WEB SECURITY

Project report submitted in partial fulfilment of the

Requirement for the of degree of

Bachelor of Technology

In

Computer Science Engineering

By:

Rahul Tiwari	**091233**
Surbhi Jain	**091242**
Nikita Gupta	**091253**
Sweta Singhal	**091288**

Under the supervision of

Mrs. Rajni Mohana

Jaypee University Of Information Technology

Waknaghat , Solan-173234, Himachal Pradesh

CERTIFICATE

This is to certify that the work titled "**WEB SECURITY**" submitted by "**Rahul Tiwari, Surbhi Jain, and Nikita Gupta**" in partial fulfillment for the award of degree of **B.Tech.** of Jaypee University of Information Technology, Waknaghat has been carried out under my supervision. This work has not been submitted partially or wholly to any other University or Institute for the award of this or any other degree or diploma.

Signature of Supervisor	.... [illegible]
Name of Supervisor(s)	Prof.,Brig(Retd) Dr S P Ghrera (HOD – CSE\IT Dept.)
	Mrs RajniMohana.(Senior Lecturer)
Date	1st ~~December~~ May 2012

Certified that this work has not been submitted partially or fully to any other University or Institute for the award of this or any other degree or diploma.

1. **Rahul Tiwari**
   (091233)
2. **Surbhi Jain**
   (091242)
3. **Nikita Gupta**
   (091253)

ACKNOWLEDGEMENT

We thank our project guide Mrs. Rajni Mohana for providing us with the opportunity of undertaking our final year project under her. We thank her for her guidance, careful supervision, critical suggestions and encouragement that has proved to be a source of immense help and supervision.

We also wish to thank our H.O.D. Dr.Prof.S.P. Ghrera for making us competent in c, c++ that we were able to make our project.

Date: 1st May, 2013

Rahul Tiwari

Surbhi Jain

Nikita Gupta

CONTENTS

ABSTRACT

The World Wide Web is widely used by businesses, government agencies, and many individuals. But the Internet and the Web are extremely vulnerable to compromises of various sorts, with a range of threats as shown. These can be described as passive attacks including eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted, and active attacks including impersonating another user, altering messages in transit between client and server, and altering information on a Web site. The web needs added security mechanisms to address these threats.

1. INTRODUCTION

1.1 History

The World Wide Web is widely used by businesses, government agencies, and many individuals. But the Internet and the Web are extremely vulnerable to compromises of various sorts, with a range of threats as shown. These can be described as passive attacks including eavesdropping on network traffic between browser and server and gaining access to information on a Web site that is supposed to be restricted, and active attacks including impersonating another user, altering messages in transit between client and server, and altering information on a Web site. The web needs added security mechanisms to address these threats.

Possible threats to-

1. Integrity:

   It provides the guarantee that data is protected from accidental or deliberate (malicious) modification. Integrity is a key concern, particularly for data passed across networks. This involves the threat to the user data or the valuable content of the person or the organization.
   This threat has very serious consequences like loss of valuable information, compromise of the machine secrecy, thus increasing the vulnerability of the web security.

Countermeasures:
Integrity for data in transit is typically provided by using hashing techniques and message authentication codes using the cryptographic checksums.

2. Confidentiality:

   Confidentiality, also referred to as *privacy*, is the process of making sure that data remains private and confidential, and that it cannot be viewed by unauthorized/illegitimate users .it involves the eavesdropping on the net, theft of info from the net, theft of the data from the client, and the information about the network configuration.
   This threat has very serious consequences like loss of information, loss of privacy.

   Countermeasures:
   Encryption is frequently used to enforce confidentiality. Access control lists are another means of enforcing confidentiality.

3. Authentication:

   It is the process of uniquely identifying the clients or the legitimate user of the web applications and services. These might be end users, other services, processes, or computers.
   In this only certain users have the right to control the process like sending, receiving or modification of the data.
   It has serious consequences like misinterpretation of the user which leads to the belief that false information is valid.

   Countermeasures:
   Cryptographic techniques are frequently used, concept of digital signature is also widely used to provide the authenticity.

4. Authorization:

   It is the process that governs the web data, applications, resources and various operations involved.in this only authenticated client is permitted to access. Resources include files, databases, tables, rows, etc., together with system-level resources such as registry keys and configuration data. Operations include performing transactions like purchasing a product, transferring money from one account to another etc.

It has serious consequences like some unauthorized user may access or control the data or account leading to mishandling of accounts.

Countermeasures:
Cryptographic techniques and the access control list are widely used.

5. Availability:

From a security perspective, availability means that systems remain available for legitimate users. Otherwise the illegitimate or the unauthorized user may flood the system with bogus requests by killing the user threads and leading to filling up the disk or memory space.
It has consequences like it becomes very annoying and disruptive for the legitimate users, preventing them from getting their work done.

Countermeasures:
This type of attack is very difficult to prevent but techniques like tracking of the source IP addresses and use of special routers can be helpful.

1.2. Project Objective

- To study the present web security concerns and threats.
- To study and analyze the working and algorithms of security solutions currently available.
- To implement the security solutions in an application that assures safe and productive web use.

1.3. Software specifications and requirements

1.3.1 Non Functional Requirements

- Efficiency (resource consumption for given load)
- Effectiveness (resulting performance in relation to effort)

- Robustness
- Security or Factor of safety
- Stability
- Testability

1.3.2 Software requirements

1. JAVA compatible running platform – Java EE Eclipse.
2. Database – MySQL
3. Jre 1.6 or above
4. Web compatible server.

1.3.3 Hardware requirements

1. Switches/Hubs
2. Wires to establish connectivity.

2. LITERATURE REVIEW

2.1. A Survey of Web Security

With no insult intended to the early Web designers, security was an afterthought.
At the outset, the Web's highest goal was seamless availability. Today, with an internationally connected user network and rapidly expanding Web functionality, reliability and security are critical. Vendors engaged in retrofitting security must contend with the Web environment's peculiarities, which include location irrelevance, statelessness, code and user mobility, and stranger-to-stranger communication. In this article, we present a survey of Web-specific security issues. Given the Web's rapid ascent, our offering is necessarily a mix of short-lived techniques and long-lived principles. Our focus is on security in the server and host environments, mobile code, data transport, and anonymity and privacy. We do not delve into cryptography, electronic commerce, or intrusion detection because they are not Web-specific, and they are well covered elsewhere.

SERVER SECURITY

In the client-server environment, we focus on the server side because the server is the central system and the repository of information resources. The server is thus the locus of threats, whereas the client is largely out of sight. Protecting the client side from the server side is generally not an issue, except where client privacy is a concern, as we discuss below. We describe security issues using a Unix-based Apache server as an exemplar. The Apache server is based on the server at the National Center for Supercomputing Applications (NCSA) and is the most widely deployed server (for more on Apache, see http://apache.org). CERN's server is almost parallel to the Apache server and is well documented. Sadly, many commercial servers have diverged from each other and from the Internet's de facto standards. There is no way for us to cover them all. However, regardless of what Web-server software is running, some things are common, such as security configuration, authentication and access issues, and methods of dealing with active content.

Configuration basics
The biggest cause of security problems is bad management. In distributed systems, the first place management affects security is in the system's configuration.
A bad system configuration can mean disaster. If configuration is not controlled, it is difficult to express management policy in the system's operational characteristics. As system complexity increases, the problem becomes acute: The inability to make systems conform to policy ensures increasing disarray and the exploitable holes that result. The Web-server configuration file lives in the server root. Configuration files are composed entirely of directives and explanatory comments. A directive is just a keyword that the HTTP daemon recognizes, followed by arguments. Directives are insensitive to case and white space. They control which file contains user names and passwords and which file

contains group namesand passwords, as well as access to the files in the document tree, including default permissions and how to override them locally.

2.2.Existing attacks to web security:

Types of attacks:

1. Denial of service attack:

   The function of a denial of service attack is fundamentally to flood its target machine with so much traffic that it prevents it from being accessible to any other requests or providing services. The target machine is kept so busy responding to the traffic it is receiving from its attacker that it has insufficient resources to respond to legitimate traffic on the network.
   Denial of service attack attempt to consume all of system resources like-CPU,memory space etc. when any of the resource reaches it limit the system stop responding and web site become inaccessible and usually a message like "web page not responding" or "web page not available" is shown to the users.
   Example - Tatkal ticket booking service of indian railways.

2. Cross-site scripting

   Also known as XSS, is the most common attack on the web sites.it involves the injection or the introduction of the client-side script i.ean attacker can send malicious content from an end-user and collect some type of data from the victim site.
   The main reason for such kind of attack is the breach of the browsers security.
   In XSS attack, the hacker infects a legitimate web page with his malicious client-side script. When a user visits this web page the script is downloaded to his browser and executed allowing the hacker to intrude.

3. SQL injection:

   This attack involves the introduction of SQL statements in a web form entry field in an attempt to get the website to pass a newly formed false SQL command to the database.
   This type of attack is also known as ector attack.
   Example- Introduction of false dataentry table in credit card and banking systems.

4. Man- in- the- middle:

   Also known as the bucket brigade attack, this type of attack involves the active eavesdropping, in which the attacker makes independent connections in between the two parties and relays messages between them, making them believe that the messages are coming from each other. all the cryptographic systems are secure to this attack.
   Example.- person A sends the message to B “meet me at 10 am”, a person C intrude in between the network and modifies to “meet me at 10 pm”.

5. Smurf attack:

   This is a type of denial of service attack, involving the respond to various ICMP requests by the various host system. Thus multiplying the traffic on the network.

6. Buffer overflow:

   This attack involves the introduction of malicious code to the victim’s machine without having any bound checking, which results in the uninterrupted execution of the code leading to filling of system memory and finally resulting in a system crash.

7. Spoofing:

   in this attack a person or program successfully pretends as another by falsifying data and thereby gaining an illegitimate advantage of the system. This is done by using the source IP address of any system and sending message to other victim IP addresses.

8. Brute Force

In cryptography, a **brute-force attack**, or **exhaustive key search**, is a cryptanalytic attack that can, in theory, be used against any encrypted data (except for data encrypted in an information-theoretically secure manner). Such an attack might be utilized when it is not possible to take advantage of other weaknesses in an encryption system (if any exist) that would make the task easier. It consists of systematically checking all possible keys until the correct key is found. In the worst case, this would involve traversing the entire search space.

The key length used in the cipher determines the practical feasibility of performing a brute-force attack, with longer keys exponentially more difficult to crack than shorter ones. A cipher with a key length of *N* bits can be broken in a worst-case time proportional to 2^N and an average time of half that. Brute-force attacks can be made less effective by obfuscating the data to be encoded, something that makes it more difficult for an attacker to recognise when he/she has cracked the code. One of the measures of the strength of an encryption system is how long it would theoretically take an attacker to mount a successful brute-force attack against it.

Brute-force attacks are an application of brute-force search, the general problem-solving technique of enumerating all candidates and checking each one.

9. Birthday Attacks

As the name suggest Birthday attack a **collision** or **clash** is a unique situation that occurs when two distinct pieces of data have the same hash value , checksum , fingerprint, or cryptographic digest. This attack can be used to abuse communication between sender and receiver.The attack depends on the higher likelihood of collisions (as specified earlier) found between random attack attempts and a fixed degree of permutations (pigeonholes), as described in the birthday paradox.

10.Preimage Attack

In cryptography, the **preimage attack** is a classification of attacks based on cryptographic hash functions for finding a message that has a specific hash value as defined. A cryptographic hash function should resist attacks on its preimage (message).

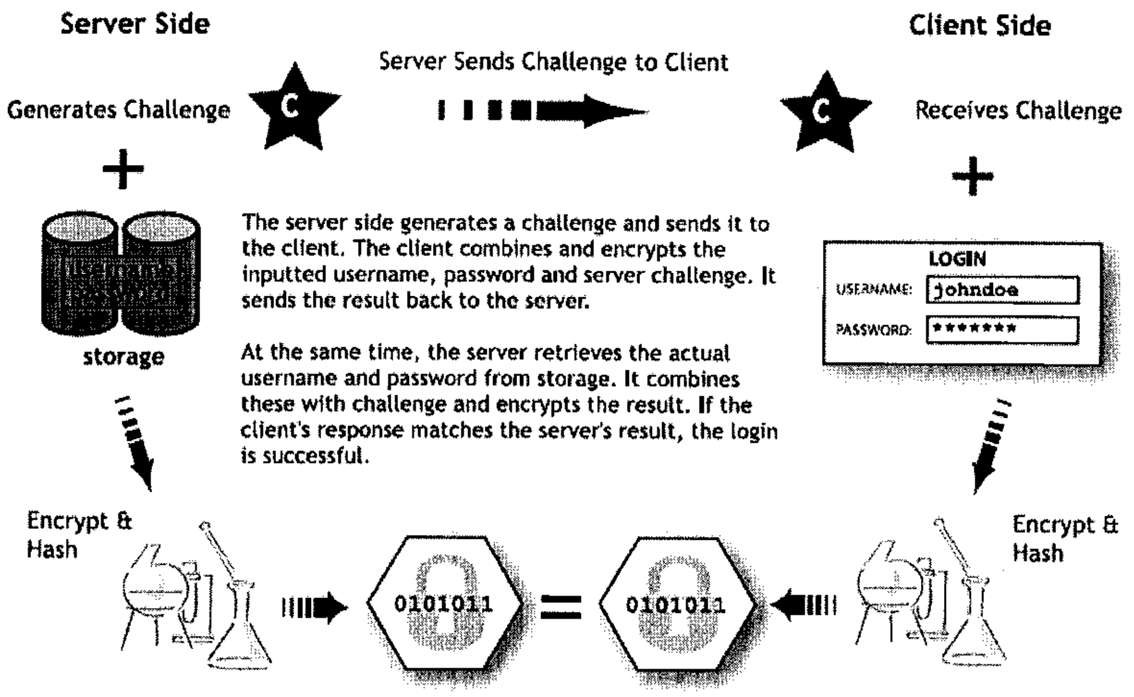
In the context of attack, there are two types of preimage resistance:

- *Preimage resistance*: for essentially all pre-specified outputs, it is computationally infeasible to find any input which hashes to that output, i.e., to find any preimage x′ such that h(x′) = y when given any y for which a corresponding input is not known.
- *second-preimage resistance*: it is computationally infeasible to find any second input which has the same output as any specified input, i.e., given x, to find a 2nd-preimage x' != x such that h(x) = h(x′).

These can be compared with a collision resistance, in which it is computationally infeasible to find any two distinct inputs x, x′ which hash to the same output, i.e., such that h(x) = h(x′).

Collision resistance implies 2nd-preimage resistance of hash functions; and collision resistance does not guarantee preimage resistance.

2.3 Existing Algorithms - Password Protection/Verification



The server side generates a challenge and sends it to the client. The client combines and encrypts the inputted username, password and server challenge. It sends the result back to the server.

At the same time, the server retrieves the actual username and password from storage. It combines these with challenge and encrypts the result. If the client's response matches the server's result, the login is successful.

The diagram as you can see tells the server as well as client side function to provide basic functionality like authentication, integrity, non repudiation, as well as confidentiality.

Server:

Server generates challenge also know as key and sends the same to client. Server already has Username as well as password saved in its storage it simply retrieves the data and encrypt it with challenge (Key).

Server also verifies whether the encrypted message from both server as well as client side is same if verified then only the login is successful.

Client:

Client receives the challenge (Key) generated by server and encrypt it along with username as well as password. Also client sends the encrypted message to server.

Cryptographic hash function:

- An algorithm that takes an arbitrary or variable block of data and returns a fixed-size bit string (in SHA-1 fixed length of 128 bit), the (**cryptographic hash value**), such that an accidental (natural) or intentional (malicious) change to the data or rather the message will (with very high probability) change the hash value H.
- Message: data to be encoded or encrypted.
- Message digest: hash value(encoded or encrypted data).

Hash Algorithms

Algorithm	Output size (bits)	Internal state size[c 1]	Block size	Length size	Word size	Rounds	Best known attacks (complexity:rounds)[c 2] Collision	Second preimage	Preimage
GOST	256	256	256	256	32	256	Yes ([illegible])	Yes ([illegible])	Yes ([illegible])
HAVAL	256/224/192/160/128	256	1,024	64	32	160/128/96	Yes	No	No
MD2	128	384	128	-	32	864	[illegible]	No	Yes ([illegible])
MD4	128	128	512	64	32	48	[illegible]	Yes	Yes
MD5	128	128	512	64	32	64	[illegible]	No	[illegible]
PANAMA	256	8,736	256	-	32	-	Yes	No	No
RadioGatún	Up to 608/1,216 (19 words)	58 words	3 words	-	1–64	-	[illegible]	No	No
RIPEMD	128	128	512	64	32	48	[illegible]	No	No
RIPEMD-128/256	128/256	128/256	512	64	32	64	No	No	No
RIPEMD-160	160	160	512	64	32	80	[illegible]	No	No
RIPEMD-320	320	320	512	64	32	80	No	No	No
SHA-0	160	160	512	64	32	80	[illegible]	No	No
SHA-1	160	160	512	64	32	80	[illegible]	No	No
SHA-256/224	256/224	256	512	64	32	64	[illegible]	No	[illegible]
SHA-512/384	512/384	512	1,024	128	64	80	[illegible]	No	[illegible]
Tiger(2)-192/160/128	192/160/128	192	512	64	64	24	[illegible]	No	[illegible]
WHIRLPOOL	512	512	512	256	8	10	[illegible]	No	No

A Comparison of MD5, SHA-1, and RIPEMD-160

	MD5	SHA-1	RIPEMD-160
Digest length	128 bits	160 bits	160 bits
Basic unit of processing	512 bits	512 bits	512 bits
Number of steps	64 (4 rounds of 16)	80 (4 rounds of 20)	160 (5 paired rounds of 16)
Maximum message size	∞	$2^{64} - 1$ bits	$2^{64} - 1$ bits
Primitive logical functions	4	4	5
Additive constants used	64	4	9
Endianness	Little-endian	Big-endian	Little-endian

Algorithm	Mbps
MD5	32.4
SHA-1	14.4
RIPEMD-160	13.6

Note: Coded by Wei Dai; results are posted at http://www.eskimo.com/~weidai/benchmarks.txt

Secure Hash Algorithm (SHA)

In this section the first topic we will cover is secure hash algorithms (SHA) and we will deal with, what is SHA, Howwe can be design and use it to provide message authentication and digital signature.

I. MOTIVATION

As we know so far, without any method that provides a hint, it is very difficult for the receiver as well as cryptanalyst to identify the legitimate plaintext automatically. And various modes of operation provide no data integrity protection any case. Besides, the whole message has to be encrypted, which causes unnecessary overhead for message authentication and integrity.Error detection code (non-cryptographic checksum) provides redundant data for automatically checking data integrity. However,

(1) Using the code directly can only provide integrity protection for modifications made due to natural causes, but not malicious changes;

(2) Simply encrypting the error detection code is not applicableas the attacker is able to identify the messages that can generate the same error detection code. This means that without knowing the value of the code, the attacker can still change the message without being detected;

(3) Encrypting the entire message still suffers fromvarious attacks.Message authentication code consists of a cryptographic checksum which uses a key in generating the encrypted code. However, the use of MAC (Message authentication code) requires a shared secret key between the communicating parties. It does not provide digital signature.

Now let's start the study of Hash algorithms, which provide a solution for the above problems. Let's first see the non-cryptographic checksum. Their main drawback is that an attacker is able to build a message that matches the non-cryptographic checksum. Based on these observations, we would like to design a "special code" where the original message cannot be regenerated on the basis of its "checksum". This thinking results in the design of hash algorithms.

II. REQUIREMENT FOR HASH FUNCTIONS

A hash functioni.eH takes a message i.eM of variable length and transforms it into a fixed-length hash value h, as mathematically shown below.

$$h = H(M) \quad (1)$$

The hash function H must have the following specified properties:

• *One-way property*: for any given value h, it is computationally infeasible as well as impossible to find $m$such that $H(m) = h$.

• *Weak collision resistance*: for any specified message m, it is computationally infeasible to formulate$y != x$i.e$H(y) = H(x)$

• *Strong collision resistance*: it is computationally infeasible to find any such existing pair (x, y), such that $H(x) = H(y)$.

III. HASH FUNCTION DESIGN

There are several point able similarities in the evolution of both hash functions as well as symmetric block ciphers. Just likesymmetric block ciphers utilizes the various rounds of substitution-permutation network (some ciphers like DES useFeistel cipher, a special case of S-P network) to encrypt each data block of the message, most hash function designs(such that SHA)share the same design structure where multiple rounds of compression functions (which further consist of Round Functions) are applied to each basicprocessing unit (block accordingly 64 bit or 124 bit or 256/ 512 bit).

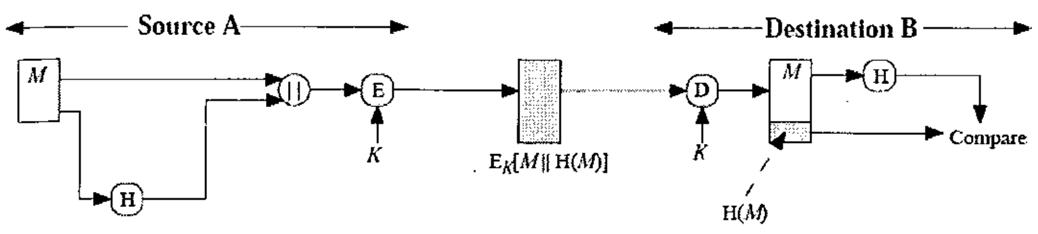
MD5 (Message Digest) and SHA-1 are two most widely and acceptable secure hash algorithms. In particular, the MD5 messagedigest algorithm was developed and coded by Ron Rivest. The algorithm consumes the variable-length input messagein512-bit blocks each, and instead produces a 128-bit fixed length message digest as output. The Secure Hash Algorithm(SHA) originated and was developed by National Institute of Standards and Technology (NIST). It is based on theMD5 algorithm. Based on different digest lengths (the block size), SHA includes algorithms such as SHA-1, SHA-256,SHA-384, and SHA-512 (being the latest).

IV. HASH FUNCTION USAGE

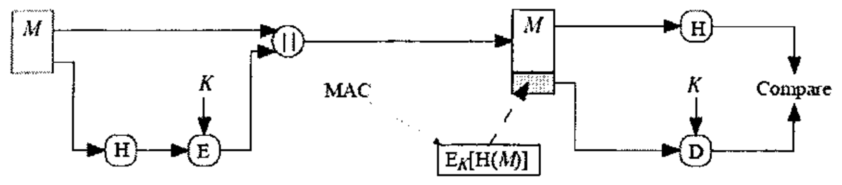
A hash function is basically a function of the input message. It does not use any sort of key. Thus, hash functionsthemselves do not provide security of any kind. However, hash functions can be used along with either of symmetric or asymmetric key (algorithms) to achieve various security goals.

A. HMAC

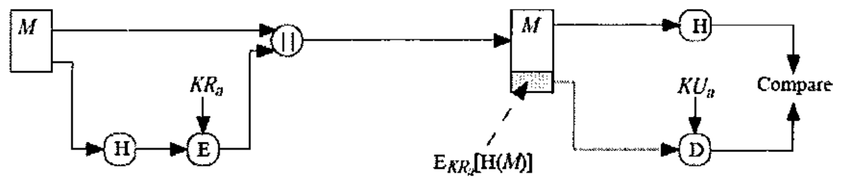
Note that in Fig. 1, no encryption is required for one of the major security concern message authentication. As we saw in the last section hash functions are generally used along with symmetric or asymmetric encryption algorithms for security but hash function generally aremuch faster than symmetric encryption algorithms such as DES and alone provide authentication, this observation shows that hash functionalone can be used to design fast message authentication mechanisms. HMAC is such a message authenticationcode known as Hash message authentication code or HMAC.



(a) Symmetric key encryption: authentication and confidentiality



(b) Symmetric key encryption: authentication



(c) Asymmetric key encryption: authentication and digital signature

Explanation:

Fig (a):

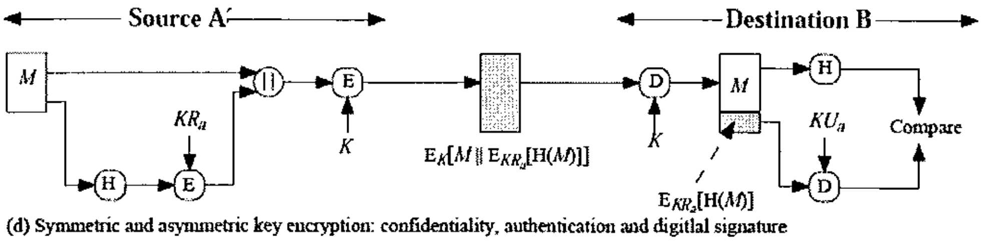
Sender:

As we see in figure (a) we have a message M which is passed through hash function H and then Encrypt using a key K to generate an encrypted message$E K[M \| H(M)]$

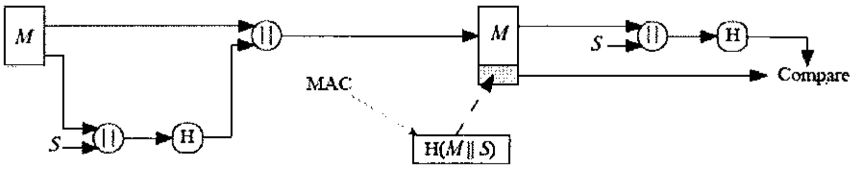
Receiver:

Receives $E K[M \| H(M)]$ and decrypt it using D function forming M the message and H (M) the hashed message now the receiver already posses the hash function H so it passes the decrypted message M from the Hash function H and if the H (M) obtained now is same as the one received then the message is validated.

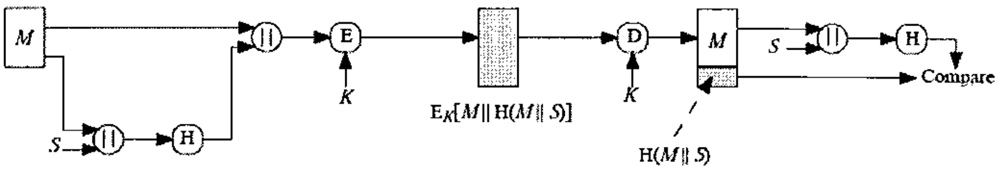
Similarly in other figures the encryption and Hashing along with Private key is introduced to fulfil other functionalities like digital signature as we can see the figure given below it provides confidentiality, authentication and digital signature all together just by introducing two keys(K , KRa , KRb) for encrypting message , private key to encrypt hashed message , public key to decrypt hashed message respectively.



(d) Symmetric and asymmetric key encryption: confidentiality, authentication and digitlal signature



(e) No encryption algorithm is used: authentication



(f) Symmetric key encryption: authentication and confidentiality

Fig. 1. Basic Usages of Hash Functions.

VI. DESIGN

Secure Hash Algorithm

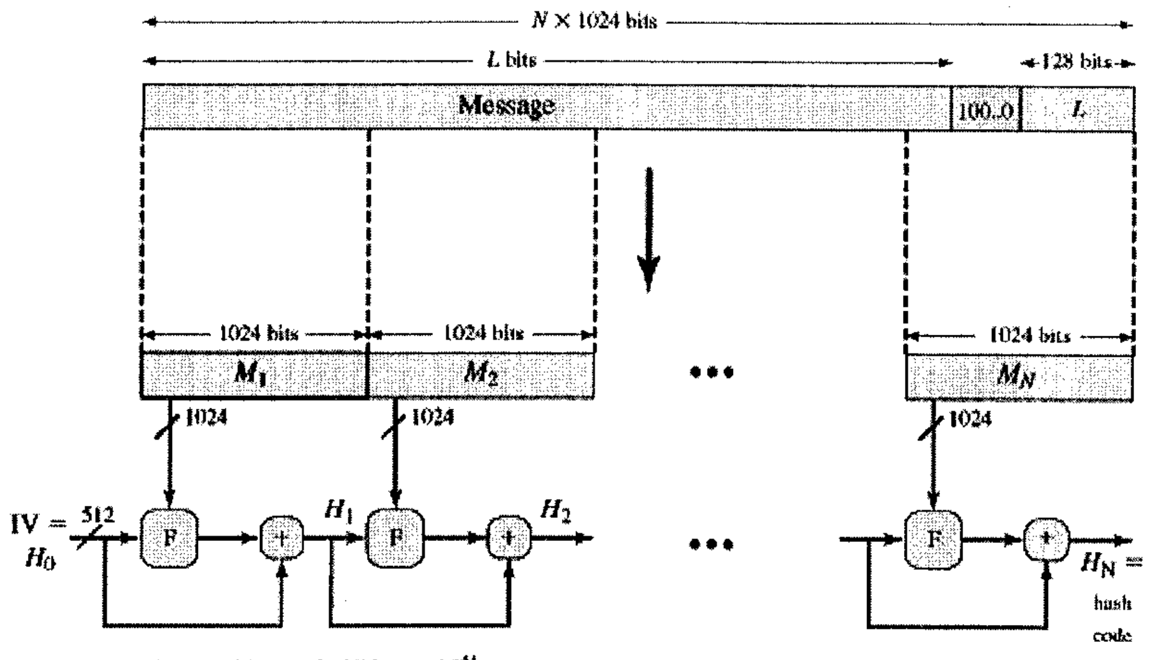
- SHA is based on the hash function MD4 and its design closely models MD4.
- There are three versions of SHA, with hash value lengths of 256, 384, and 512 bits, known as SHA-256, SHA-384, and SHA-512.
- designed for compatibility with increased security provided by the AES cipher

SHA-512 LOGIC

- The algorithm takes as input a message with a maximum length of less than 2^128 bits and produces as output a 512-bit message digest.

- The input is processed in 1024-bit blocks. Figure 12.1 depicts the overall processing of a message to produce a digest.

SHA-512 Overview



Explanation:

As we can see in the above figure we have a message of variable length L and along with this we have two padding bits one comprises of the length of the message 128 bits whereas other consist of parity bits like all bits 0' only the MSB being 1'. As we know the main function of hash function is to take in variable length message and produce fixed length bits like in case of SHA-1 1024 bits each. The function F is Round function discussed later.

In detail....

Now examine the structure of SHA-512, noting that the other versions are quite similar. The processing consists of the following steps:

• Step 1: Append padding bits

• Step 2: Append length

• Step 3: Initialize hash buffer

• Step 4: Process the message in 1024-bit (128-word) blocks, which forms the heart of the algorithm

• Step 5: Output the final state value as the resulting hash

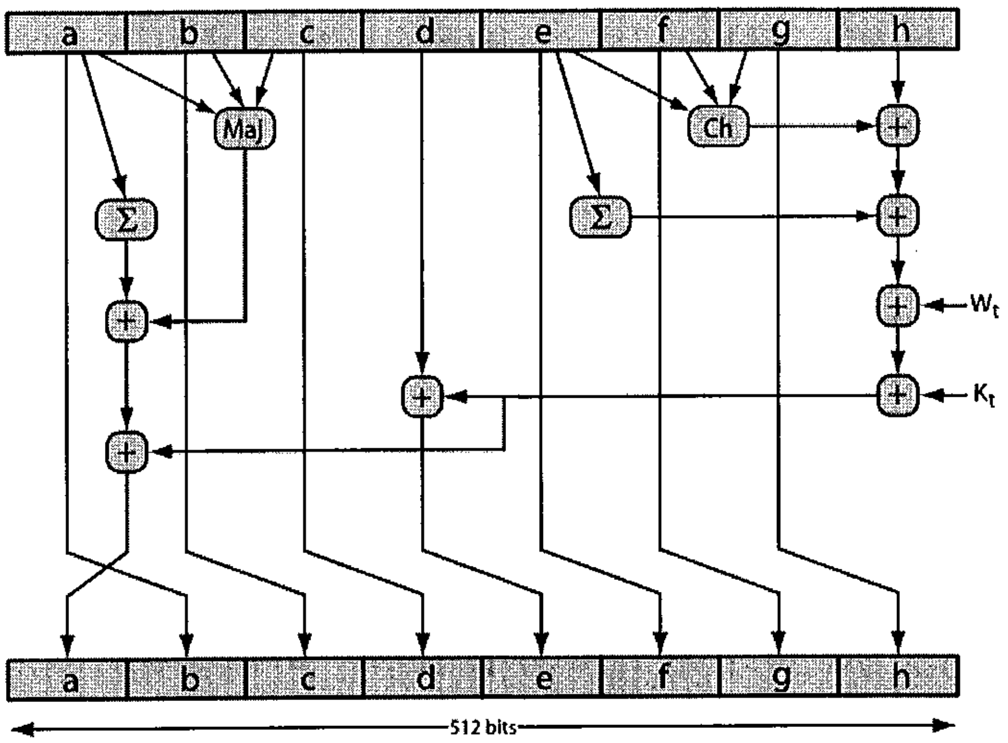
SHA-512 Compression Function

- heart of the algorithm
- processing message in 1024-bit blocks
- consists of 80 rounds
    - updating a 512-bit buffer
    - using a 64-bit value Wt derived from the current message block
    - and a round constant Kt that represents the first 64 bits of the fractional parts of the cube root of first 80 prime numbers

SHA-512 Compression Function (explanation)The SHA-512 Compression Function is the heart of the algorithm. In this Step 4, it processes the message in 1024-bit (128-word) blocks, using a module that consists of 80 rounds, labeled F in Stallings Figure 12, as shown in Figure 12.2. Each round takes as input the 512-bit buffer value, and updates the contents of the buffer. Each round t makes use of a 64-bit value Wt derived using a message schedule from the current 1024-bit block being processed. Each round also makes use of an additive constant Kt, based on the fractional parts of the cube roots of the first eighty prime numbers. The output of the eightieth round is added to the input to the first round to produce the final hash value for this message block, which forms the input to the next iteration of this compression function, as shown on the previous slide

SHA-512 Round Function

These a,b,c,d,e,f,g,h are the registers used to save the data after each round of permutation and combination. As we can see in the figure itself there are many functions like summation, majority, xoring,oring etc.



The structure of each of the 80 rounds is shown in Stallings Figure 12.3. Each 64-bit word shuffled along one place, and in some cases manipulated using a series of simple logical functions (ANDs, NOTs, ORs, XORs, ROTates), in order to provide the avalanche & completeness properties of the hash function. The elements are:

Ch(e,f,g) = (e AND f) XOR (NOT e AND g)

Maj(a,b,c) = (a AND b) XOR (a AND c) XOR (b AND c)

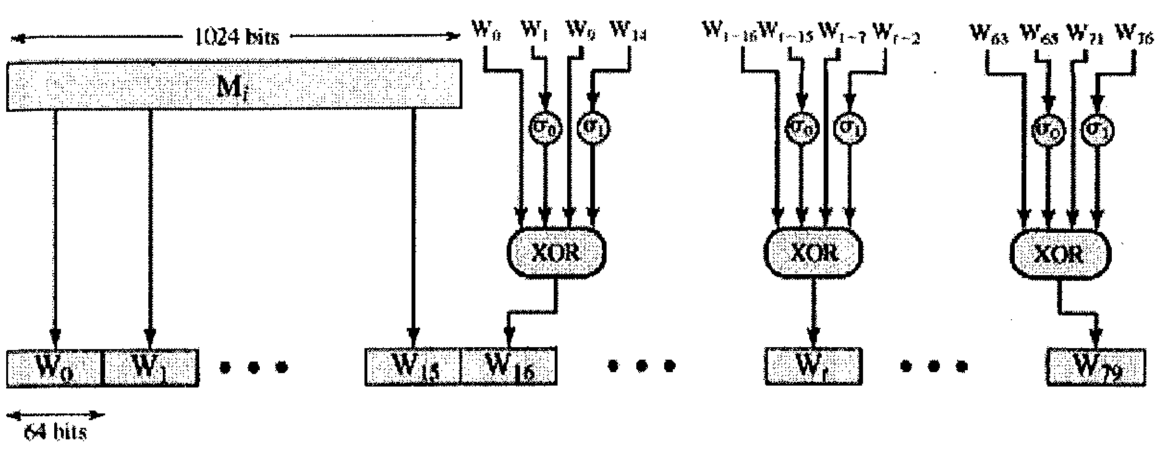
∑(a) = ROTR(a,28) XOR ROTR(a,34) XOR ROTR(a,39)

∑(e) = ROTR(e,14) XOR ROTR(e,18) XOR ROTR(e,41)

+ = addition modulo 2^64

Kt = a 64-bit additive constant

Wt = a 64-bit word derived from the current 512-bit input block.



1024 bits
M_i
W_0 W_1 W_9 W_{14}
W_{t-16} W_{t-15} W_{t-7} W_{t-2}
W_{63} W_{65} W_{71} W_{76}
σ_0
σ_1
XOR
W_0
W_1
W_{15}
W_{16}
W_t
W_{79}
64 bits

the 64-bit word values Wt are derived from the 1024-bit message. The first 16 values of Wt are taken directly from the 16 words of the current block. The remaining values are defined as a function of the earlier values using ROTates, SHIFTs and XORs as shown. The function elements are:

∂0(x) = ROTR(x,1) XOR ROTR(x,8) XOR SHR(x,7)∂1(x) = ROTR(x,19) XOR ROTR(x,61) XOR SHR(x,6)

2.4 Existing algorithms -URL Filtering

Algorithm/ Logic

Several service and device providers like Cisco, Websense, Surfcontrol, Blue Coat, and Gemtek provide network-based URL filtering (NUF) as a solution to classify, monitor, and control web traffic.

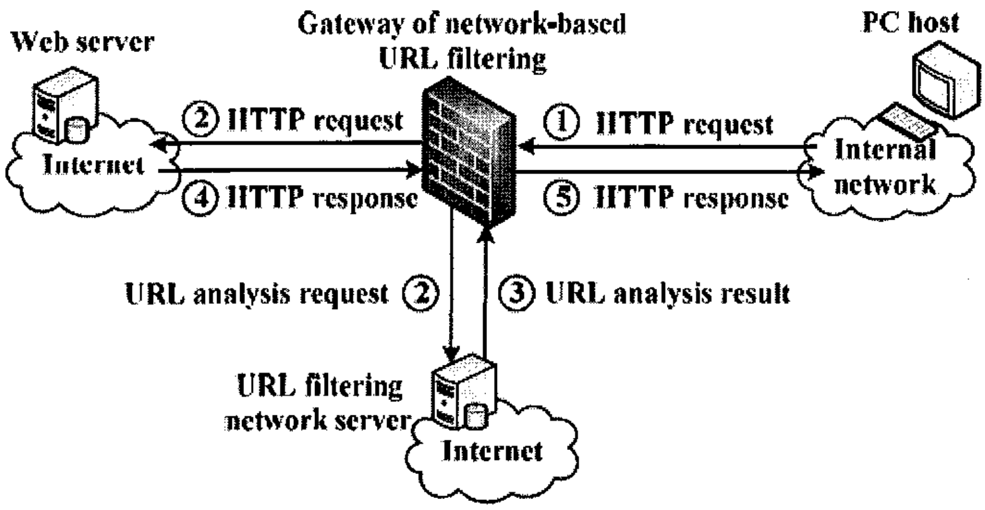


Fig. 1. The general overview of network-based URL filtering.

SPS: a simple filtering algorithm to thwart phishing attacks

Daisuke Miyamoto, Hiroaki Hazeyama, and YoukiKadobayashi
Nara Institute of Science and Technology, 8916-5 Takayama, Ikoma, Nara, Japan
*f*daisu-mi, hiroa-ha, youki-kg@is.naist.jp

Abstract.

In this paper, we explain that by only applying a simple filtering algorithm into various proxy systems, almost all phishing attacks can be blocked without loss of convenience to the user. We propose a system based on a simple filtering algorithm which we call the *Sanitizing Proxy System (SPS)*. The key idea of SPS is that Web phishing attack can be immunized by removing part of the content that traps novice users into entering their personal information. Also, since SPS sanitizes all HTTP responses from suspicious URLs with warning messages, novice users will realize that they are browsing phishing sites. The SPS filtering algorithm is very simple and can be described in roughly 20 steps, and can also be built in any proxy system, such as a server solution, a personal firewall or a browser plug-in. By using SPS with a transparent proxy server, novice users will be protected from almost all Web phishing attacks even if novice users misbehave. With a deployment model,

robustness and evaluation, we discuss the feasibility of SPS in today's network operations.

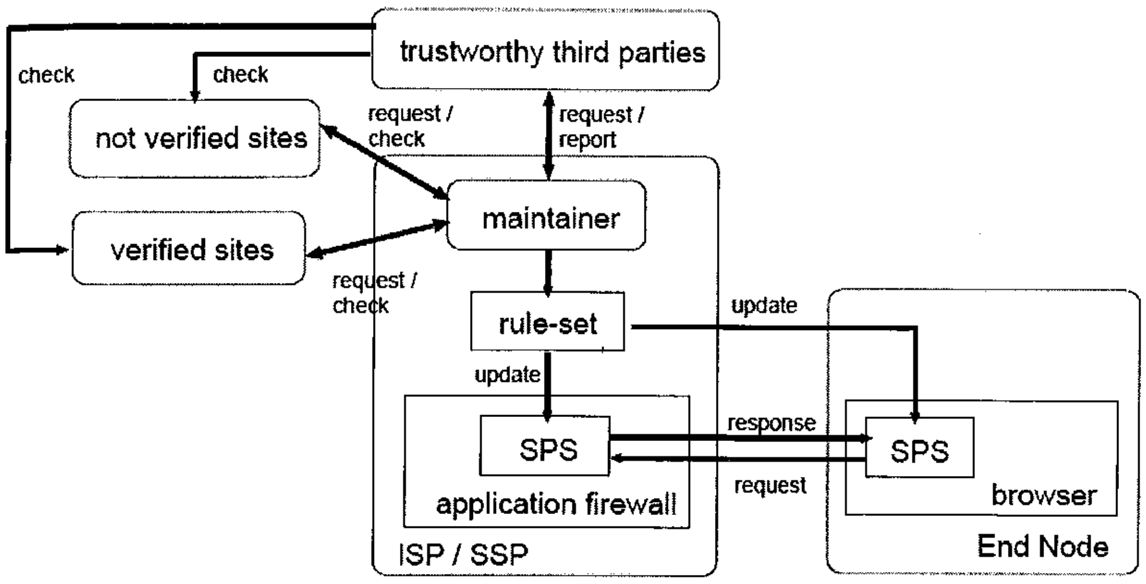


Fig. 4. Deployment Model of SPS

3. DESIGN AND ANALYSIS

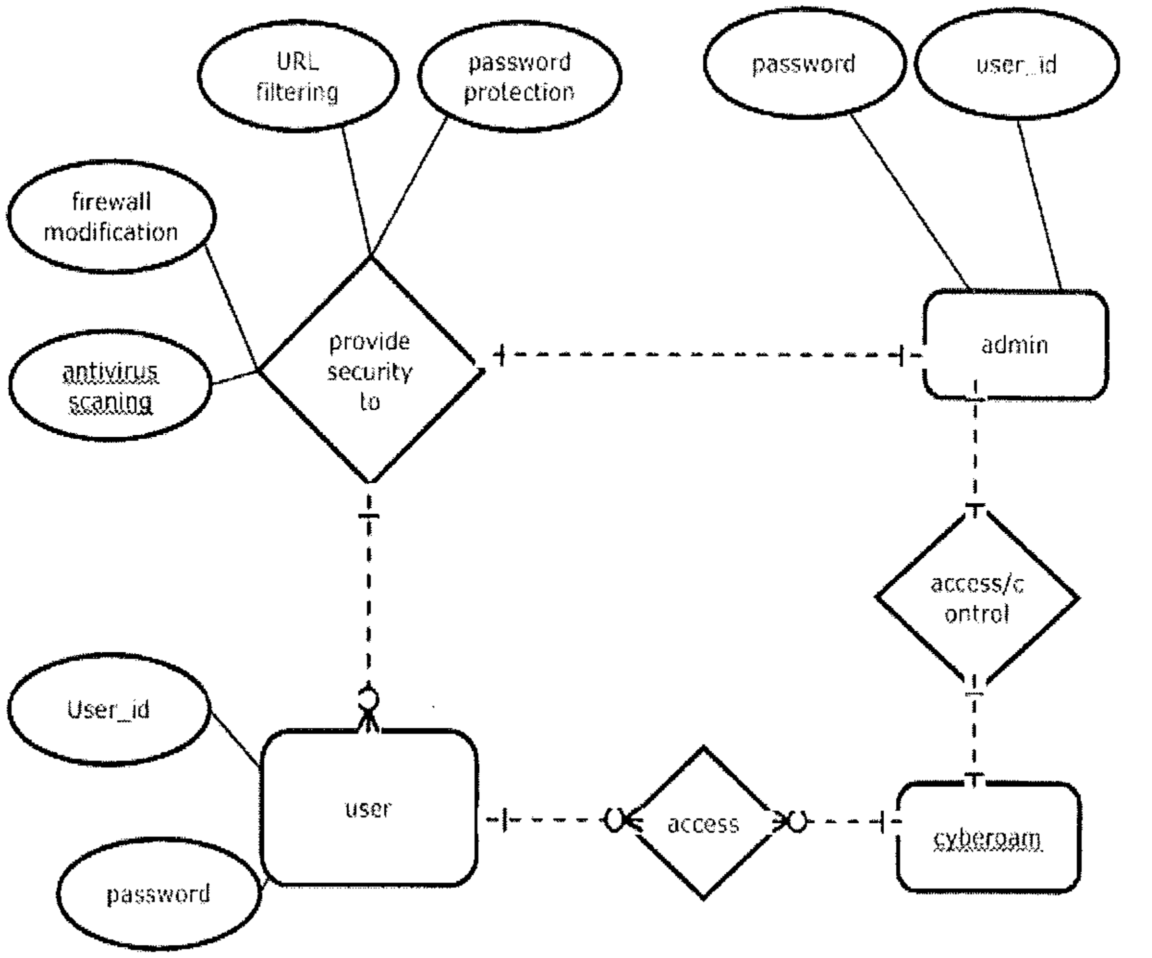
The objective of the system design is to deliever the requirements as specified in the feasibility report.

System design goes through two phases:
- Logical design
- Physical design.

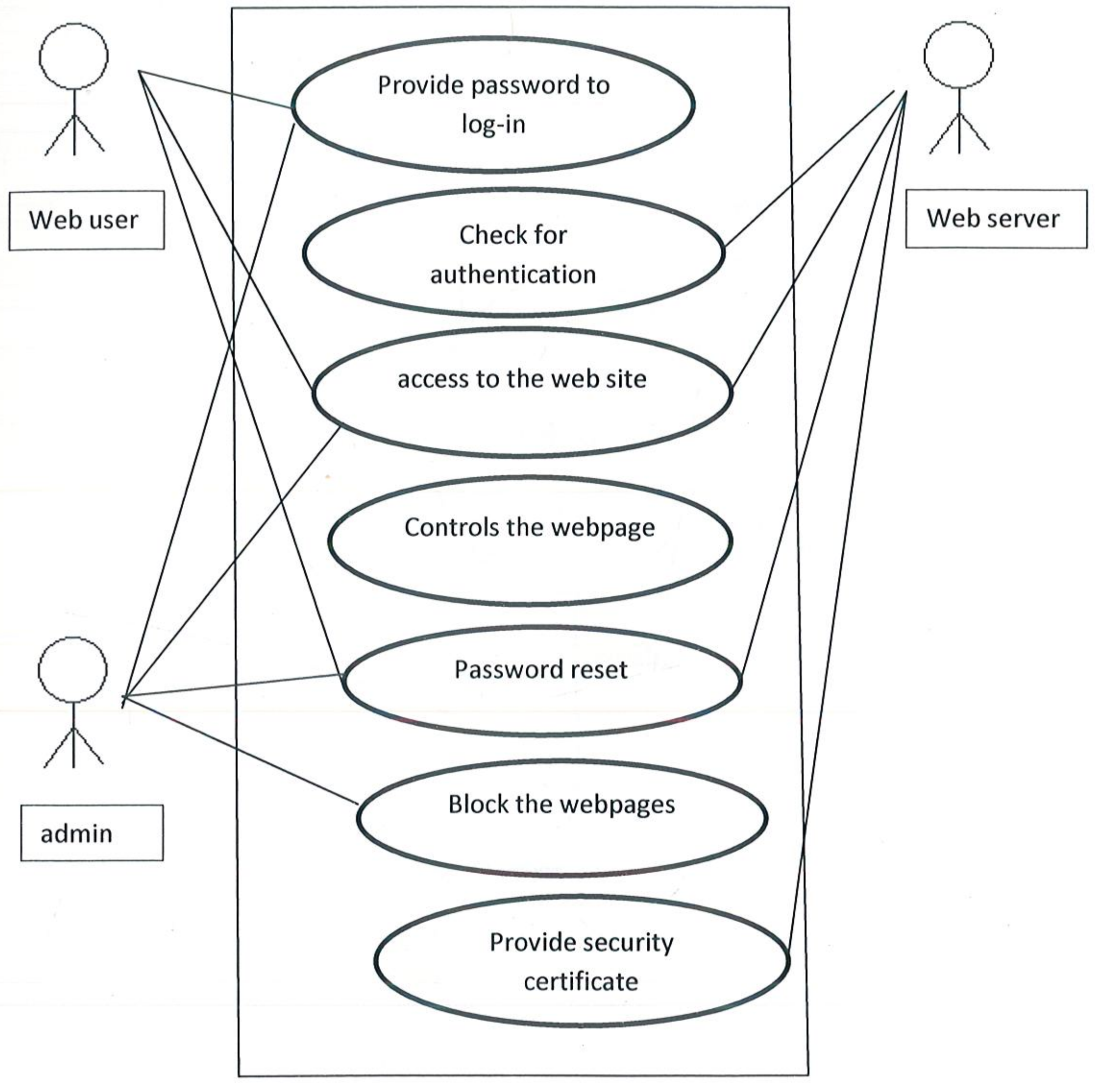
Logical design includes data flow diagram,use case diagram and defines the boundaries of the system.
Physical design defines the structure of the system.

3.1 E-R Diagram

This E-R diagram clearly shows the various attributes of the entities and the relation among them.Here,the Admin having his own unique user id and password can access the web server(cyberoam) and provide security to many users each also having unique user id and password at a time.Apart from that many user can access the server at the same time and server can handle the many request too.

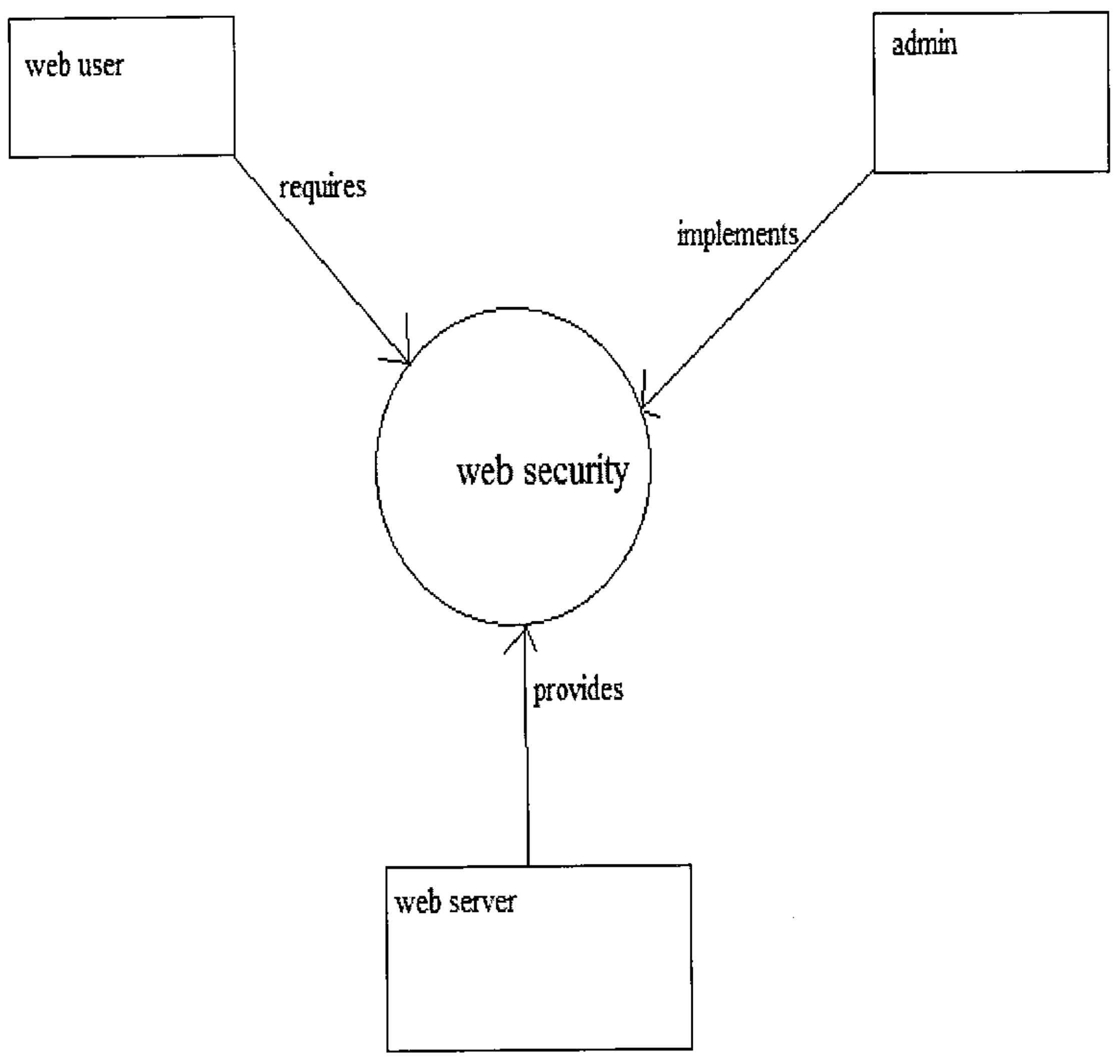


3.2 Use Case diagram



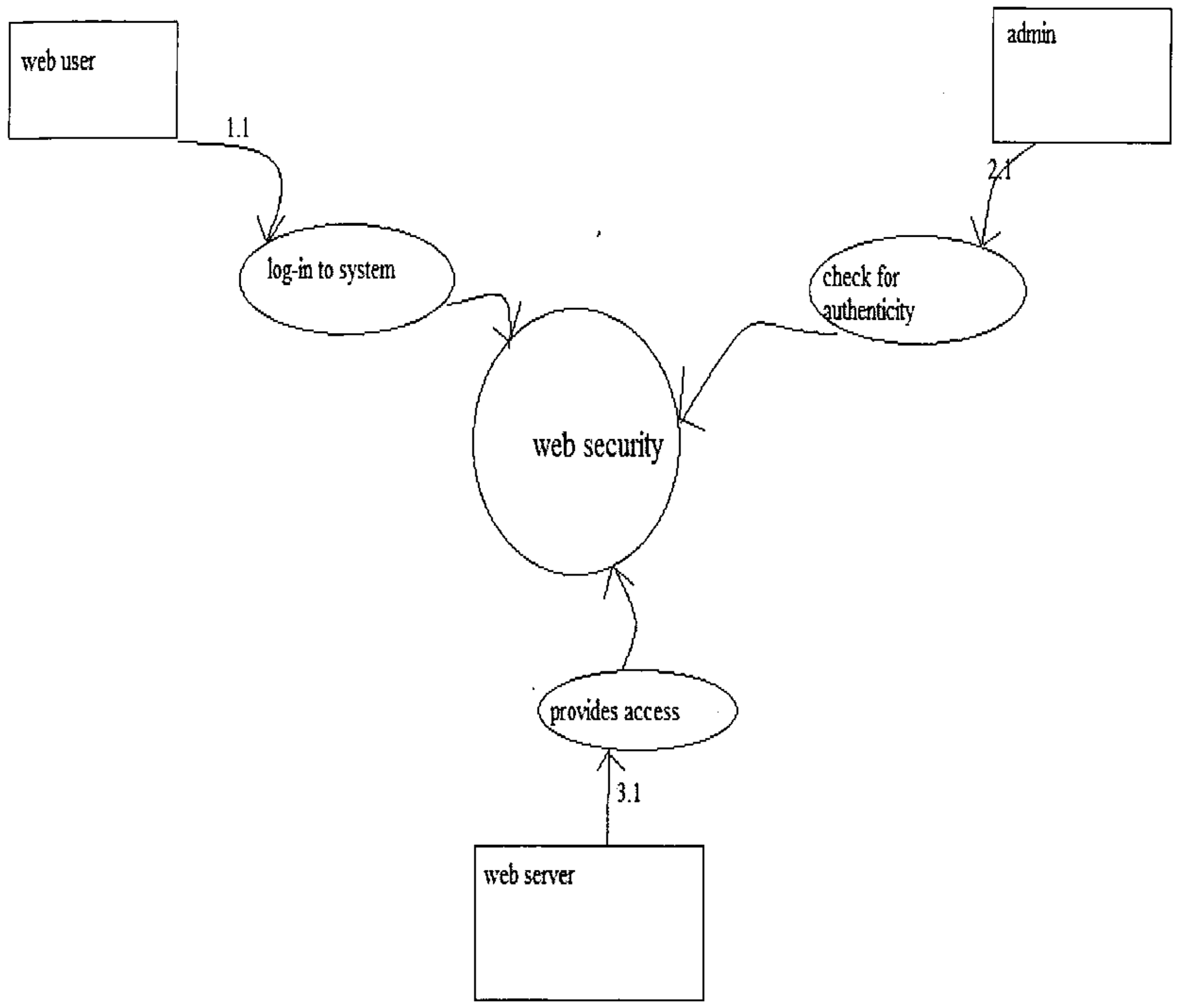
In this Use Case Diagram we have three Actors involved in the process of providing web security. Here, web user provide the login password to access the network or the web page and can also request for the password reset. Admin has control over the server and the network and allow the user to access server and provide other services. similarly web server play its role by allowing to access those pages having security certificate.

3.3 Data Flow Diagram : 0 level



This DFD(0-level) shows flow of data or info and the role each among the user,admin and the web server play in providing web security.

3.4 Data Flow Diagram: 1 level



This DFD (1-level) provides more information than the 0-level and explains the role of each in a more broad way,like a user has to log in to acess the server ,and the role of Admin is to check the authenticity by providing certain unique ids and securing password of each,similarly the web server provides acess when the authenticity is checked.

4. PROGRAMMING AND ALGORITHMS

Password Protection

AIM :

To provide secure login by multiple client to a centralized server.

Algorithm :

At Client End:

1. Initialize hostname (host), port address (port).
2. Establish a connection to the previously initialized host using Socket class object.
3. Retrieve the user name (user) and password (password) from the html document using parameters else initialize them.
4. Define a time stamp (t1) and set it to current time, large random number (q1).
5. Form a message digest using username , password , time stamp random number and send it to the server for authentication.

At Server End:

1. Bind the server that is create a server socket to accept the connection from client.
2. Accept the connection with the client.
3. Receive and read the username (user), time stamp(t1) , random number (q1) , length of message digest (length) , and message digest(p1).
4. Look up for the associated password with the received username in the passwords database.
5. Create a new message digest to verify the authenticity of received message digest using username ,password,time stamp , random number.

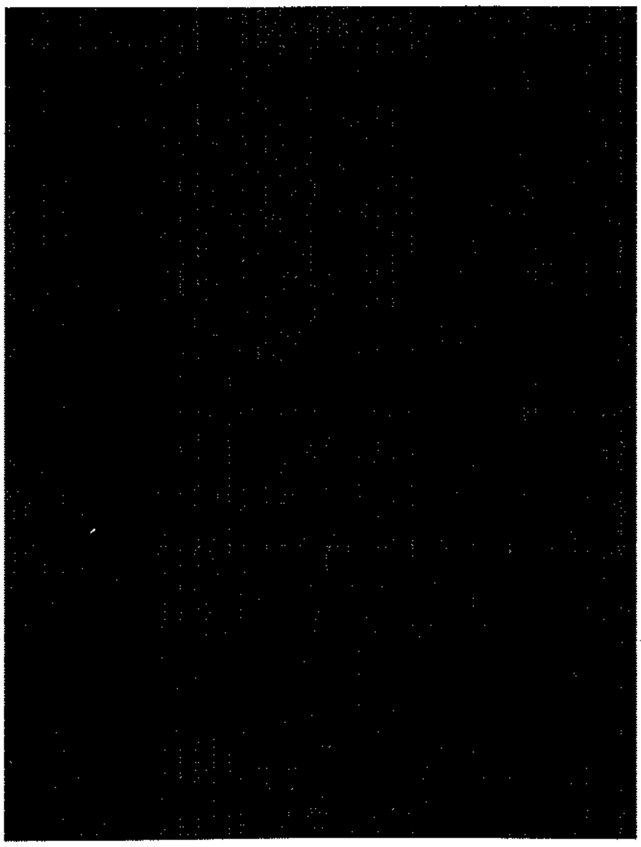
6. Verify whether the received digest is equalto the system generated message digest.
7. Continue with the established connection if the condition in step 6 is satisfied else disconnect the connection.

Creating Message Digest in JAVA:

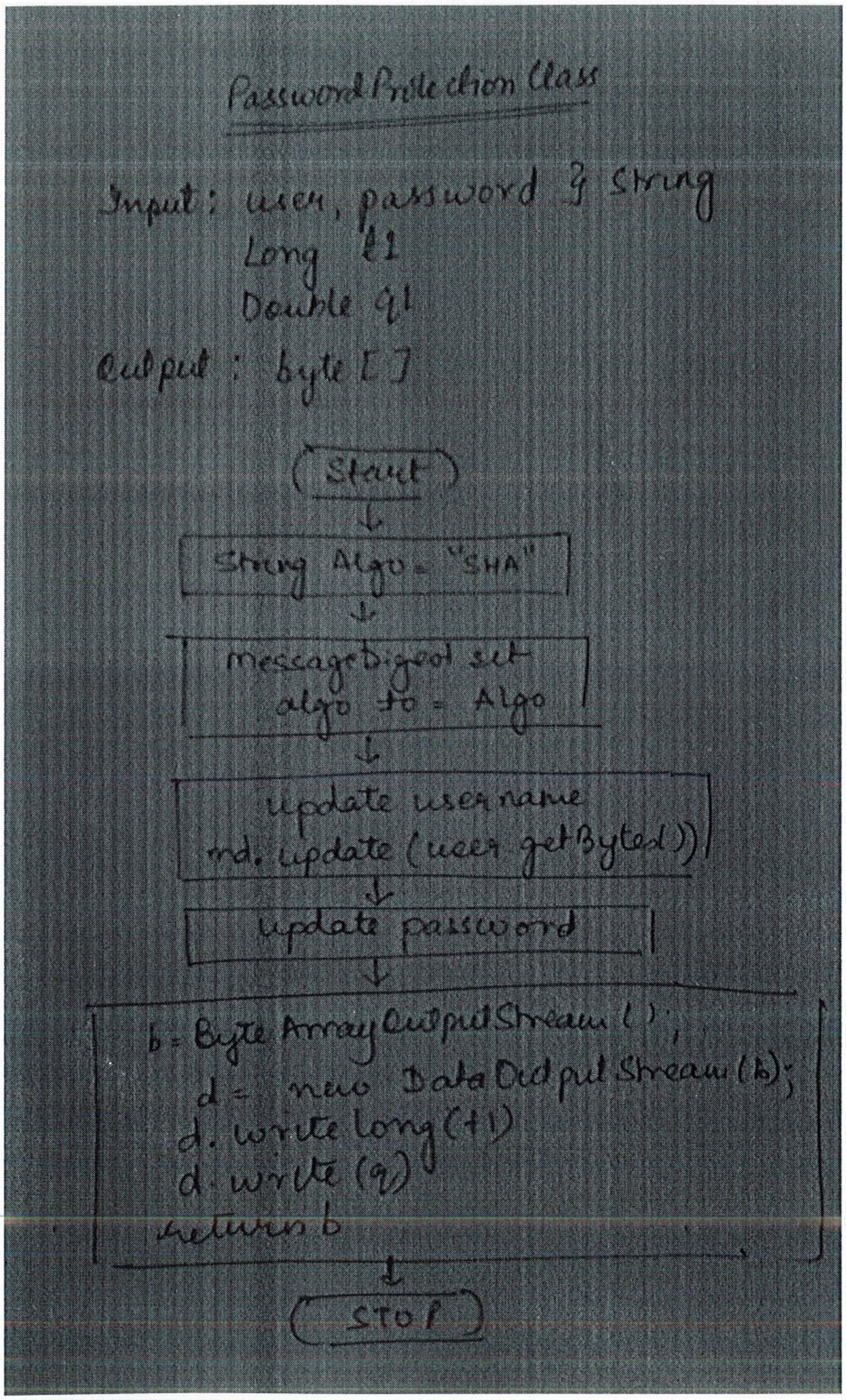
1. Declare and define a message digest class object in java using getInstance method.
2. Feed the data to be digested that is message to the object in binary form using update method.
3. Create the digest and store it in a binary array using digest method through message digest class object.

Flowchart :

Class for Client

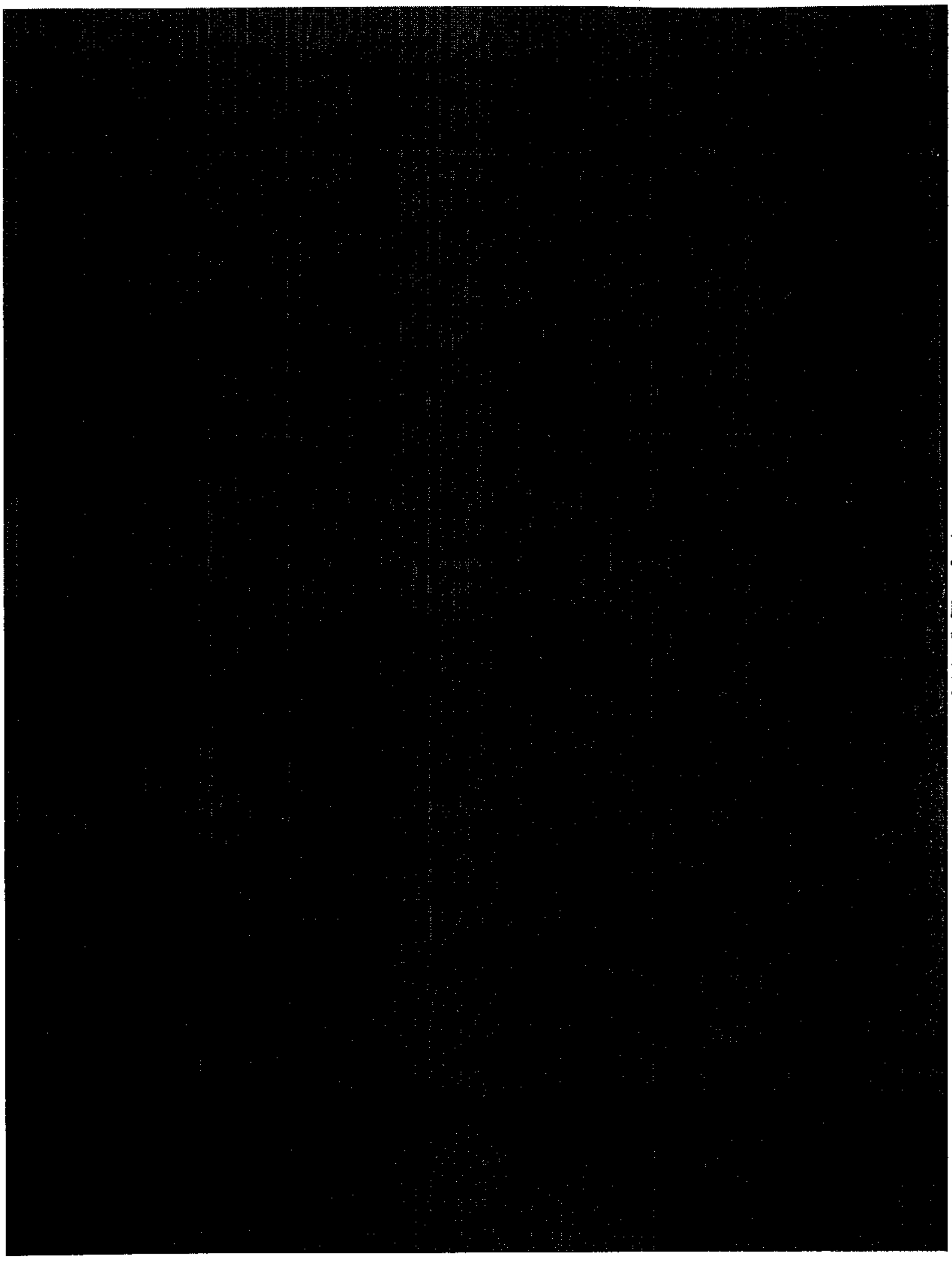


Class for MessageDigest Creation


Password Protection Class
Input : user, password } String
Long t1
Double q1
Output : byte [ ]
Start
String Algo = "SHA"
messageDigest set algo to = Algo
update user name
md. update (user getBytes())
update password
STOP

```
b = ByteArrayOutputStream ( );
  d = new DataOutputStream (b);
d. write long (t1)
d. write (q1)
returns b
```

Class for Server



5. APPLICATIONS

Applications of hash functions are abundant in cryptography and programming practice. We describe several scenarios, which can be used as models for assessing security of real-life applications.

5.1 Digital signatures

Historically, digital signatures were the first application of cryptographically secure hash functions. Hash functions serve a dual role in practical signature schemes: they expand the domain of messages that can be signed by a scheme and they are an essential element of the scheme's security. The first role arises from the fact that it is much easier to design a signature scheme which is secure for signing messages of a fixed length. Consider, for example, the RSA scheme, where the signature of a message $0 \leq M < N$ is $\sigma(M) = MD \bmod N$. Notice that the restriction that $0 \leq M < N$ is necessary, since $\sigma(M) = \sigma(M+N)$. A collision resistant hash function that hashes arbitrary-length strings into numbers between 0 and N lets the signer sign any message M by exponentiation (raising to a secret power) the hash of the message $\sigma(H(M)) = H(M)d \bmod N$.

This is a generic mechanism that transforms any signature scheme good for signing messages of a fixed length and a collision-resistant hash function into a signature scheme that can handle arbitrary-length messages. The mechanism is called the hash-and-sign paradigm and is the basis for all modern practical signature schemes. It is also quite remarkable that virtually all practical signature schemes are either outright insecure or lack proof of security in any reasonable model for signing even fixed-length messages if the message is not hashed. For instance, there is a trivial attack against the RSA signature scheme without a hash: any valid signature C =MD mod N enables the attacker to compute a signature on M2 mod N, which is C2 mod N = (M2)d mod N. The second role of hash functions is to prevent this or similar attacks. It is not well understood which properties of the hash function (akin to one-way or collision-resistance) are sufficient to fulfil this role.

A rare example of a signature scheme that uses a hash function only for domain extension (i.e., it does not require a hash function for signing short messages) is the Cramer-Shoup signature scheme [CS00].

For any hash-and-sign signature scheme, such as FIPS-approved RSA, DSA, and
ECDSA [NIST00], a practical collision-finding attack on the underlying hash function is devastating. Indeed, if the attacker can prepare two documents M1 and M2, whose hashes are identical, then a hash-and-sign signature on M1 is also a valid signature on M2: $\sigma(H(M1)) = \sigma(H(M2))$. It means that if the attacker can obtain a signature on
M1, he automatically gets a signature on M2, which the signer has never examined and certainly did not expect to sign. H The hash-and-sign paradigm can be symbolically represented as σ (H(M)), where H is a hash function and $\sigma(\cdot)$ is the output of a signature scheme. It is imperative that the hash function be collision-resistant if we are to stay within this paradigm. Alternatively, we may combine a signature scheme with a keyed hash function satisfying the following definition:

Definition We say that the keyed function H: {0, 1}× {0, 1}ℓ 7−→ {0, 1}n is target-collision resistant (TCR) if it is hard for the attacker to win the following game:

1. The attacker chooses x ∈ {0, 1}.
2. A random key k is chosen from {0, 1} ℓ.

3. The attacker outputs $y \in \{0, 1\}$. The game is won if and only if $Hk(x) = Hk(y)$.
H Consider the following signature scheme, symbolically represented by $(k,\sigma(k||Hk(M))$.
To sign a message M, the signer generates a random key k and signs the concatenation of $k||Hk(M)$. The signature is then constructed as a pair that consists of k and $\sigma(k||Hk(M))$. It is easy to verify that such a signature scheme is secure as long as the hash function is TCR and $\sigma(\cdot)$ is unforgeable for messages of length n.
H There is compelling theoretical evidence that TCR hash functions might be easier to construct than keyless collision-resistant hash functions in the mold of MD5 or SHA-1, although no standard TCR functions currently exist.

5.2 MAC

Message-authentication code (MAC), introduced in Section 4.2, is a keyed hash function satisfying certain cryptographic properties. Not surprisingly, a MAC can be based on a collision-resistant keyless hash function (although a na¨ıve way of doing so can be easily broken, as described in Section 4.2). In fact, the most commonly used MAC construction, called HMAC, falls into this category. Both TLS and IPSec standards suggest using HMAC based on either MD5 or SHA-1. Assuming that H is a collision resistant hash function, HMAC with secret key $k \in \{0, 1\}^\ell$ is defined as follows: $HMACk(M) = H(k \oplus c1||H(k \oplus c2||M))$, where c1 and c2 are two ℓ-bit constants.

HMAC is provably secure under the assumption that H is collision-resistant and the compression function C(h, x) is pseudo-random if h is unknown. None of the existing attacks on hash functions works if the initial part of the hash function's input is unknown, although such attacks are impossible to rule out in the future. It is important to note that MAC can be based on other primitives than collision resistant hashes. One recently standardized construction, CMAC, proposes to use a block cipher, such as AES, in the CBC mode [NIST05]. Another construction, which can be as fast as 1.5 cycles/byte (cf. Table 3), is called UMAC and currently exists in the form of an Internet draft; its RFC is forthcoming [B+05].

IMPLEMENTATION

Interface designing of login page

//this is the program for creating the interface of the login form,written in Java using servlets and //the swing package

```
importjava.awt.*;

        importjava.awt.event.*;

        importjavax.swing.*;

        public class loginpage extends JFrame

        {

                privateJLabel tag1;

                privateJLabel tag2;

                privateJTextFielduserarea;

                privateJPasswordFieldPasswordarea;

                privateJButtonloginbutton;

                privateJPanelcontentPane;


publicloginpage()

                {

                        create();

                        this.setVisible(true);

                }


        private void create()
```

```
                {
                        tag1 = new JLabel();
                        tag2 = new JLabel();
                        userarea = new JTextField();
                        Passwordarea = new JPasswordField();
                        loginbutton = new JButton();
                        contentPane = (JPanel)this.getContentPane();
tag1.setHorizontalAlignment(SwingConstants.RIGHT);
        tag1.setText("user_id:");
                        tag2.setHorizontalAlignment(SwingConstants.RIGHT);
                        tag2.setText("user_password:");

userarea.addActionListener(new ActionListener(){

                                public void actionPerformed(ActionEvent e)
                                {
                                        userarea_response(e);
                                }

                          });

                        Passwordarea.addActionListener(new ActionListener() {
                                public void actionPerformed(ActionEvent e)
                                {
                                        Passwordarea_response(e);
                                }
                            });
```

```
          loginbutton.setText("LOGIN");
          loginbutton.addActionListener(new ActionListener() {
                public void actionPerformed(ActionEvent e)
          {
                      loginbutton_response(e);
                }
           });
          contentPane.setLayout(null);
          contentPane.setBorder(BorderFactory.createEtchedBorder());
          contentPane.setBackground(new Color(150, 150,150));
          additems(contentPane, tag1, 5,20,106,20);
          additems(contentPane, tag2, 5,57,106,20);
          additems(contentPane, userarea, 120,10,200,25);
          additems(contentPane, Passwordarea, 120,45,200,25);
          additems(contentPane, loginbutton, 130,80,80,30);
    this.setTitle("Login for Registered Users");
          this.setLocation(new Point(76, 182));
          this.setSize(new Dimension(335, 141));
          this.setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
          this.setResizable(false);
    }

    private void additems(Container container,Component c1,int a,intb,intwidth,int
height)
    {
          c1.setBounds(a,b,width,height);
```

```
                    container.add(c1);

                }


private void userarea_response(ActionEvent e)

                {

                }


private void Passwordarea_response(ActionEvent e)

                  {

                }


                private void loginbutton_response(ActionEvent e)

                {

                    System.out.println("\nloginbutton_response(ActionEvent e) called.");

                    String username = new String(userarea.getText());

                    String password = new String(Passwordarea.getText());


                    if(username.equals("") || password.equals(""))

                            {

                                loginbutton.setEnabled(false);

                            JLabelerrorFields = new JLabel("please enter the
username/password to login.");

                            JOptionPane.showMessageDialog(null,errorFields);

                            userarea.setText("");

                        Passwordarea.setText("");

                        loginbutton.setEnabled(true);

                            this.setVisible(true);
                        }
                        else
                        {

                }
                 }


public static void main(String[] args)
            {
                    JFrame.setDefaultLookAndFeelDecorated(true);
                    JDialog.setDefaultLookAndFeelDecorated(true);
                    newloginpage();
            };
}
```

Codes:

MESSAGE DIGEST

```
/*
defines two static methods. The makeDigest() method creates a message digest from its
input data. It uses a helper method, makeBytes(), whose purpose is to convert a long and a double
into an array of bytes.
*/


// importing the necessary packages into the java project

import java.io.*;
import java.security.*;
```

```
// main class function

public class PasswordProtection
{

 public static byte[] makeDigest(string user,Stringpassword,long t1,double q1)
   throws NoSuchAlgorithmException
   {
MessageDigest md= MessageDigest.getInstance("SHA");
md.update(user.getBytes());
md.update(password.getBytes());
md.update(makeBytes(t1,q1));
    return md.digest();
   }

 public static byte[] makeBytes(long t,double q)
  throws IOException
   {
ByteArrayOutputStreambyteOut = new ByteArrayOutputStream();
DataOutputStreamdataOut = new DataOutputStream(byteOut);
dataOut.writeLong(t);
dataOut.writeDouble(q);
    return byteOut.toByteArray();
   }
}
```

CLIENT END CONNECTION

```
import java.io.*;
import java.net.*;
import java.security.*;
import java.util.Date;

import PasswordProtection

public class PClient
{
 public void sendAuthentication(String user,String password,
OutputStreamoutstream)
    throws IOException,NoSuchAlgorithmException
    {
DataOutputStream out = new DataOutputStream(outStream);
      long t1= (new Date()).getTime();
      double q1 = Math.random();
      byte[] p1 = PasswordProtection.makeDigest(user,password,t1,q1);
out.writeUTF(user);

out.writeLong(t1);
out.writeDouble(q1);
out.writeInt(p1.length);
out.write(p1);
out.flush();
    }


 public static void main()
    {
     String host = new String("localHost");
int port = 8080;
     String user = "surbhi";
```

```
          String password = "project";
          Socket s = new Socket(host,port);
pClient client = new Pclient();
client.sendAuthentication(user,password,s.getOutputStream());
s.close();
          }
}
```

SERVER END CONNECTION

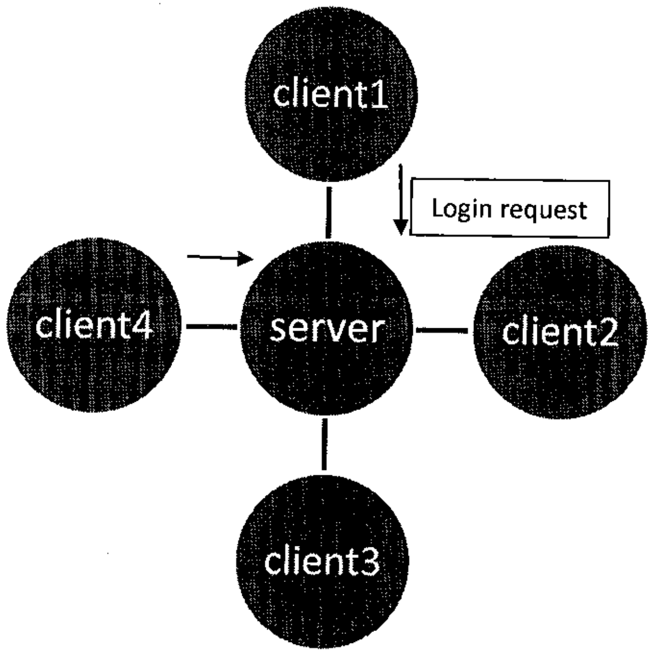
```
//this program implements the server connection of password verification



import java.io.*;
import java.net.*;
import java.security.*;

//import java.util.Date;

import PasswordProtection;

public class PServer {
 public boolean authenticate(InputStreaminStream)
   throws IOException, NoSuchAlgorithmException
    {
DataInputStream in = new DataInputStream(inStream);
    String user= in.readUTF();
    long t1 = in.readLong();
    double q1 = in.readDouble();
int length = in.readInt();
    byte[] p1= new byte[length];
in.readFULLY(p1);
    String password = new String("project");
    byte[] check = PasswordProtection.makeDigest(user,password,t1,q1);
```

```
        return MessageDigest.isEqual(p1,check);
       }

     public static void main() throws Exception
       {
    int port=8080;
    ServerSocket s = new ServerSocket(port);
        Socket client=s.accept();
        if(s.authenticate(client.getInputStream()))
    System.out.println("succesful password");
        else
    System.out.println("login failed");
    s.close();
       }
}
```

The above code implementing client and server end connection mainly provides a basic connection between a single client and server, without any database connectivity.

Therefore below is the revised code that provides network connection between multiple clients and server and uses a database at the server side to retrieve password corresponding to the client that made the login request.

The environment simulated is depicted by the figure below:



CLIENT END CONNECTION – Client1

```
import java.io.*;

import java.net.*;

import java.security.*;

import java.util.Date;


publicclass client1 {

        publicvoid sendAuthentication(String user,String password,

                OutputStream outstream)

            throws IOException,NoSuchAlgorithmException

                {

                    DataOutputStream out = new DataOutputStream(outstream);

            long t1= (new Date()).getTime();

            double q1 = Math.random();
        byte[] p1 = PasswordProtection.makeDigest(user,password,t1,q1);
            out.writeUTF(user);


            out.writeLong(t1);
            out.writeDouble(q1);
            out.writeInt(p1.length);
            out.write(p1);
            out.flush();


        }
    publicvoid sendMessage(OutputStream outstream) throws IOException{
        PrintWriter pw= new PrintWriter(outstream,true);
        BufferedReader buf = new BufferedReader(new
InputStreamReader(System.in));
        while(true)
        {
            String msg= buf.readLine();
            pw.println(msg);
        }


    }




    publicstaticvoid main(String[] args) throws UnknownHostException, IOException,
NoSuchAlgorithmException {


        String host = new String("localHost");
```

```
int port = 8080;
        String user = "surbhi";
        String password = "project";
        Socket s = new Socket(host,port);
        client1 client = new client1();
        client.sendAuthentication(user,password,s.getOutputStream());
        client.sendMessage(s.getOutputStream());
        s.close();
        }
}
```

CLIENT END CONNECTION – Client2

```
import java.io.BufferedReader;

import java.io.DataOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStream;

import java.io.PrintWriter;

import java.net.Socket;

import java.net.UnknownHostException;

import java.security.NoSuchAlgorithmException;

import java.util.Date;



publicclass client2 {



        publicvoid sendAuthentication(String user,String password,

                OutputStream outstream)

            throws IOException,NoSuchAlgorithmException

                {

                    DataOutputStream out = new DataOutputStream(outstream);

            long t1= (new Date()).getTime();

            double q1 = Math.random();

            byte[] p1 = PasswordProtection.makeDigest(user,password,t1,q1);

                out.writeUTF(user);


                out.writeLong(t1);
```

```
                    out.writeDouble(q1);
                    out.writeInt(p1.length);
                    out.write(p1);
                    out.flush();
                }
        publicvoid sendMessage(OutputStream outstream) throws IOException{
                PrintWriter pw= new PrintWriter(outstream,true);
                BufferedReader buf = new BufferedReader(new
InputStreamReader(System.in));
                while(true)
                {
                    String msg= buf.readLine();
                    pw.println(msg);
                }

        }




        publicstaticvoid main(String[] args)throws UnknownHostException, IOException,
NoSuchAlgorithmException  {


                String host = new String("localHost");
int port = 8080;
    String user = "nikita";
    String password = "gupta";
    Socket s = new Socket(host,port);
    client2 client = new client2();
```

```
        client.sendAuthentication(user,password,s.getOutputStream());
        client.sendMessage(s.getOutputStream());
        s.close();
          }

    }
```

CLIENT END CONNECTION – Client3

```
import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStream;
import java.io.PrintWriter;
import java.net.Socket;
import java.net.UnknownHostException;
import java.security.NoSuchAlgorithmException;
import java.util.Date;


publicclass client3 {


        publicvoid sendAuthentication(String user,String password,
                OutputStream outstream)
            throws IOException,NoSuchAlgorithmException
                {
                    DataOutputStream out = new DataOutputStream(outstream);
            long t1= (new Date()).getTime();
            double q1 = Math.random();
            byte[] p1 = PasswordProtection.makeDigest(user,password,t1,q1);
                out.writeUTF(user);


                out.writeLong(t1);
```

```
                    out.writeDouble(q1);

                    out.writeInt(p1.length);

                    out.write(p1);

                    out.flush();

                  }


        publicvoid sendMessage(OutputStream outstream) throws IOException{

                PrintWriter pw= new PrintWriter(outstream,true);

                BufferedReader buf = new BufferedReader(new
InputStreamReader(System.in));

                while(true)

                {

                        String msg= buf.readLine();

                        pw.println(msg);

                }


        }



        publicstaticvoid main (String[] args) throws UnknownHostException, IOException,
NoSuchAlgorithmException {


                String host = new String("localHost");

int port = 8080;

    String user = "sweta";

    String password = "projec1t";

    Socket s = new Socket(host,port);

    client3 client = new client3();
```

```
        client.sendAuthentication(user,password,s.getOutputStream());
        client.sendMessage(s.getOutputStream());
        s.close();
          }
    }
```

SERVER END CONNECTION

```
import java.io.*;

import java.net.*;

publicclass mainServer {


        publicstaticintport = 8080;

        publicstaticvoid main(String[] args) throws IOException {

                ServerSocket server = new ServerSocket(port);

                System.out.println("server has been set up and is ready for connection");

                while(true)

                {

                  Socket client = server.accept();

                new serverEcho(client).start();

                }

        }


}
```

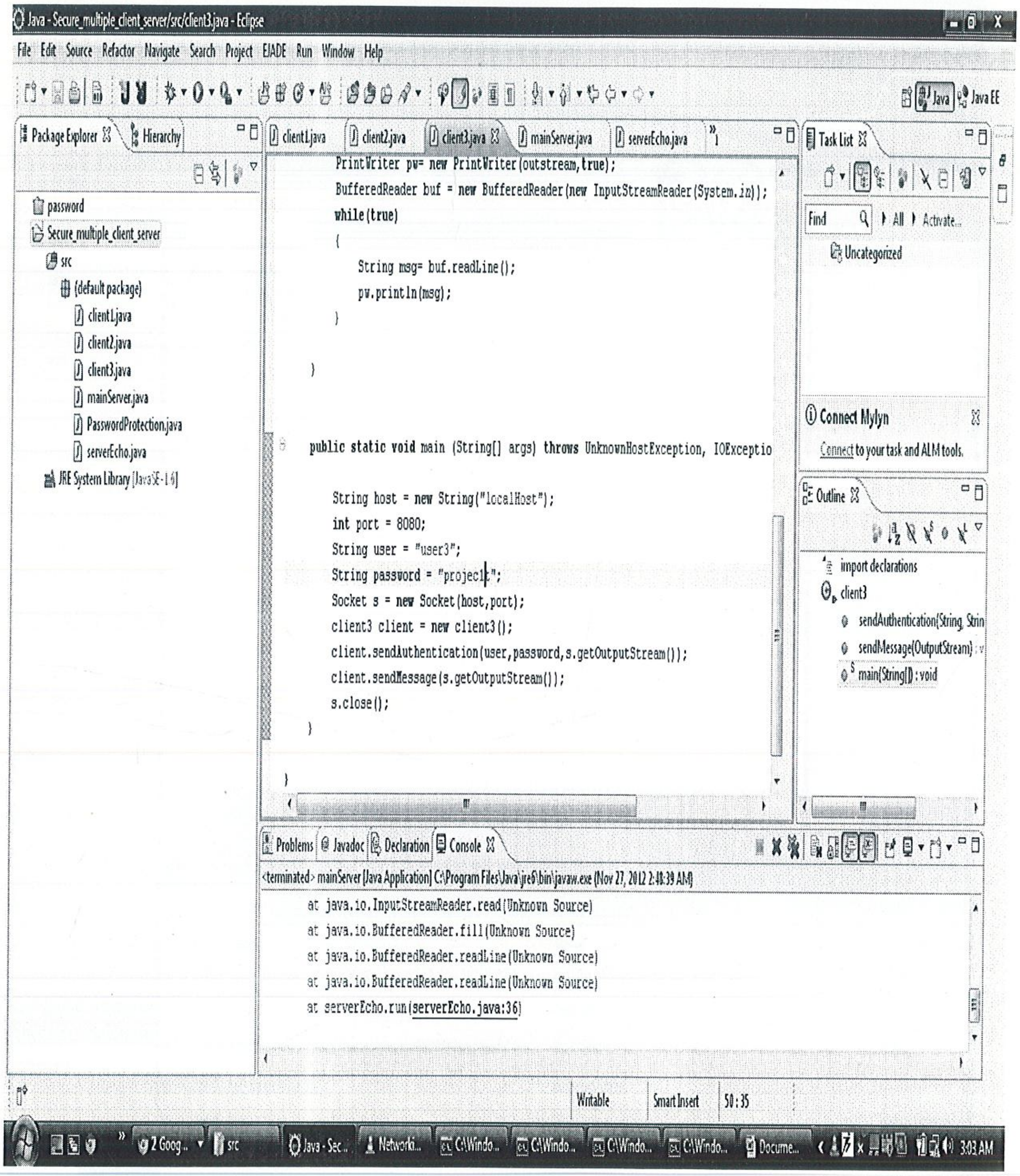
SERVER THREAD CONNECTION

```
        import java.net.*;
        import java.security.MessageDigest;
        import java.security.NoSuchAlgorithmException;
        import java.io.*;
        import java.sql.*;

        public class serverEcho extends Thread {

                Socket client;
```

```
String user;
serverEcho(Socket client)
{
 this.client=client;
}
public String getPassword(String user) throws SQLException, Exception{
        Class.forName("com.mysql.jdbc.Driver");
        String password = new String("");
        int flag=0;
        Connection con =null;
        Statement stmt=null;
        ResultSet rs= null;
        con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/users","root","root")
;
        stmt=con.createStatement();
        rs=stmt.executeQuery("Select * from user_info");
        while(rs.next()&&flag==0){
                //System.out.println(rs.getString("username"));
                if(user.equals(rs.getString("username"))){

                        password = rs.getString("password");
                        flag=1;


                }
        }

        if(flag==1)
                return password;
        else
        {
                System.out.println("USERNAME DOES NOT EXIST");
return password;
```

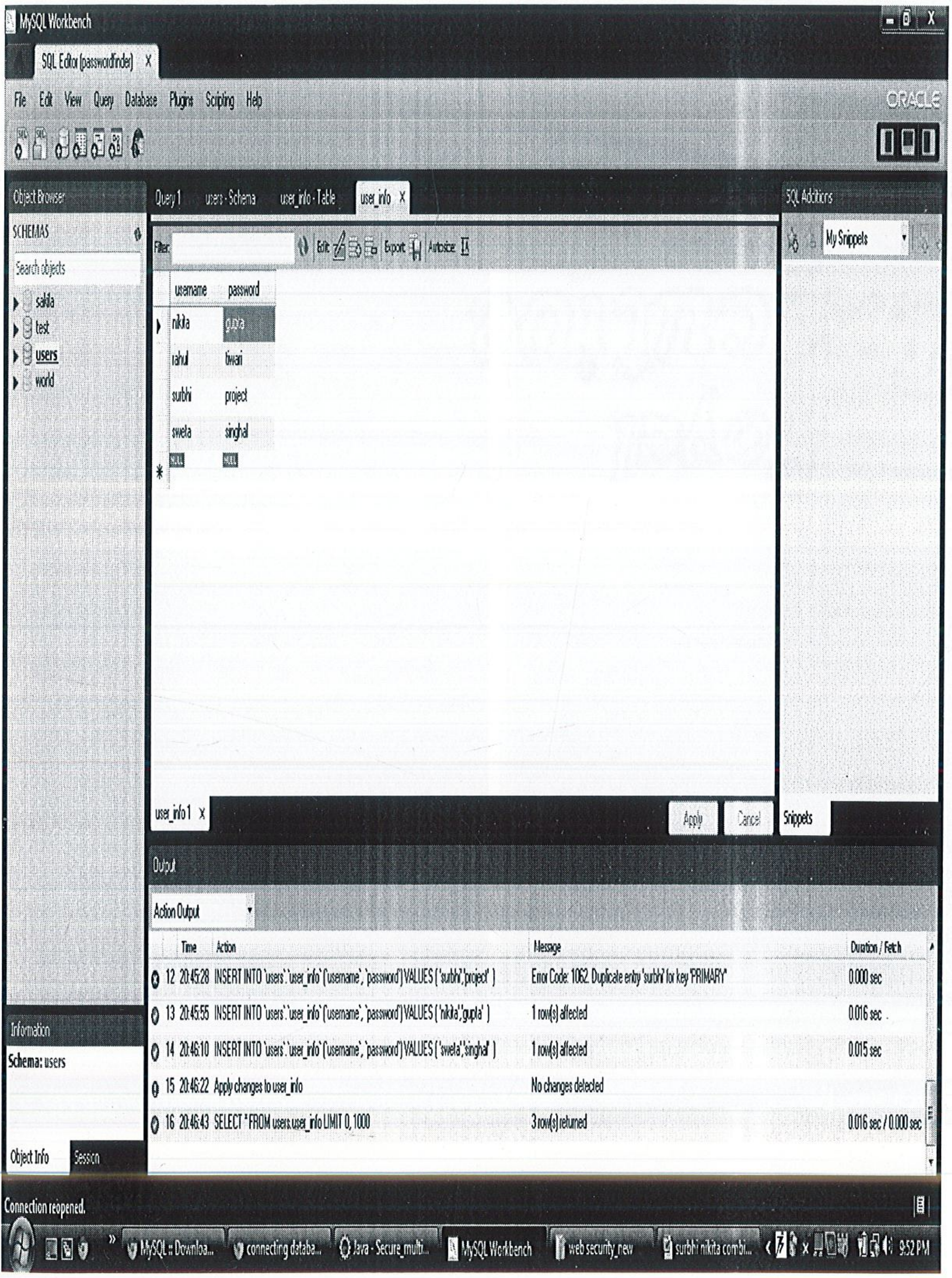
```
        }


    }
    public boolean authenticate(InputStream inStream)
      throws Exception
      {
        DataInputStream in = new DataInputStream(inStream);
        user= in.readUTF();
        long t1 = in.readLong();
        double q1 = in.readDouble();
        int length = in.readInt();
        byte[] p1= new byte[length];
        in.readFully(p1);
        String password = getPassword(user);
        byte[] check = PasswordProtection.makeDigest(user,password,t1,q1);
        return MessageDigest.isEqual(p1,check);
      }
    public void run(){
            try {
                    if(authenticate(client.getInputStream()))
                    {
                        System.out.println(user +":   Login succesful \n");
      String msg= null;
      BufferedReader buff = new BufferedReader
        (new InputStreamReader(client.getInputStream()));
      while((msg=buff.readLine())!=null){
    System.out.println("message from : "+ user + "   "+msg+"\n");
      }
      client.close();
                    }
                     else{
                            System.out.println(user+":   login failed \n");
                            client.close();
                        }

                    } catch (NoSuchAlgorithmException e) {

                            e.printStackTrace();
                    } catch (IOException e) {

                            e.printStackTrace();
                    } catch (SQLException e) {

                            e.printStackTrace();
                    } catch (Exception e) {

                            e.printStackTrace();
                    }
            }

    }
```

7.Results :



```java
        PrintWriter pw= new PrintWriter(outstream,true);
        BufferedReader buf = new BufferedReader(new InputStreamReader(System.in));
        while(true)
        {
            String msg= buf.readLine();
            pw.println(msg);
        }


    }



    public static void main (String[] args) throws UnknownHostException, IOExceptio

        String host = new String("localHost");
        int port = 8080;
        String user = "user3";
        String password = "project";
        Socket s = new Socket(host,port);
        client3 client = new client3();
        client.sendAuthentication(user,password,s.getOutputStream());
        client.sendMessage(s.getOutputStream());
        s.close();
    }


}
```

```
<terminated> mainServer [Java Application] C:\Program Files\Java\jre6\bin\javaw.exe (Nov 27, 2012 2:48:39 AM)
        at java.io.InputStreamReader.read(Unknown Source)
        at java.io.BufferedReader.fill(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at java.io.BufferedReader.readLine(Unknown Source)
        at serverEcho.run(serverEcho.java:36)
```

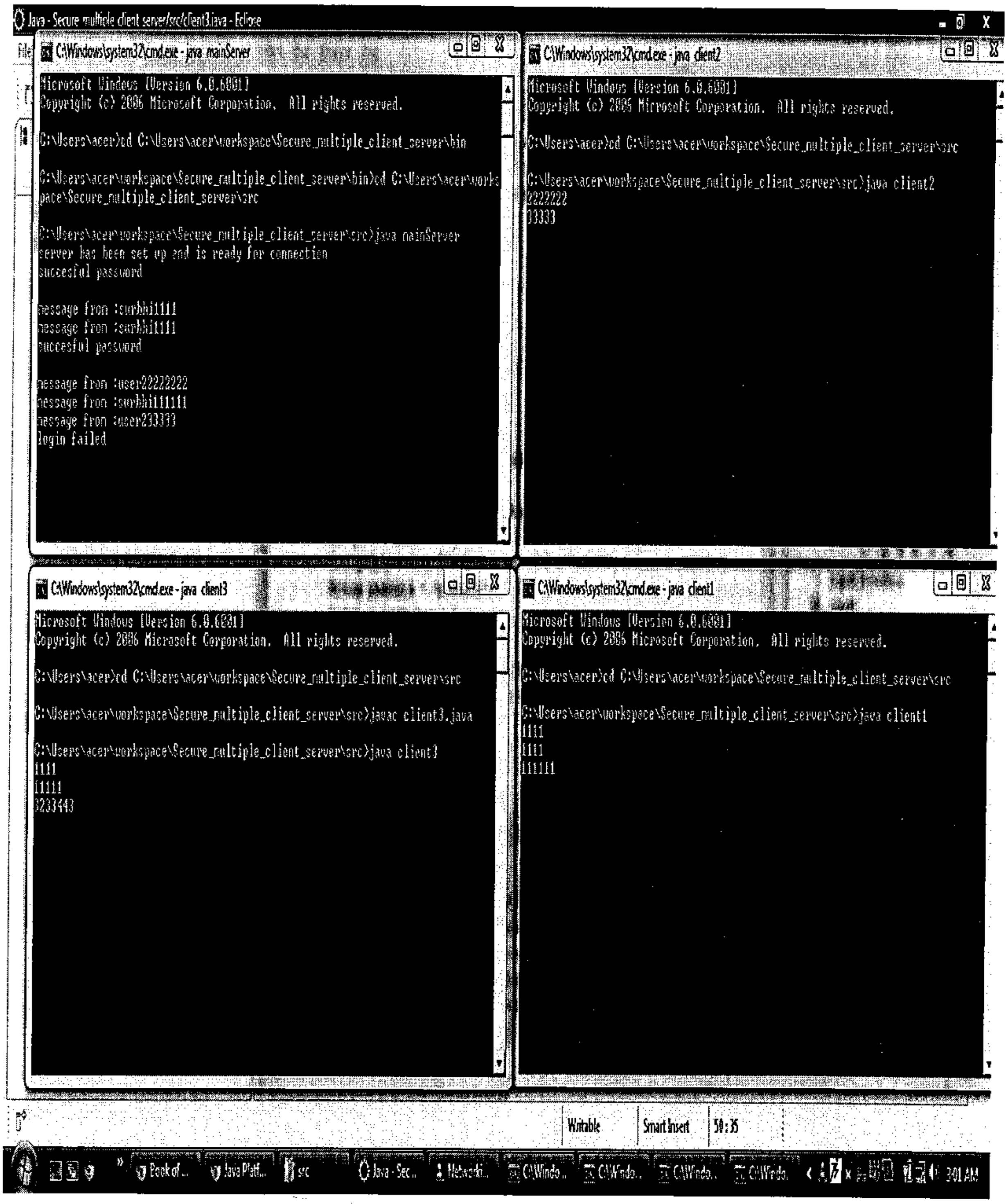


username	password
nikita	gupta
rahul	tiwari
surbhi	project
sweta	singhal
NULL	NULL

Output

Action Output

```
   Time      Action                                                                   Message                                               Duration / Fetch
12 20:45:28  INSERT INTO `users`.`user_info` (`username`, `password`) VALUES ('surbhi','project' )  Error Code: 1062. Duplicate entry 'surbhi' for key 'PRIMARY'  0.000 sec
13 20:45:55  INSERT INTO `users`.`user_info` (`username`, `password`) VALUES ('nikita','gupta' )    1 row(s) affected                                     0.016 sec
14 20:46:10  INSERT INTO `users`.`user_info` (`username`, `password`) VALUES ('sweta','singhal' )   1 row(s) affected                                     0.015 sec
15 20:46:22  Apply changes to user_info                                                      No changes detected
16 20:46:43  SELECT * FROM users.user_info LIMIT 0, 1000                                     3 row(s) returned                                     0.016 sec / 0.000 sec
```

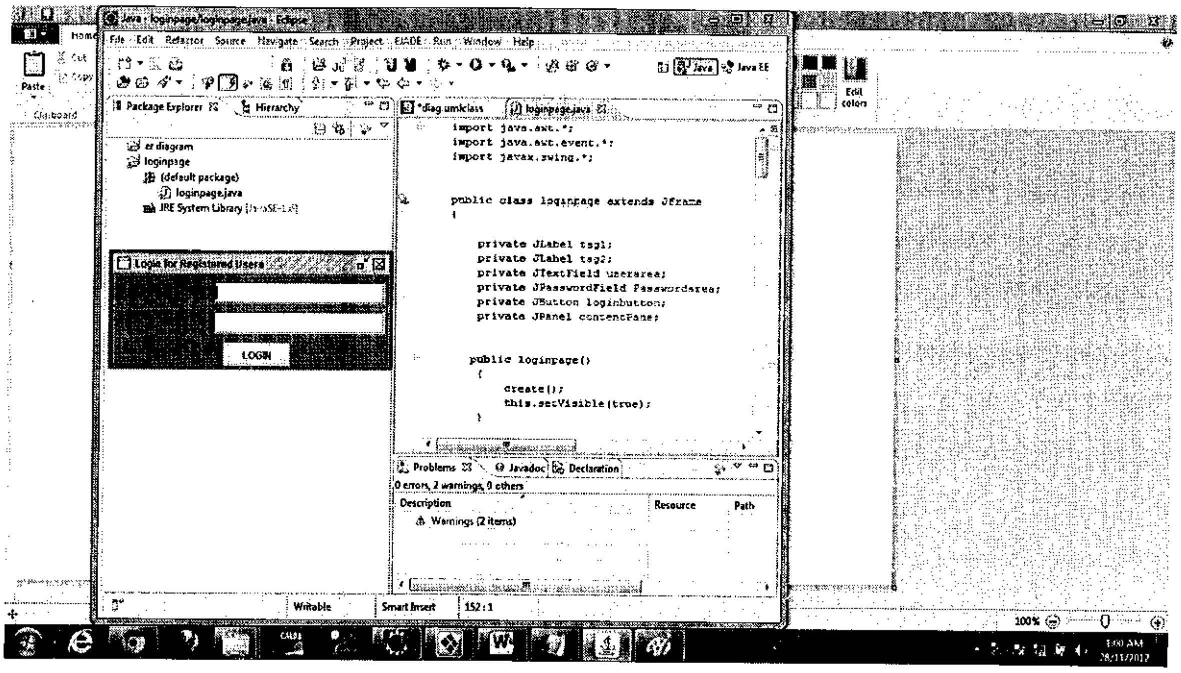

Java - Secure multiple client server/src/client3.java - Eclipse
C:\Windows\system32\cmd.exe - java mainServer
C:\Windows\system32\cmd.exe - java client2
C:\Windows\system32\cmd.exe - java client3
C:\Windows\system32\cmd.exe - java client1
Writable
Smart Insert
58 : 35

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\acer>cd C:\Users\acer\workspace\Secure_multiple_client_server\bin

C:\Users\acer\workspace\Secure_multiple_client_server\bin>cd C:\Users\acer\works
pace\Secure_multiple_client_server\src

C:\Users\acer\workspace\Secure_multiple_client_server\src>java mainServer
server has been set up and is ready for connection
succesful password

message from :surbhi1111
message from :surbhi1111
succesful password

message from :user22222222
message from :surbhi111111
message from :user233333
login failed
```

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\acer>cd C:\Users\acer\workspace\Secure_multiple_client_server\src

C:\Users\acer\workspace\Secure_multiple_client_server\src>java client2
2222222
33333
```

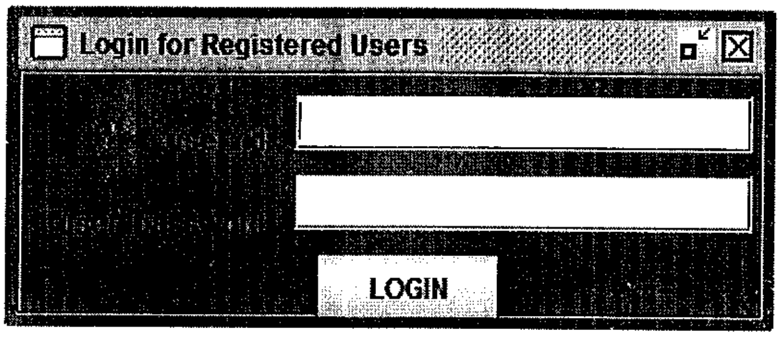
```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\acer>cd C:\Users\acer\workspace\Secure_multiple_client_server\src

C:\Users\acer\workspace\Secure_multiple_client_server\src>javac client3.java

C:\Users\acer\workspace\Secure_multiple_client_server\src>java client3
1111
11111
3233443
```

```
Microsoft Windows [Version 6.0.6001]
Copyright (c) 2006 Microsoft Corporation.  All rights reserved.

C:\Users\acer>cd C:\Users\acer\workspace\Secure_multiple_client_server\src

C:\Users\acer\workspace\Secure_multiple_client_server\src>java client1
1111
1111
111111
```


```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class loginpage extends JFrame
{

    private JLabel tag1;
    private JLabel tag2;
    private JTextField userarea;
    private JPasswordField Passwordarea;
    private JButton loginbutton;
    private JPanel contentPane;


   public loginpage()
    {
        create();
        this.setVisible(true);
    }
```

0 errors, 2 warnings, 0 others

Description	Resource	Path
Warnings (2 items)		



8.Conclusion and future scope

Strong Passwords: We can encourage all users to choose strong passwords that cannot be found in a dictionary or is directly related to user's personel information like date of birth, name etc. and that are not simple sequences of dictionary words. Requiring users to choose strong passwords is an effective measure of avoiding dictionary attacks. Strong passwords can easily be created by making them as long as possible; including alphabets, numericals, and special characters; and using passwords that are different from those that you have used on other systems. You can also create strong passwords from long phrases. For example, consider the phrase "Rome was not built in a day". If it is easy for you to remember such a quote, you can transform it into a password such as R1ws2N3bt4I5aa6D. The first letter of alternate word in the phrase has been used in uppercase, and other words are transformed into sequence number, first and last letter of the word followed by next number.

"Honeypot" Passwords: To enable catching of attackers who are trying to hack into a password protected system, simple passwords and usernames are used as "honey" to attract the attackers. For example, some systems might have a default user id called "guest" that has the password "guest123." You do not expect normal users to use this guest account. You can set up your system such that if the attacker tries to log in using a default password for the guest user, you can set that as a trigger so that your system administration staff can be notified. When somebody tries logging into it, you know that it may be an indication that an attacker is trying to break into your system. Once the system administration staff is notified that somebody might be trying to break into the system, you can then take action to identify which IP address the attacker is coming from.

Password Filtering: we can test the password strength of the password selected by the user. If the user chooses a password that matches some dictionary pattern we can conclude it as easy to guess and ask the user to choose another password.

Aging Passwords: Every time a user uses a password for the session there is a possibility that an attacker is monitoring the user's activity. So it advantageous for us to prompt the user to change their password periodically , after regular intervals. Or we can limit the user to use each password to a fixed number of logins only.

Pronounceable Passwords: Password security system designers noticed that users sometimes want to choose dictionary words because they are easy to remember. Hence, they decided to create pronounceable passwords that may be easy to remember because users can sound them out, but would not be words in the dictionary. Pronounceable passwords are made up of syllables and vowels connected together that are meant to be easy to remember. Some examples of pronounceable passwords—generated by a package called Gpw (www.multicians.org/thvv/gpw.html) areahrosios, chireckl, and harciefy.

Limited Login Attempts: The number of failed login attempts before locking or disabling the user's account can be limited to some fixed number. The advantage of limited login attempts is that it will reduce the probability of an attacker successfully guessing the password and intruding the user's account. The downside of using this approach is that if a legitimate user happens to incorrectly enter her password just a few times, then her account will be locked, which would then require to ask the system administrator for resetting the password. Another disadvantage of account locking is that it gives an attacker the ability to launch a DoS attack against one or more accounts. For instance, if the attacker gets a lot of usernames in the system and tries a couple of random guesses for each username's password, the attacker can end up locking a large fraction of the user accounts in a system.

Last Login: Another feature that can be provided to increase the security of the password system is that the last date , time, and location about the all the login attempts in the previous day are displayed every time when a user logs in. User can monitor their account easily and check when was their account misused. If a user ever notices an unauthorized behavior or login activity between when and where she last logged in and when and where the system *reported* that she last logged in, she can notify the system administration staff or customer service. For example, if a user usually logs in once a month from her home in Delhi, but upon login, the system informs her that the last time she logged in was at 3 a.m., three days ago in shimla, she will realize that something is wrong. She can then notify the appropriate personnel, and the security issue can be reactively dealt with. If the last login mechanism did not exist, then the occurrence of the attack may not have been noticed.

Image Authentication

One of the latest attempts at making password systems less vulnerable to phishing attacks has been to use images as a second factor in authenticating the user. While the user is creating the account, a user is advised to choose an image in addition to a username and password. When the user is presented with a login page, the user is asked for his username first. Upon entering a username, the user is shown the image that he chose when signing up, in addition to being prompted for his password. The intent of using image authentication is to prevent the user from providing his password to an fake web site. While an fake web site may be able to succesfully imitate a legitimate web site's home page, the impostor will not know what image a user has selected. Hence, after a user enters his username into a web site that uses image authentication, he should not enter his password if the web site does not display the same image that he selected when he signed up.

Yahoo mail implement image authentication to provide better security options to its clients.

One-Time Passwords

The final type of password system that can be incorporated in password secure systems is called a *one-time password* system. In all of the techniques that are available to us so far, one of things that gives the attacker undue advantage and an upperhand is that each user uses a same password multiple times to log into a system. However, every time that a user logs into a system, there is a potential threat that the login session is susceptible to eavesdropping and it is possible for that password to be eavesdropped on or found by an attacker. This is especially a problem if that password has not changed over a long period of time. In a one-time password system, every time a user logs in, the user is expected to log in with a different password. The one-time password system used to be implemented by giving users lists of passwords. These lists were essentially small books full of passwords customized for users each time they would log in. For example, the first time that the user logs in, she would use the first password on the list. The next time she logs in, she would be instructed to use the second password on the list. The system could also choose a random password number and expect the user to enter that number. These lists, however, became cumbersome for users.

Most one-time password systems today are ones in which the user is given some device with a small amount of computing power that is used to compute passwords. The device can be used as a source of passwords. The users, when they log into a system, take out the onetime password device, read off the password from that device, and enter it into the computer system. All the passwords that are generated by this device are based off of some cryptographic algorithm. There is typically some seed (initial value) that is used to generate an entire stream of many passwords over time. That seed is also known by the server. Therefore, given the current time and the seed, the server can check that the password the user is entering is correct. The functionality provided in these one-time password devices are now integrated into PDAs, cell phones, and other mobile devices that users already carry. One-time passwords end up being a very good system for ensuring password security. In fact, some banks have started to give one-time password devices to some of their users in order to log into their web-based bank accounts. Hopefully, there will be more usage of one-time passwords in the future.

REFERENCES:

- Introduction to Network Security – by Neal Krawetz
- Computer Network Security – Jie Wang
- Guide to Computer Network Security – Joseph MiggaKizza
- http://csrc.nist.gov/publications/nistpubs/
- Cisco IronPort URL Filters_datasheet
- Web Screening Model Using Multiple Features (IEEE Publication)
- A High-Performance URL Lookup Engine for URL Filtering Systems (IEEE Publication).
- An Efficient Caching Mechanism for Network-based URL Filtering (IEEE Publication).