, in general, quite efficient in terms of both computation and communication to all parties. Take U, for example; it needs to calculate 3 and 4 exponentiations in the two protocols, respectively, and 2 of them can be performed offline. This means U only computes 1 and 2 exponentiations in real time in the respective protocols; the communication overhead for U is particularly low in terms of both bits and rounds. As a result, our protocols can readily support wireless applications.

3.6 QUADRATIC RESIDUE MODULO:

An integer 'a' which is not a multiple of a prime p is called a quadratic residue modulo p if the quadratic equation x2 = a mod p has a solution. If it has no solution then 'a' is called a quadratic non-residue modulo p. In mathematics, a number *q* is called a quadratic residue modulo *n* if there exists an integer *x* such that:

$$x^2 \equiv q \pmod{n}.$$

Otherwise, *q* is called a quadratic non-residue. For example, $3^2 \equiv 2 \pmod{7}$, and thus 2 is a quadratic residue modulo 7. In effect, a quadratic residue modulo *n* is a number that has a square root in modular arithmetic when the modulus is *n*.Quadratic residues occur in a random pattern; this has been exploited in applications to cryptography.

3.7 HASH FUNCTION:

A hash function H is a transformation that takes an input m and returns a fixed-size string, which is called the hash value h (that is, h = H(m)). Hash functions with just this property have a variety of general computational uses, but when employed in cryptography, the hash functions are usually chosen to have some additional properties.

The basic requirements for a cryptographic hash function are as follows.

- The input can be of any length.

- The output has a fixed length.
- H(x) is relatively easy to compute for any given *x*.
- H(x) is one-way.
- H(x) is collision-free.

A hash function H is said to be one-way if it is hard to invert, where hard to invert" means that given a hash value h, it is computationally infeasible to find some input *x* such that H(x) = h. If, given a message *x*, it is computationally infeasible to find a message *y* not equal to *x* such that H(x) = H(y), then H is said to be a weakly collision-free hash function. A strongly collision-free hash function H is one for which it is computationally infeasible to find any two messages *x* and *y* such that H(x) = H(y).

DATABASE DESIGN

Table 3:

Service Server

Field	Type	Null	Key
User_Id	Text	No	PRI
Password_share 1	Int	No	
Password_share 2	Int	No	

Table 4:

Control Server

Field	Type	Null	Key
User_Id	Text	No	PRI
Password_share 1	Int	No	
Password_share 2	Int	No	

PERFORMANCE EVALUATION

4. PERFORMANCE EVALUATION

Testing is a critical element of software quality and assurance and represents the ultimate review of specification design and coding. It is a vital activity that has to be enforced in the development of any system. This could be done in parallel during all the phases of system development. The feedback received from these texts can be used for further enhancement of the system under consideration. The testing phase conducts test using the Software Requirement Specification as a reference and with the goal to see whether system satisfies the specified requirements.

Standard procedures have been followed in testing our system. Test cases are generated for each screen. These test cases will cover every possibility which could result in both positive and negative results. These test plans are maintained for any further testing done on the system.

The main types of tests carried out are:

- Unit Testing.
- Integration Testing.
- System Testing.
- Validation Testing

4.1 Unit Testing

Each and every module is tested separately to check if its intended functionality is met. Some unit testing performed are:

1. Validating the User
2. Loading Database and applications

4.2 Integration Testing

It is the testing performed to detect errors on interconnection between modules. The applications should connect to respective databases. The application should connect to respective databases. The application events should be backed up in the log file for future recovery.

4.3 System Testing

The system is tested against the system requirements to see if all the requirements are met and if the system performs as per the specified requirements. The system is tested as a whole to check for its functionality.

4.4 Validation Testing

This test is done to check for the validity of the entered input. The user inputs to the corresponding application input fields are verified before updating in the database.

CONCLUSION

5. CONCLUSION

This system a password-based Two-server authentication and key exchange system has great potential for practical applications; it can be directly applied to existing standard single-server password applications, e.g., FTP and Web applications. It can also be applied in such way that a single control server supports multiple service servers. This system possesses many advantages, such as elimination of a single point of vulnerability, avoidance of PKI, low cost, easy use and high efficiency.

FUTURE ENHANCEMENTS

6. FUTURE ENHANCEMENT

The security model underlying our proposed protocols assumes that the control server can only be controlled by a passive adversary. As we have claimed, this assumption, while strong, is quite logical considering the positioning of the two servers in the two-server model and the applications of the model to federated enterprises. It is, however, clear that weakening of this assumption should be of both practical and theoretical significance, which we shall take as our future work.

APPENDICES

APPENDIX

SOURCE CODE:

check1.jsp :

```
<%@ include file="hash.jsp" %>
<%@ include file="enc.jsp" %>
<%@ include file="rand.jsp" %>
<%@ include file="prime.jsp" %>
<%@ page import="java.sql.*"%>
<%@ page import="java.net.*"%>
<%!
String user;
String server;
String pass;
double x=rand();//non - reputed values
double a=37;//Large Prime Number
double n=1234;//registered value
String fx1,y,gx,gy,gxy,gfxy,hgx,hgy,hgxy,hgfxfy;
double fx2,fx;
double k,y1,gx1,gy1,gxy1,fy,fxy,fgxy,fgxgy,gfxy1,fxs,fys,fxys;
String yd,xe,ydd;
double xe1,y4;

double q=PrimeNumber();
double p=((2*q)+1);
double Q=((2*p)+1);
double pi;
double pi1;
double pi2;
double g1;
double g2;
double b1;
double B1,B2,B;
double S1,Su,A,S2;
%>
```

```
<%
user=request.getParameter("T1");
pass=request.getParameter("T2");
server=request.getParameter("server2");

if(user!=null){

System.out.println("welcome to first server "+user);

fx=(Math.pow(x,a))%n;

String fxs=new String();
fxs=fxs.valueOf(fx);
String xs=new String();
xs=xs.valueOf(x);
String hfx=compare(fxs);
String hx=compare(fxs);
Object obj=(Object)user;
Object obj1=(Object)pass;
session.setAttribute("uname",obj);
session.setAttribute("pass",obj1);

InetAddress host = null;
host = InetAddress.getLocalHost();

String ho=host.getHostAddress();
System.out.println("The  Value of Ho()InetAddress.getLocalHost()is:"+ho);

response.sendRedirect("http://90.0.0.192:8081/imptwoserver/check1.jsp?use
r="+user+"&server="+ho+"&fx="+fx+"&x="+x+"&hfx="+hfx+"&hx="+hx
);
}
%>

<%
user=session.getAttribute("uname").toString();

try
{
        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

        Connection
conn=DriverManager.getConnection("jdbc:odbc:twoserver","","");
        Statement st=conn.createStatement();
        String query="select pass1  from login where uname='"+user+"'";

    ResultSet res=st.executeQuery(query);
    while(res.next())
    {
         String pss=res.getString(1);
         pi1=Double.parseDouble(pss);
     }
  }
  catch(Exception e)
  {
          System.out.println(e);
  }

  %>

  <%

   gfxy=request.getParameter("gfxy");

   if(gfxy!=null)
   {
            gx=request.getParameter("gx");
            gy=request.getParameter("gy");
            gx1=Double.parseDouble(gx);
            gfxy1=Double.parseDouble(gfxy);
            gy1=Double.parseDouble(gy);
            fgxgy=(Math.pow((gx1*gy1),a))%n;

            if(gfxy1!=fgxgy)
            {
                     System.out.println("Server 2 is autenticated");

     %>
     <%
```

```
            Object[] quad1=quad(p);

            String s3=quad1[0].toString();
            String s4=quad1[1].toString();

            g1=Double.parseDouble(s3);
            g2=Double.parseDouble(s4);
            b1=1;

             B1=((Math.pow(g1,b1)*Math.pow(g2,pi1))%p);
             String b1s=new String();
             b1s=b1s.valueOf(B1);
             String hb1=compare(b1s);


      response.sendRedirect("http://90.0.0.192:8081/imptwoserver/check2.j
sp?basicprotocol=1&B1="+B1+"&hb1="+hb1);

%>

<%
        }
        else
        {
                System.out.println("Server 2 is unautenticated");
        }
  }

  %>
```

check2.jsp :

```
<%@ include file="hash.jsp" %>
<%@ include file="prime.jsp" %>
<%@ page import="java.sql.*"%>
<%!
String pss,pss1;
String user;
String pass;
```

```
double q=PrimeNumber();  //select the prime
double p=((2*q)+1); //check is this prime
double Q=((2*p)+1); //
double pi;
double pi1;
double pi2;
double g1;
double g2;
double g3=3.0;
double b1;
double B1,B2,B;
double S1,Su,A,S2,sub,sub1,SS;
int flag=0;
double ss3;
double ss4;

%>

<%
Object[] quad1=quad(p);
String s3=quad1[0].toString();
String s4=quad1[1].toString();
g1=Double.parseDouble(s3);
g2=Double.parseDouble(s4);
b1=3;

%>

<%

user=session.getAttribute("uname").toString();
pass=session.getAttribute("pass").toString();

try
{        Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
   Connection
conn=DriverManager.getConnection("jdbc:odbc:twoserver","","");
         Statement st=conn.createStatement();
```

```
      String query="select pass1,pass2 from login where
uname='"+user+"'";

       ResultSet res=st.executeQuery(query);
      while(res.next())
      {
        pss=res.getString(1);
        pss1=res.getString(2);
          pi1=Double.parseDouble(pss);
                flag=1;

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

  %>

  <%

  if((request.getParameter("basicprotocol"))!=null)
  {
          if(flag==1)
          {
                  String pass1=new String();
                  String pass2=new String();
                  double q=PrimeNumber();
                  double hash=Math.abs((pass.hashCode()));
                  double ss=hash%q;
                  if(ss<2)
                  {
                          if(ss==1)
                          {
                                  ss3=0;
                                  ss4=1;

                          }
                  }
                  ss3=(ss/2.0)-2.0;
                  ss4=ss-ss3;
```

```
            pass1=pass1.valueOf(ss3);
            pass2=pass2.valueOf(ss4);
            if(!(pss.equals(pass1)))
            {
                    System.out.println("invalid Password 1");
                    flag=3;
%>

<script>
alert("INVALID Password");
var sURL ="http://90.0.0.191:8001/imptwoserver/login.jsp";
window.location.href = sURL;
</script>

<%

              }
              else if(!(pss1.equals(pass2)))
              {
                      System.out.println("invalid Password 2");


 %>

 <script>
 alert("INVALID Password");
 var sURL ="http://90.0.0.191:8001/imptwoserver/login.jsp";
    window.location.href = sURL;
 </script>



 <%
                      flag=3;
                }
                if(flag!=3)
                {
                        flag=2;
                }
          }

          if(flag==2)
    {       String bpv=request.getParameter("B1");
            B1=Double.parseDouble(bpv);
            String bpv1=request.getParameter("B2");
            B2=Double.parseDouble(bpv1);
            B=(B1*B2)%p;
            System.out.println("the value of B1 is "+B1);
            System.out.println("the value of B is "+B);
            response.sendRedirect("client.jsp?B="+B);

      }
      flag=0;

}
if(flag==0)
 {

 %>

  <script>
  alert("INVALID USER");
  var sURL ="http://90.0.0.191:8001/imptwoserver/login.jsp";
     window.location.href = sURL;
  </script>


   <%
   }

    %>
    <%
    if((request.getParameter("Su"))!=null)
    {       String Su1=request.getParameter("Su");
            Su=Double.parseDouble(Su1);
            String A1=request.getParameter("A");
            A=Double.parseDouble(A1);
            S1=(Math.pow(A,b1))%p;
            sub=S1;
            S1=(Math.pow(g3,S1))%p;
            System.out.println("b1 = "+b1);
            System.out.println("pi1 = "+pi1);
            System.out.println("g1 = "+g1);
```

```
        System.out.println("g2 = "+g2);
        System.out.println("the value of S1 is "+S1);
        System.out.println("the value of A is "+A);
        System.out.println("the value of Su is "+Su);

         response.sendRedirect("http://90.0.0.192:8081/imptwoserver/check2.j
sp?Su="+Su+"&A="+A+"&S1="+S1);

}


 %>

 <%
 if((request.getParameter("Su1"))!=null)
 {
         String Su2=request.getParameter("Su1");
         double Su3=Double.parseDouble(Su2);
         String S111=request.getParameter("S11");
         double S1111=Double.parseDouble(S111);
         String S211=request.getParameter("S21");
         double S2111=Double.parseDouble(S211);
         sub1=S2111*sub;
         if(Su3==(Math.pow(g3,sub1)))
         {
                 System.out.println("valid Client");
                 System.out.println("The Key Value at the SS is......"+sub1);
                 Object obs=session.getAttribute("uname");
                 String uu=obs.toString();

    %>

    <html><body><br><br>
    <%


    %>
    <script>
    alert("VALID Password");
    var sURL ="http://90.0.0.191:8001/imptwoserver/valid.jsp";
        window.location.href = sURL;
</script>

<%
        }
        else
        {
                System.out.println("INvalid Client");
        }
 }


 %>
```

check3.jsp :

```
<%@ include file="hash.jsp" %>
<%@ include file="enc.jsp" %>
<%@ include file="rand.jsp" %>
<%@ include file="quad.jsp" %>
<%@ page import="java.sql.*"%>
<%@ page import="java.net.*"%>
<%!
String user;
String server;
String pass;
double x=rand();//non - reputed values
double a=37;//Large Prime Number
double n=1234;//registered value
String fx1,y,gx,gy,gxy,gfxy,hgx,hgy,hgxy,hgfxfy;
double fx2,fx;
double k,y1,gx1,gy1,gxy1,fy,fxy,fgxy,fgxgy,gfxy1,fxs,fys,fxys;
String yd,xe,ydd;
double xe1,y4;
%>

 <%
 server=request.getParameter("server2");
 if(server!=null)
  {
        System.out.println("Response Is Give by a Server 2 and His Address
is "+server);
        fx1=request.getParameter("fx");

        fx2=Double.parseDouble(fx1);
        y=request.getParameter("y");

        xe=request.getParameter("x");

        xe=xe.replaceAll(" ","+");

        try
        {
                byte[] keyBytes1=new byte[8];
         String baseDir = this.getServletContext().getRealPath("/");
                File webRootDirectory = new File(baseDir);
                File targetFile = new File(webRootDirectory,
  "imptwoserver//enc1.dat");
                BufferedInputStream buf2=new BufferedInputStream(new
  FileInputStream(targetFile));
                buf2.read(keyBytes1);
                int le=keyBytes1.length;

                SecretKeySpec key2 = new SecretKeySpec(keyBytes1,
  "DES");
                DesEncrypter(key2);

          yd = decrypt(xe);
          ydd=decrypt(y);

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

          gx=request.getParameter("gx");
          gy=request.getParameter("gy");

          gxy=request.getParameter("gxy");
```

```
    hgx=request.getParameter("hgx");
    hgy=request.getParameter("hgy");
    hgxy=request.getParameter("hgxy");
    xe1=Double.parseDouble(yd);
    String hgxs=compare(gx);
    String hgys=compare(gy);
    String hgxys=compare(gxy);

    if(hgxs.equals(hgx))
    {
            y1=Double.parseDouble(yd);
            y4=Double.parseDouble(ydd);
            gx1=Double.parseDouble(gx);
            gy1=Double.parseDouble(gy);
            gxy1=Double.parseDouble(gxy);
            k=(fx2*gy1)%n;
            if(!(xe1==x))
            {
                    System.out.println("Intrusion Attempted");
%>
<html><title>Intrusion Occured</title><body><h1>Intrusion Occured in
server <h1></body></html>
<%
            }
            else
            {
                    System.out.println("Intrusion Not Attempted");
                    fy=(Math.pow(y4,a))%n;
                    fxy=(Math.pow((x*y4),a))%n;
                    fgxy=(Math.pow(gxy1,a))%n;
                    String fgxys=new String();
                    fgxys=fgxys.valueOf(fgxy);
                    String hfgxy=compare(fgxys);

        response.sendRedirect("http://90.0.0.192:8081/imptwoserver/check1.j
sp?fy="+fy+"&fxy="+fxy+"&fgxy="+fgxy+"&y="+ydd+"&fx="+fx2+"&g
x="+gx+"&gy="+gy+"&hfgxy="+hfgxy);
            }
        }
        else
        {
                System.out.println("Intrusion Attempted");
        }
}
%>
```

client.jsp :

```
<%@ include file="prime.jsp" %>
<%!
String user;
String s;
double q=PrimeNumber();
double p=((2*q)+1);
double Q=((2*p)+1);
double a=3;
double g1;
double g2;
double g3=3.0;
double b1;
double b2;
double B1,B2,B;
double S1,Su,A,S2,res;



 %>
 <%

 Object[] quad1=quad(p);

 String s3=quad1[0].toString();
 String s4=quad1[1].toString();

 g1=Double.parseDouble(s3);
 g2=Double.parseDouble(s4);

 b1=3;

 %>
 <%
```

```
System.out.println("a :"+a);
System.out.println("q :"+q);
System.out.println("p :"+p);
System.out.println("Q :"+Q);
A=(Math.pow(g1,a));
A=A%p;
System.out.println("A :"+A);
Su=(Math.pow(g1,(a*(b1+b2))))%p;
res=Su;
Su=(Math.pow(g3,Su))%p;
System.out.println("The Value Of Su...Improved......"+Su);

response.sendRedirect("check2.jsp?Su="+Su+"&A="+A);
%>


<%

System.out.println("The Key Value in client is......"+res);

%>
```

enc.jsp :

```
<%@ page import="javax.crypto.spec.IvParameterSpec"%>
<%@ page import="java.security.spec .*"%>
<%@ page import="java.io.*"%>
<%@ page import="javax.crypto.*"%>
<%@ page import="javax.crypto.CipherInputStream"%>

<%!
Cipher ecipher;
Cipher dcipher;
public void DesEncrypter(SecretKey key)
{
        try
        {
                ecipher = Cipher.getInstance("DES");
        dcipher = Cipher.getInstance("DES");
```

```
    ecipher.init(Cipher.ENCRYPT_MODE, key);
    dcipher.init(Cipher.DECRYPT_MODE, key);
    }
    catch (javax.crypto.NoSuchPaddingException e)
    { }
    catch (java.security.NoSuchAlgorithmException e)
    { }
    catch (java.security.InvalidKeyException e)
    { }
}

public String encrypt(String str)
{
        try
    {
        byte[] utf8 = str.getBytes("UTF8");
        byte[] enc = ecipher.doFinal(utf8);

       return new sun.misc.BASE64Encoder().encode(enc)
         }
         catch (javax.crypto.BadPaddingException e)
         { }
         catch (IllegalBlockSizeException e)
         { }
         catch (UnsupportedEncodingException e)
         { }
         catch (java.io.IOException e)
         { }
      return null;
    }

    public String decrypt(String str)
    {
      try
      {
          byte[] dec = new sun.misc.BASE64Decoder().decodeBuffer(str);
        byte[] utf8 = dcipher.doFinal(dec);
        return new String(utf8, "UTF8");
      }
      catch (javax.crypto.BadPaddingException e)
```

```
{
   System.out.println(e);
}
catch (IllegalBlockSizeException e)
{
   System.out.println(e);
}
catch (UnsupportedEncodingException e)
{
   System.out.println(e);
}
catch (java.io.IOException e)
{
   System.out.println(e);
}
return null;
}

%>
```

hash.jsp :

```
<%@ page language="java" %>
<%@ page import="javax.crypto.*"%>
<%@ page import="java.security.*"%>
<%@ page import="java.io.*"%>
<%@ page import="java.security.spec .*"%>
<%@ page import="javax.crypto.spec.*"%>
<%!
public String compare(String msg)
{
         String digestB64=new String();
         try
         {
         byte[] keyBytes1=new byte[8];

                  String baseDir = this.getServletContext().getRealPath("/");

                  File webRootDirectory = new File(baseDir);
```

```
            File targetFile = new File(webRootDirectory,
"imptwoserver//enc.dat");

            BufferedInputStream buf2=new BufferedInputStream(new
FileInputStream(targetFile));
            buf2.read(keyBytes1);
            int le=keyBytes1.length;

            SecretKeySpec key = new SecretKeySpec(keyBytes1,
 "HmacMD5");
        Mac mac = Mac.getInstance(key.getAlgorithm());
        mac.init(key);

        String str = "This message will be digested";

        byte[] utf8 = msg.getBytes("UTF8");
        byte[] digest = mac.doFinal(utf8);

        digestB64 = new sun.misc.BASE64Encoder().encode(digest);


        }
        catch (InvalidKeyException e)
        { }
        catch (NoSuchAlgorithmException e)
        { }
        catch (UnsupportedEncodingException e)
        { }
        catch (Exception e)
        { }
        return digestB64;
    }
    %>
```

login.jsp :

```
<html>

<head>
<meta http-equiv="Content-Language" content="en-us">
```

```
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
<title>LOGIN FORM</title>
</head>

<body bgcolor="#CCFFFF">

<p>&nbsp;</p>
<p align="center"><b><font color="#FF0000" size="5">LOGIN
FORM</font></b></p>
<center>
<fieldset style="width: 338px; height: 100px; padding: 2">
<legend>
<p align="center"><b><font color="#0000FF" size="4">Sign
In</font></b></p>
</legend>
<form method="POST" action="check1.jsp">
        <!--webbot bot="SaveResults" U-File="D:\Documents and
Settings\Administrator\Desktop\_private\form_results.csv" S-
Format="TEXT/CSV" S-Label-Fields="TRUE" -->
<p align="center">&nbsp;<p align="center">&nbsp;<font size="4"
type="times new roman" color="black">&nbsp;
        USER NAME&nbsp;&nbsp; <input type="text" name="T1"
size="20" tabindex="1"></font><p align="center">
        <font size="4">&nbsp;&nbsp; PASSWORD&nbsp;&nbsp;&nbsp;
        <input type="password" name="T2" size="20"
tabindex="2"></font><p align="center">
        <input type="submit" value="Submit" name="B1"
tabindex="3"></form>
<p align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;<a
href="newuser.jsp">NewUser</a>&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;</p>

</fieldset><p align="center">&nbsp;</p>
</center>
</body>

</html>
```

newuser.jsp :

```
<%@ page import="java.sql.*"%>
<%@ include file="rand.jsp" %>
<%@ include file="prime.jsp" %>
<html>

<head>
<meta http-equiv="Content-Language" content="en-us">
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
<title>NEW USER FORM</title>
</head>

<body bgcolor="#CCFFFF">

<p>&nbsp;</p>
<p align="center"><b><font color="#FF0000" size="5">NEW USER
FORM</font></b></p>
<center>
<fieldset style="width: 338px; height: 100px; padding: 2">
<legend>
<p align="center"><b><font color="#0000FF" size="4">Sign
UP</font></b></p>
</legend>
<form method="POST" action="newuser.jsp">
        <!--webbot bot="SaveResults" U-File="D:\Documents and
Settings\Administrator\Desktop\_private\form_results.csv" S-
Format="TEXT/CSV" S-Label-Fields="TRUE" -->
<p align="center">&nbsp;<p align="center">&nbsp;<font size="4"
type="times new roman" color="black">&nbsp;
        USER NAME&nbsp;&nbsp; <input type="text" name="T1"
size="20" tabindex="1"></font><p align="center">
        <font size="4">&nbsp;&nbsp; PASSWORD&nbsp;&nbsp;&nbsp;
        <input type="password" name="T2" size="20"
tabindex="2"></font><p align="center">
        <input type="submit" value="Submit" name="B1"
tabindex="3"></form>
<p
align="center">&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&nbsp;&
nbsp;&nbsp;</p>
```

```
</fieldset><p align="center">&nbsp;</p>
</center>
</body>

</html>
<%!
double ss3;
double ss4;

%>

<%
if((request.getParameter("T2")!=null) &&
(request.getParameter("T1")!=null))
{
        String uname=request.getParameter("T1");
        String pass=request.getParameter("T2");
        String pass1=new String();
        String pass2=new String();
        double q=PrimeNumber();
        System.out.println("Random Number:"+q);
        System.out.println("Pass.hashCode():"+pass.hashCode());
        double hash=Math.abs((pass.hashCode()));
        System.out.println("Hash"+hash);
        double ss=hash%q;
        if(ss<2)
        {
                if(ss==1)
                {
                        ss3=0;
                        ss4=1;
                }
        }
        ss3=(ss/2.0)-2.0;
        ss4=ss-ss3;
        pass1=pass1.valueOf(ss3);
        pass2=pass1.valueOf(ss4);
        try
        {
                Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
                Connection
conn=DriverManager.getConnection("jdbc:odbc:twoserver","","");
                Statement st=conn.createStatement();
                String query="insert into login
values('"+uname+"','"+pass1+"','"+pass2+"')";

        int res =st.executeUpdate(query);
        System.out.println("new User Success fully Created");
        %>

  <script>
  alert("User Enrolled");

  </script>
  <%


  response.sendRedirect("http://90.0.0.192:8081/imptwoserver/newuser.jsp?T
  1="+uname+"&T2="+pass);
        }
        catch(Exception e)
        {
   %>

    <alert>

          </script>
    <%          System.out.println("The Exception in New user is:"+e);
              }
    }

    %>
```

prime.jsp :

```
<%@ include file="quad.jsp" %>
<%!
public  boolean isPrime ( int num )
```

```
{
     boolean prime = true;
  int limit = (int) Math.sqrt ( num );
  for ( int i = 2; i <= limit; i++ )
  {
      if ( num % i == 0 )
      {
      prime = false;
                 break;
      }
   }
   return prime;
 }

 public int PrimeNumber()
 {
      int p,Q;
      int ii=0;
      int val[]=new int[5];

    for ( int i = 2; i <= 40; i++ )
    {
            if ( isPrime ( i) )
            {
                 p=(2*i)+1;
                 if ( isPrime (p) )
                 {
                      Q=(2*p)+1;
                      if ( isPrime (Q) )
                      {
                           double cc=(double)p;
                           Object[] oo=quad(cc);
                           int len=oo.length;
                           val[ii]=i;
                           ii++;
                      }
                 }
            }
       }
       return val[2];
}
%>
```

rand.jsp :

```
<%@page import="java.util.*" %>
<%!
public double rand()
{
        Random r=new Random(1000);
        int a=r.nextInt(1000);
        double e=5678;
        return e;
}
public double rand1()
{
        Random r=new Random(1000);
        int a=r.nextInt(1000);
        double d=4612;
        return d;
}
%>
```

valid.jsp :

```
<html>

<head>
<meta name="GENERATOR" content="Microsoft FrontPage 5.0">
<meta name="ProgId" content="FrontPage.Editor.Document">
<meta http-equiv="Content-Type" content="text/html; charset=windows-
1252">
<title>New Page 1</title>
</head>

<body>
<center><u><b><font color="#800000" size="6">WELCOME TO
KUMARAGURU COLLEGE OF
TECHNOLOGY</font></b></u></center>
```

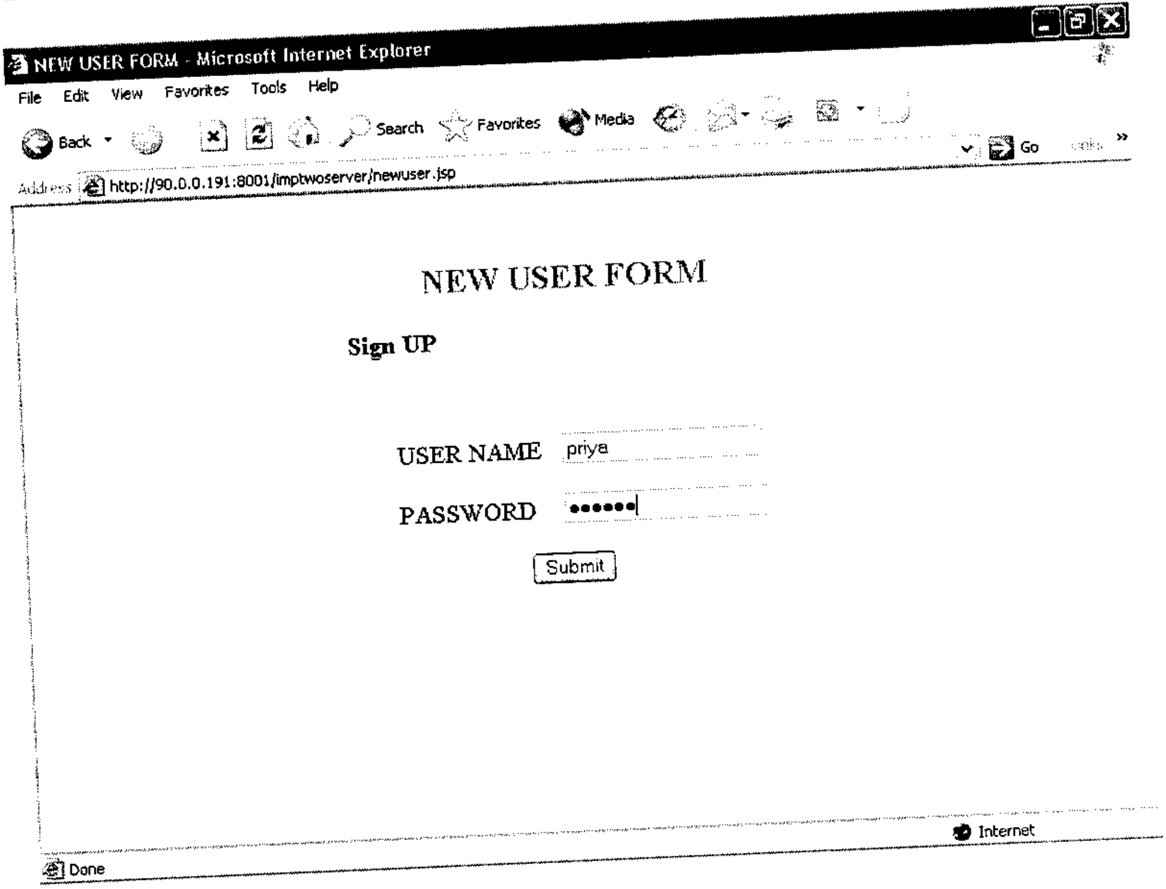
```
</body>

</html>
```

quad.jsp :

```
<%@ page import="java.util.*" %>
<%!
int ss1=0;
public Object[] quad(double a)
{
        Vector v=new Vector();
        for(int i=0;i<a;i++)
        {
                for(int j=0;j<a;j++)
                {
                        double ss=(j*j)-i;
                        if(ss!=0 && ((ss%a)==0.0))
                        {
                                if(ss1==i)
                                { }
                                else
                                {
                                        String s=new String();
                                        s=s.valueOf(i);
                                        Object obj=(Object)s;
                                        v.add(obj);
                                        ss1=i;
                                }
                        }
                }
        }
        Object[] ob=v.toArray();
        System.out.println("check"+ob.length);
        return ob;
    }
    %>
```

SCREENSHOT:

1. NEW USER FORM:



NEW USER FORM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back
Search
Favorites
Media
Go
Address
Submit
Done
Internet

http://90.0.0.191:8001/imptwoserver/newuser.jsp

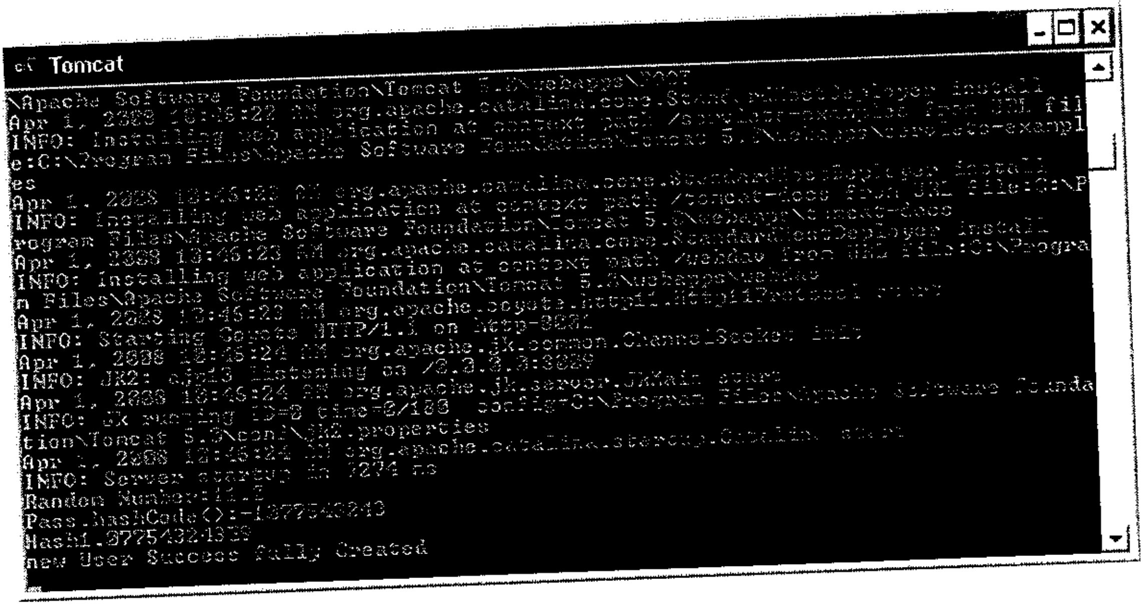
NEW USER FORM

Sign UP

USER NAME priya

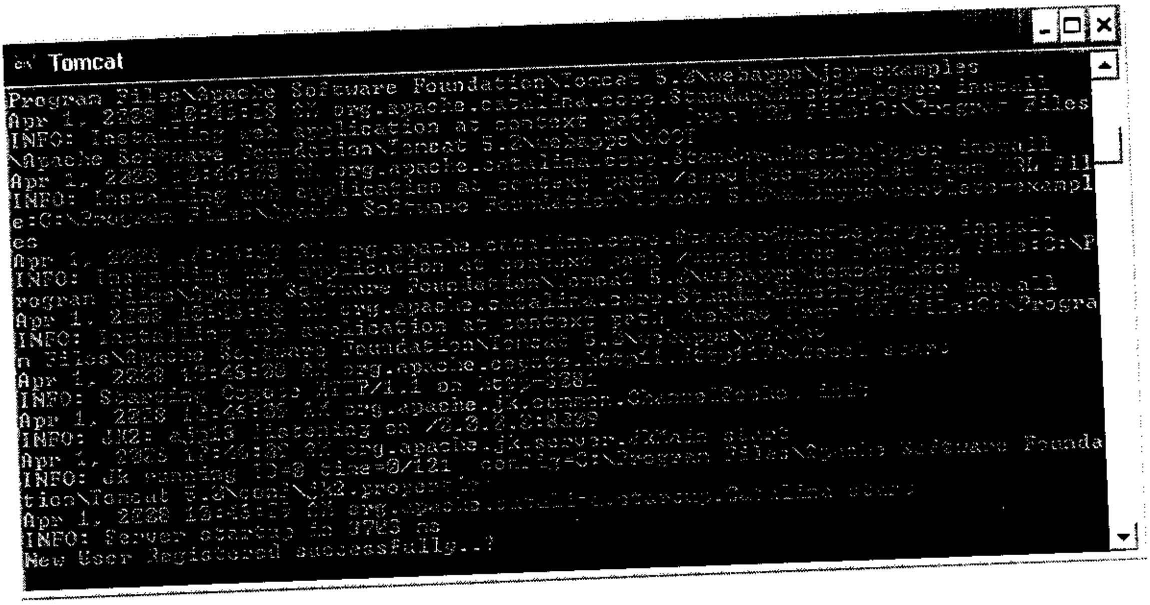
PASSWORD ●●●●●●

2. SERVICE SERVER:



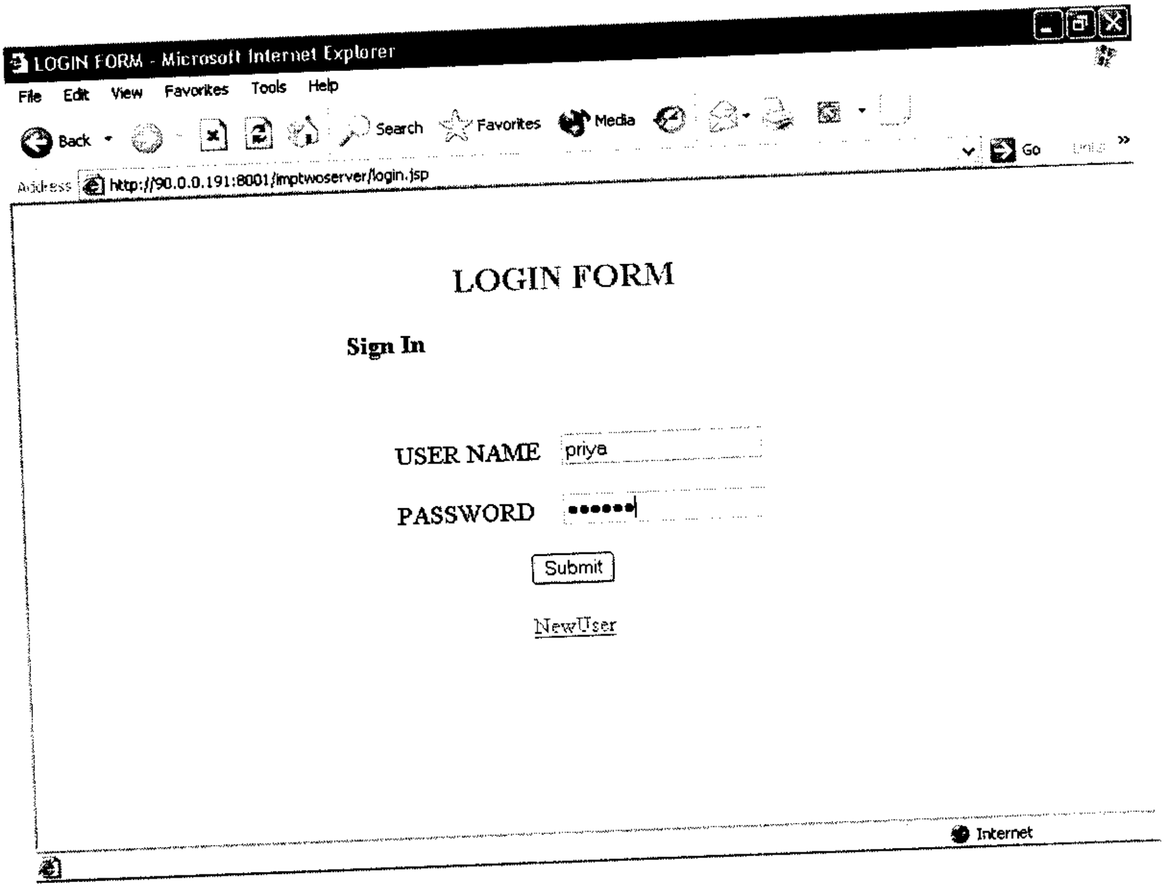
```
\Apache Software Foundation\Tomcat 5.0\webapps\ROOT
Apr 1, 2008 10:46:22 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /servlets-examples from URL fil
e:C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\servlets-exampl
es
Apr 1, 2008 10:46:23 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /tomcat-docs from URL file:C:\P
rogram Files\Apache Software Foundation\Tomcat 5.0\webapps\tomcat-docs
Apr 1, 2008 10:46:23 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /webdav from URL file:C:\Progra
m Files\Apache Software Foundation\Tomcat 5.0\webapps\webdav
Apr 1, 2008 10:46:23 AM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8081
Apr 1, 2008 10:46:24 AM org.apache.jk.common.ChannelSocket init
INFO: JK2: ajp13 listening on /0.0.0.0:8009
Apr 1, 2008 10:46:24 AM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/188  config=C:\Program Files\Apache Software Founda
tion\Tomcat 5.0\conf\jk2.properties
Apr 1, 2008 10:46:24 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 7274 ms
Random Number:11.2
Pass.hashCode():-1377543243
Hash1.3775432433
new User Success fully Created
```

3. CONTROL SERVER:



```
Program Files\Apache Software Foundation\Tomcat 5.0\webapps\jsp-examples
Apr 1, 2008 10:46:18 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path  from URL file:C:\Program Files
\Apache Software Foundation\Tomcat 5.0\webapps\ROOT
Apr 1, 2008 10:46:18 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /servlets-examples from URL fil
e:C:\Program Files\Apache Software Foundation\Tomcat 5.0\webapps\servlets-exampl
es
Apr 1, 2008 10:46:18 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /tomcat-docs from URL file:C:\P
rogram Files\Apache Software Foundation\Tomcat 5.0\webapps\tomcat-docs
Apr 1, 2008 10:46:18 AM org.apache.catalina.core.StandardHostDeployer install
INFO: Installing web application at context path /webdav from URL file:C:\Progra
m Files\Apache Software Foundation\Tomcat 5.0\webapps\webdav
Apr 1, 2008 10:46:20 AM org.apache.coyote.http11.Http11Protocol start
INFO: Starting Coyote HTTP/1.1 on http-8081
Apr 1, 2008 10:46:20 AM org.apache.jk.common.ChannelSocket init
INFO: JK2: ajp13 listening on /0.0.0.0:8009
Apr 1, 2008 10:46:20 AM org.apache.jk.server.JkMain start
INFO: Jk running ID=0 time=0/121  config=C:\Program Files\Apache Software Founda
tion\Tomcat 5.0\conf\jk2.properties
Apr 1, 2008 10:46:20 AM org.apache.catalina.startup.Catalina start
INFO: Server startup in 3703 ms
New User Registered successfully..!
```

4. LOGIN FORM:


LOGIN FORM - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites Media
Address http://90.0.0.191:8001/imptwoserver/login.jsp
Go
Internet

LOGIN FORM

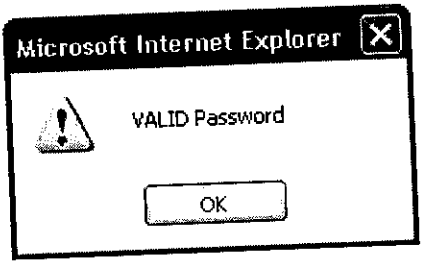
Sign In

USER NAME priya

PASSWORD ●●●●●●

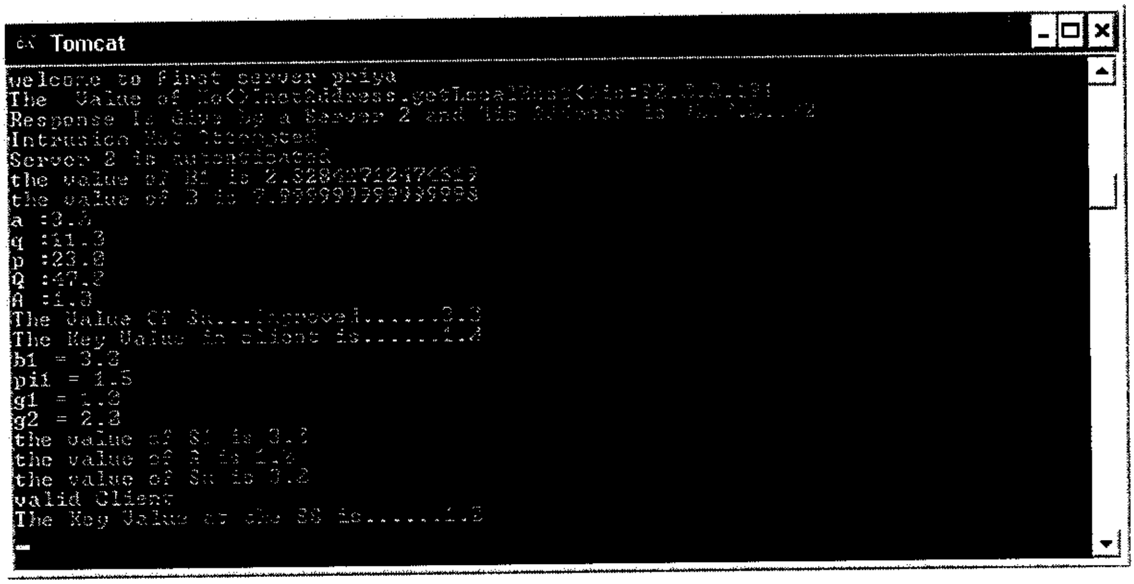
Submit

NewUser



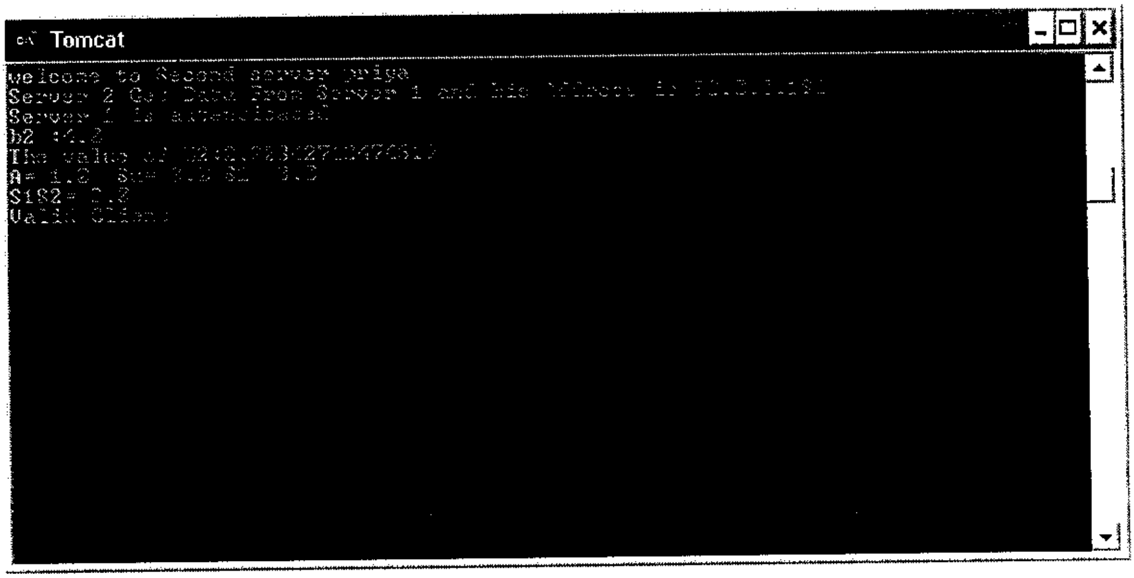
VALID Password

5. SERVICE SERVER:



```
welcome to first server priya
The  Value of Ho()InetAddress.getLocalHost()is=[illegible]
Response Is Give by a Server 2 and its Address is [illegible]
Intrusion Not Attempted
Server 2 is autenticated
the value of B1 is 2.32842712474619
the value of B is 7.99999999999998
a :3.5
q :11.3
p :23.0
Q :47.2
A :1.3
The Value Of Su...Improved......3.3
The Key Value in client is......1.2
b1 = 3.2
pi1 = 1.5
g1 = 1.0
g2 = 2.2
the value of S1 is 3.2
the value of A is 1.2
the value of Su is 3.2
valid Client
The Key Value at the SS is......1.2
```

6. CONTROL SERVER:



```
welcome to Second server priya
Server 2 Get Data From Server 1 and his Address is [illegible]
Server 1 is autenticated
b2 :4.2
The value of B2=2.82842712474619
A= 1.2  Su= 3.2 S1 3.2
S1S2= 3.2
Valid Client
```

REFERENCES

REFERENCES

[1] Jim Keogh, “J2EE, The Complete Reference”, Tata McGraw Hill, Edition 2002.

[2] Pallavi Jain and Shadab Siddiqui, “J2EE Professional Projects”, Prentice-Hall of India Private Limited, Edition 2002.

[3] Paul Whitehead, “JavaServer Pages:Your Visual Blueprint to Designing Dynamic Contents With JSP”, Hungry Minds Inc, Edition 2001.

[4] J. Brainard, A. Juels, B. Kaliski, and M. Szydlo, “A New Two-Server Approach for Authentication with Short Secrets,” Proc. USENIX Security Symp., 2003.

[5] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated Key Exchange Secure Against Dictionary Attacks,” Advances in Cryptology (Eurocrypt ’00), pp. 139-155, 2000.

[6] D.P. Jablon, “Password Authentication Using Multiple Servers,” RSA Security Conf., pp. 344-360, 2001.

[7] J. Katz, R. Ostrovsky, and M. Yung, “Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords,” Proc. Advances in Cryptology (Eurocrypt ’01), pp. 475-494, 2001.

[8] http://tomcat.apache.org/

[9] http://www.webopedia.com/

[10] http://java.sun.com/