SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY COMPUTER SYSTEMS AND NETWORKS DEPARTMENT

Enhancement of SSH Authentication by Public Key Cryptography

تطوير الهويه لل(SSH) عن طريق استعمال تشفير المفتاح العام

A Project Submitted As One Of The Requirements For Obtaining A Bachelor Of Honor Computer Systems And Networks

October 2015

SUDAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
COLLEGE OF COMPUTER SCIENCE & INFORMATION TECHNOLOGY COMPUTER SYSTEMS AND NETWORKS DEPARTMENT

Enhancement of SSH Authentication by Public Key Cryptography

Proposed By:

MAZIN ALI ABDELRAHIM BABKAIR
MOHAMED ALKHTIM AHMED ALFADNI
MUHANNED ADIL OMER AHMED
JADALLAH MOHAMED JADALLAH MOHAMED KHAIR

Signature of Supervisor	**DATE**
Eng. Mohamed Elnour Abd Elhafez	**October 2015**

ACKNOWLEDGEMENT

At the beginning and in the end all thanks belongs to ALLAH, we thank the almighty for giving us the will power and patience to complete this work; truly without hisgrace nothing is achievable.a lot of appreciation and gratitude to the ones who have put their trust on us to complete this research.

Our supervisor:eng\ Mohamed Elnour all the regards and respect to the light of the dark roads we did across:

- T.Majda Omer.
- T.Daliya Mahmud.
- T Intesar Al-Haj

Thanks to every teacher who have taught usthroughout our college years and got us to where we are today.

Thanks to all of our colleagues in batch eight, we are honored to have studied with such a batch… for everyone knows our names, thank you.

ABSTRACT

Remote access is the ability of users/system administrators to access their public and non-public computing resources from external location using the public network (internet), several systems could be used to facilitate this service such as Secure Shell (SSH).

As general, such systems available publicly by default regarding to its nature, therefore in this study we use public key as replacement of passwords in SSH authentication to overcome any expected unauthorized access to our service.

The proposed solution in this study consists of remote access server, remote access client, and certificate authority to make sure that public key is the right one, we added an option in CA of which cryptography algorithm to use for data, using this replacement of public key will lead to better and faster authentication than using passwords, as shown in this research, we used just RSA algorithm for encryption, but we recommend to use other algorithms as options in CA form, and also make this program of ours as a service in Linux kernel.

المستخلص

الوصول عن بعد هو قدرة المستخدمين و مسؤولي النظام للوصول إلى موارد الحاسب العامة و الخاصة بهم من مكان خارجي باستخدام الشبكة العامة (الإنترنت) ،عموما يمكن استخدام عدة أنظمة لتسهيل هذه الخدمة مثل بروتوكول . (SSH) ، طبيعة هذا النوع من الانظمة يجعلها في الاصل متاحة للكل، لذلك في هذه الدراسة فقد تم استخدام المفتاح العمومي كما استبدال كلمات المرور في المصادقة SSH للتغلب على أي وصول غير المصرح به المتوقع لخدمتنا.

الحل المقترحة في هذه الدراسة تتكون من خادم بعيد الوصول، عميل وصول بعيد، ومصدق للتأكد من أن المفتاح العام هو الصحيح.

أضفنا خيارا في CA منها خوارزمية التشفير لاستخدام البيانات، وذلك باستخدام هذا الاستبدال من المفتاح العمومي سيؤدي إلى مصادقة أفضل وأسرع من استخدام كلمات المرور، كما هو مبين في هذا البحث، استخدمنا فقط خوارزمية RSA لتشفير، ولكن من المستحسن ل استخدام خوارزميات أخرى كخيارات في CA شكل، وأيضا جعل هذا البرنامج لنا كخدمة في نواة لينكس.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

CGI	Common Gateway Interface
DES	Data Encryption Standard
DMZ	Demilitarized Zone
DNS	Domain Name Service
FIFO	First In First Out
FTP	File Transfer Protocol
ICMP	Internet Control Message Protocol
IDEA	International Data Encryption Algorithm
IPsec	Internet Protocol Security
MPLS	Multiprotocol Label Switching
NAT	Network Address Translation
NFS	Network File System
NIDS	Network Intrusion Detection System
PKI	Public Key Identifier
PPP	Point-to-point Protocols
SMB	Server Message Block
SSH	Secure Shell
SSL	Secure Sockets Layer
TCP/IP	Transport Control Protocol/Internet Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
VPN	Virtual Private Network

CHAPTER 1

INTRODUCTION

1.1. Background

With new security threats cropping up every day, network managers are understandably protective of their computing assets. Enhanced security measures, however, can inflict significant hardships on legitimate users and can lead to frustration, productivity losses, and dangerous attempts at circumvention of restrictions. Equipping yourself with proper tools for connectivity can make a task easier while still maintaining network security and integrity. Secure Shell (SSH) is one of the most valuable tools in the Information Technology (IT) toolkit. It is typically used for logging into remote servers to provide a shell access to do maintenance, read emails, restart services, or any other task required by administrator. Using SSH as a replacement for Telnet is familiar to most users; however it includes many features and benefits that could be utilized. When properly configured, a server and client can connect and communicate virtually any service using the SSH link, and since it is inherently more secure than Telnet, FTP, or other unencrypted protocols, most network managers are accommodating to requests to open ports and allow SSH communications through the firewall. In this research SSH will be utilized as remote access solution for FAMIS information system.

1.2. Problem Statement

Remote access to networks resources (such as servers, storage) in secure manner are becoming increasingly important, in most cases it provides the possibility of effective administration and management using public network (Internet).

Basically the most common technique for remote system authentication is username and password which is enabled by default, especially in SSH; which is not powerful enough for most of common attacks to break authority of the system.

1.3. Objectives

The proposed objectives are aim to: -

- Replace default login to user of SSH by public key scheme instead of passwords.
- Add a certificate authority to sure that public keys are used for authentication are valid.
- Add algorism to encrypt CA data.

1.4. Scope

Is to design a code that simulate SSH mechanism and working environment with replacing authentication by public key cryptography instead of passwords and create CA to register and make sure that all users of SSH service are authenticated.

1.5. Importance

Security has become an important factor of any organization, big organizations have distributed server to hold their data and information in different locations, access to these servers will be default physically but by using remote access techniques , but will be lack of authentication, so in this project we designed a code that use public key cryptography to enhance authentication of remote access.

1.6. Dissertation Layout

This research is divided into five chapters. Chapter two gives an introduction to remote access solutions, and it shows also the security techniques used with remote access solutions, as well provide an idea about how to assist the security level using security utilities. Chapter three presents the whole remote access solution security design. Chapter4 contains Implementation and Testing. Finally Chapter 5 presents conclusion and the fuller recommendations.

CHAPTER 2

LITERATURE REVIEW

2.1. Introduction

This chapter consist of concepts that is important to our project, and related studies; Organizations have many options for providing remote access to their computing resources. The remote access methods most commonly used for teleworkers have been divided into four categories based on their high-level architectures: tunneling, portals, remote desktop access, and direct application access. The remote access methods in all four categories have some features in common [4].

- They are all dependent on the physical security of the client devices.
- They can use multiple types of server and user authentication mechanisms.
- They can use cryptography to protect the data followed.
- They can allow teleworkers to store data on their client devices [4].

2.1.1.Communication Tunneling

Many remote access methods offer a secure communications tunnel through which information can be transmitted between networks, including public networks such as the Internet. Tunnels are typically established through virtual private network (VPN) technologies. Once a VPN tunnel has been established between a teleworker's client device and the organization's VPN gateway, the teleworker can access many of the organization computing resources through the tunnel. It also uses cryptography to protect the confidentiality and integrity of the transmitted information between the client device and the VPN gateway. In tunneling solutions, the application client software and data at rest resides on the client device, are not protected by the tunneling solution and should be protected by other means. Figure2.1 represents the tunneling architecture [4].

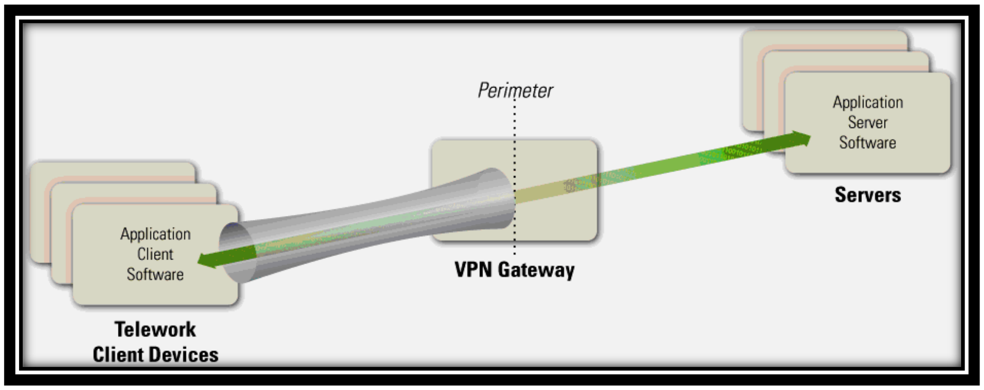


Figure 2.1 Tunneling Architecture [4]

The types of VPNs most commonly used for teleworkers are Internet Protocol Security (IPsec) and Secure Sockets Layer (SSL) tunnels. Tunneling may also be achieved by using Secure Shell (SSH), although this is less commonly used and is often considered more difficult to configure and maintain as compared to IPsec or SSL tunnel VPNs [4].

Many communication encryption protocols can be expanded into tunneling protocols in the same way that SSL is used for SSL VPNs. For example some use un-standardized methods such as SSH protocol to create tunnels. In general, standardized tunneling protocols can be configured to have the same cryptographic strength and to use the typical mechanism for authenticating the two parties to each other. Different tunneling systems can tunnel various protocols; for example, the IPsec has standard extensions that allows to tunnel Layer 2 protocols such as the Point-to-Point Protocol (PPP) and Multiprotocol Label Switching (MPLS). In general, almost any communication encryption protocol can be made to tunnel almost any layer [4].

2.1.2.Application Portals

A portal is a server that offers access to one or more applications through a single centralized interface. A teleworker uses a portal client on a telework client device to access the portal. Most portals are web based; the portal client is a regular web browser. Figure 2.2 shows the basic portal solution architecture. The application client software is installed on the portal server, and it communicates with application server software on servers within the organization. The portal server communicates securely with the portal client as needed [4].

Portals have most of the same characteristics as tunnels: portals protect information between client devices and the portal, and they can provide authentication, access control, and other security services. However, there is an important difference between tunnels and portals. The location of the application client software and associated data. In a tunnel, the software and data are on the client device; in a portal, they are on the portal server. A portal server transfers data to the client device as rendered desktop screen images or web pages, but data is typically stored on the client device much more temporarily than data for a tunneled solution is:

- In Tunnel
- A Portal Server [4]

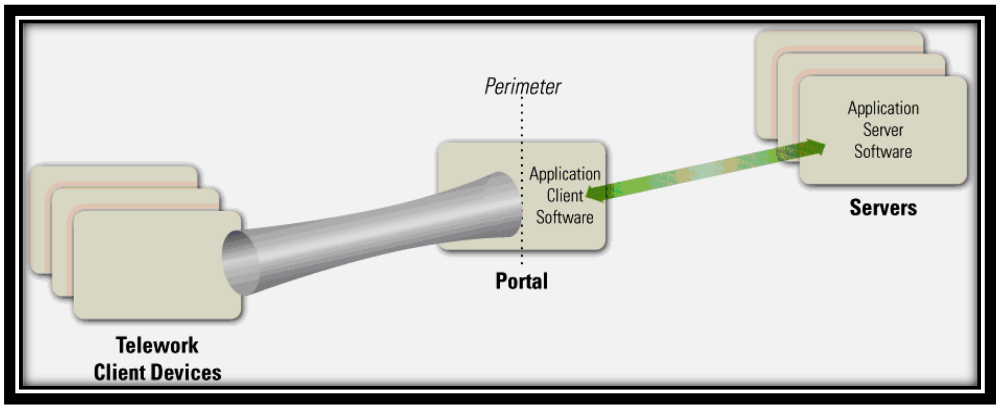


Figure 2.2: Portal Architecture [4]

2.1.3.Remote Desktop Access

A remote desktop access solution gives a teleworker the ability to remotely control a particular desktop computer at the organization, most often the user's own computer at the organization office, from a telework client device. The teleworker has keyboard and mouse control over the remote computer and monitor that computer's screen on the local telework client device's screen. Remote desktop access allows the user to access all of the applications, data, and other resources that are normally available from their computer in the office. Figure 2.3 shows the basic remote desktop access architecture [4].

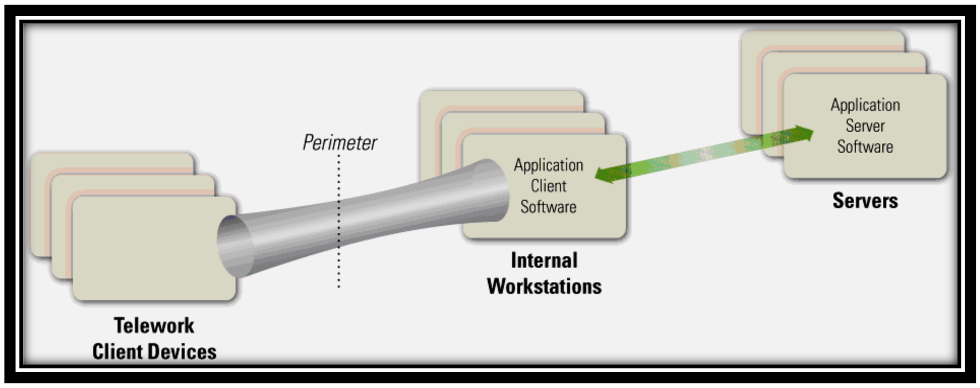


Figure 2.3 Remote Desktop Access Architecture [4]

2.1.4.Direct Application Access

Remote access can be accomplished without using remote access software. A teleworker can access an individual application directly, with the application providing its own security (communications encryption, user authentication, etc.) Figure 2.4 shows the high-level architecture for direct application access. The application client software installed on the telework client device initiates a connection with a server, which is typically located at the organization's perimeter zone [4].

The direct application access architecture is generally only acceptable if the servers being accessed by the teleworkers are located on the organization's network perimeter, and not internal networks. Servers on the perimeter are directly accessible from the Internet, so they should be well-secured to reduce the likelihood of compromise. Many organizations choose to provide direct application access to only a few lower-risk applications that are widely used, such as email, and use tunnel or portal methods to provide access to other applications, particularly those that would be at too much risk if they were directly accessible from the Internet [4].

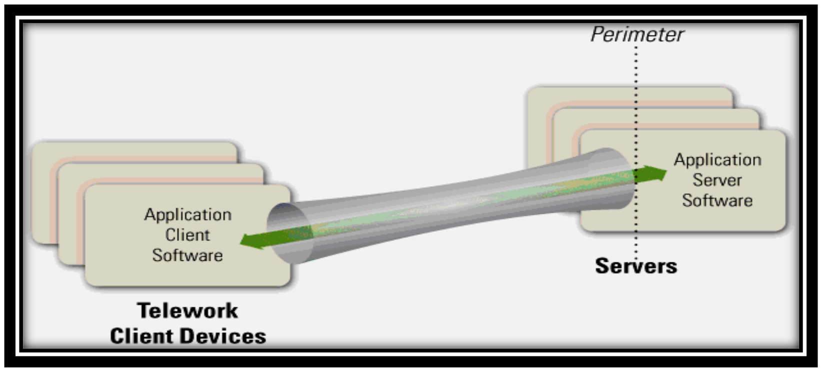


Figure 2.4 Direct Application Access Architecture [4]

2.2. Remote Access Authentication, Authorization, and Access Control

Most of the computing resources used through remote access are available only to an organization user, and often only a subset of those users. To ensure that access is restricted properly, remote access servers should authenticate each teleworker before granting any access to the organization resources, and then use authorization technologies to ensure that only the necessary resources can be used [4].

2.2.1.Remote Access Authentication

There are many ways to authenticate remote access users, there include the use of passwords, digital certificates, or hardware authentication tokens. If passwords are the only form of authentication for a remote access solution, then generally the remote access solution authentication mechanism should be different from the organization other authentication mechanisms, such as email or directory service passwords, unless direct application access is being used. Having different passwords reduces the impact that a compromise of remote access credentials would have on other information resources, and vice versa, and it is particularly important if users are entering passwords into telework devices not controlled by the organization. However, having different passwords for remote access and other systems is often not enforceable, and it should be assumed that some users will use the same passwords for both. Organizations with higher security needs or with concerns about the security of passwords should consider using authentication that does not rely solely on passwords, such as multifactor authentication [4].

2.2.2.Remote Access Authorization

After verifying the identity of a remote access user, organizations may choose to perform checks involving the telework client device to determine which internal resources the user should be permitted to access. These checks are sometimes called health, suitability, screening, or assessment checks. The most common way of implementing this is having the remote access server perform “health checks” on the teleworker client device. These health checks require software on the system of user that controlled by the remote access server to verify compliance with certain requirements from the organization secure configuration baseline. The user antimalware software being up-to-date, the operating system being fully patched, and the system being owned and controlled by the organization. If the user has acceptable authorization credentials but the client device does not pass the health check, then the user and device may be granted limited access to the internal network. i.e no network access at all, or access to a quarantine network so that the security deficiencies can be fixed. This decision can also be based on the part of the network that the device is trying to access [4].

2.2.3.Access Control for Network Communications

A major component of controlling access to network communications and protecting their content is the use of cryptography. At a minimum, any sensitive information passing over the Internet, wireless networks, and other untrusted networks should have the confidentiality and integrity preserved through use of cryptography. Some remote access methods such as internet protocol security (IPsec) and secure sockets layer (SSL) virtual private networks (VPN), often inherently mechanisms for encrypting communications and verifying their integrity. Other remote access methods may use Transport Layer Security (TLS) or other cryptographic mechanisms to provide protection [4].

Access control for network communications may also involve determining which traffic should be protected. Some remote access solutions offer options for this; for example, many VPN clients have a feature called split tunneling where it is tunnel all communications involving the organization internal resources through the VPN, and protecting them, however, it will exclude all other communications from going through the tunnel [4].

2.2.4.Access Control for Applications

Different types of remote access architectures offer different levels of granularity for application access control. Tunnels often have a mechanism for an administrator to specify which ports on which hosts the teleworker has access to; this can limit access so that only specific applications can be used. Portals, by their nature, limit the teleworker to applications run on the portal server. Similarly, direct application access limits the teleworker to a specific application on a single server. Remote desktop access can only provide access control to applications by combining its policies with the access control restrictions that are in place on the internal workstations [4].

2.3. Secure Shell (SSH)

SSH is a program to log into computers, switches ... etc over a network, to execute commands in a remote machine, and to move files from one machine to another. It provides strong authentication and secure communications over unsecure channels. It is intended as a replacement for telnet, rlogin, rsh, and rcp. For SSH2, there is a replacement of file transfer protocol (FTP), it is secure file transfer protocol (SFTP). Additionally, SSH provides secure X connections and secure forwarding of arbitrary TCP connections. There is ability to use SSH as a tool for things like rsync and secure network backups. The traditional BSD 'r' - commands (rsh, rlogin, rcp) are vulnerable to different kinds of attacks. Somebody who has root access to machines on the network, or physical access to the wire, can gain unauthorized access to the systems in a variety of ways. It is also possible for such person to log all the traffic to and from our system, including passwords (which SSH never sends in the clear text) [5, 6].

There are two versions of Secure Shell available: SSH1 and SSH2. Thus, it should be noted that the SSH1 and SSH2 protocols are in fact different and not compatible with each other [5, 6].

SSH uses the ciphers shown in Table 2.1 for encryption.

Table 2.1 Encryption Ciphers [5]

Cipher	SSH1	SSH2
DES	Yes	No
3DES	Yes	Yes
IDEA	Yes	No
Blowfish	Yes	Yes
Twofish	No	Yes
Arcfour	No	Yes
Cast128-cbc	no	Yes

SSH Uses the following ciphers for authentication:

Table 2.2 Authentication Ciphers [5]

Cipher	SSH1	SSH2
RSA	yes	Yes
DSA	no	Yes

SSH versions (SSH1 and SSH2) are support different types of authentications, it is as follow.

- Password (the /etc/passwd or /etc/shadow in UNIX).
- User public key (RSA or DSA, depending on the release).
- Kerberos (for SSH1).
- Hostbased (.rhosts or /etc/hosts.equiv in SSH1 or public key in SSH2) [5].

2.4. Public Key Cryptography

Public-key cryptography refers to a set of cryptographic algorithms that are based on mathematical problems that currently admit no efficient solution -- particularly those inherent in certain integer factorization, discrete logarithm, and elliptic curve relationships. It is computationally easy for a user to generate a public and private key-pair and to use it for encryption and decryption. The strength lies in the "impossibility" (computational impracticality) for a properly generated private key to be determined from its corresponding public key. Thus the public key may be published without compromising security. Security depends only on keeping the private key private. Public key algorithms, unlike symmetric key algorithms, do not require a secure channel for the initial exchange of one (or more) secret keys between the parties.

Because of the computational complexity of asymmetric encryption, it is typically only used for short messages, typically the transfer of a symmetric encryption key. This symmetric key is then used to encrypt the rest of the potentially long & heavy conversation. The symmetric encryption/decryption is based on simpler algorithms and is much faster.

Message authentication involves hashing the message to produce a "digest," and encrypting the digest with the private key to produce a digital signature. Thereafter anyone can verify this signature by (1) computing the hash of the message, (2) decrypting the signature with the signer's public key, and (3) comparing the computed digest with the decrypted digest. Equality between the digests confirms the message is unmodified since it was signed, and that the signer, and no one else, intentionally performed the signature operation — presuming the signer's private key has remained secret. The security of such procedure depends on a hash algorithm of such quality that it is computationally impossible to alter or find a substitute message that produces the same digest - but studies have shown that even with the MD5 and SHA-1 algorithms, producing an altered or substitute message is not impossible. The current hashing standard for encryption is SHA-2. The message itself can also be used in place of the digest.

Public-key algorithms are fundamental security ingredients in cryptosystems, applications and protocols. They underpin various Internet standards, such as Transport Layer Security (TLS), S/MIME, PGP, and GPG. Some public key algorithms provide key distribution and secrecy (e.g., Diffie–Hellman key exchange), some provide digital signatures (e.g., Digital Signature Algorithm), and some provide both (e.g., RSA).

Public-key cryptography finds application in, amongst others, the IT security discipline information security. Information security (IS) is concerned with all aspects of protecting electronic information assets against security threats. Public-key cryptography is used as a method of assuring the confidentiality, authenticity and non-reputability of lectronic communications and data storage. [6]

2.5. Certificate Authority

In cryptography, a certificate authority or certification authority (CA) is an entity that issues digital certificates. A digital certificate certifies the ownership of a public key by the named subject of the certificate. This allows others (relying parties) to rely upon signatures or on assertions made by the private key that corresponds to the certified public key. In this model of trust relationships, a CA is a trusted third party—trusted both by the subject (owner) of the certificate and by the party relying upon the certificate. Many public-key infrastructure (PKI) schemes feature CAs. [6]

2.6. Related Studies

The following is a summary of recent work related to this research.

“Adding Public Key Security to SSH” [1]

The author of this paper presents an investigation of the effects of the “man-in-the-middle Attack” and provides a solution to avoid the inherent security flaw by using the concept of the public key.

However, SSH protocol has an inherent security flaw. It is vulnerable to the “man-in-the-middle Attack”, when a user establishes his first SSH connection from a particular client to a remote machine. My thesis entails designing, evaluating and prototyping a public key infrastructure which can be used with the SSH2 protocol, in an academic setting, thus eliminating this vulnerability due to the man in the middle attack. The approach presented is different from the one that is based on the deployment of a Certificate Authority. My scheme does not necessarily require third party verification using a Certificate Authority; it is decentralized in nature and is relatively easy to set up [1].

“Tunneling TCP over TCP – A study of a real system” [2]

The work focuses on the effects of TCP in TCP in VPNs. The aim was to test the impact on performance by using a Virtual Ethernet Interface, to tunnel a TCP connection through another TCP connection. The authors wanted to test the theoretical claims, that problems such as TCP meltdown will occur, causing decreased good put [2].

Questions raised by the authors were:

- Is TCP in TCP an actual problem which would confirm the theoretical claims?
- If it is a problem, when do the problems occur?
- Is it realistic that the levels of packet loss will occur, which is needed for triggering the problem?

To investigate, a test system was created. The test system consisted of three computers, two of the computers where end hosts and the third computer acted as a forwarding router between the end hosts. The middle computer had the purpose of emulating packet loss that can occur on a real, bigger, network [2].

In the tests the emulator Linux Traffic control Network Emulator, tcNetem, was used. In the tests only unidirectional packet loss was emulated. Ethereal was used to monitor the traffic. The tests where performed using SSH as VPN solution. Two different VPN configurations where tested. One was created using the port forwarding feature of SSH. The other by using the IP forwarding features [2].

The results of the simulations performed by the authors have showed that the TCP meltdown does not occur for realistic levels of packet loss, i.e. lower than 10%. The decrease in bandwidth is only noticeable when the packet loss consists of ACKs and is approximately unchanged for packet loss of packets containing payload data [2].

CHAPTER 3

SYSTEM DESIGN AND IMPLEMENTATION

3.1 Introduction

This chapter consist of methodology of the work in this research, model of proposed design with explaining of each component of system, system analysis using UML, and implementation of project.

3.2 Methodology

The theme of the research that we wish to carryout is divided into three main phases, are described below:

Phase 1 (Analysis): This part of the research that by the end of it, the general perception about virtual project laboratories systems and perform special (security) analysis of OpenSSH, Snort and iptables.

Phase 2 (Implementation and Design): According to the software configurations and its extended application we need to go through five essential design issues, it is requirement specification, holistic view of the system architecture, implementation, and design of security monitoring and reporting interfaces.

Phase 3 (Testing and Modification): This part is about testing the whole project in order to verifying its capabilities and limitations then solve any discovered malfunctioning. The expected duration is estimated to be one month.

3.3 Proposed Design

As any other information system, the proposed design consist of main server where the projected has been build, the following figure (Figure 3.1) shows the whole system connectivity and network traffic flow.

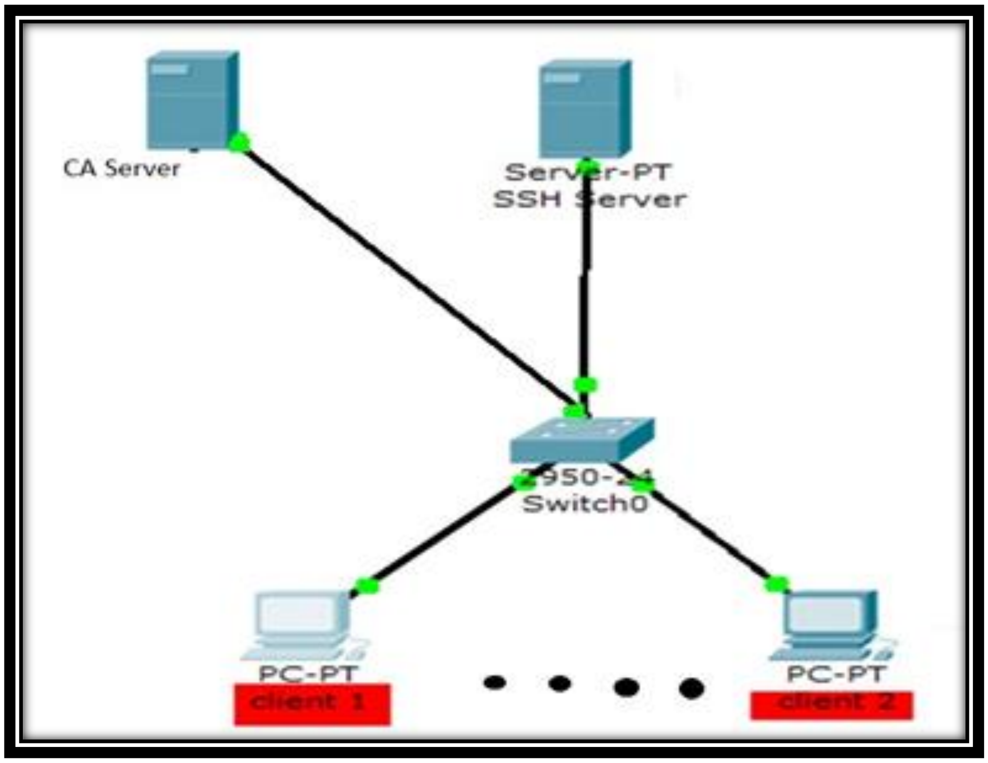


Figure 3.1: System Design Diagram

3.1.1 System Components

- Server: which receives IDs from clients then decrypts it and compare IDs with certificate authority public keys to authorize authentication of clients.
- clients: which send a message contains of two pair ; first part is ID of client in plain form , the other part is also ID but is encrypted by RSA algorithm then client send message to server to complete operation.

3.1.2. Network Connectivity

The system is connected to the internet directly with a public internet protocol (IP). According to the design.

3.1.3. Certificate Authority (CA)

certificate authority is consist of table of authorized or non-authorized public keys and other information like E-mail owner name authorization validation date (beginning and /or ending) encryption algorithms option and other information about as full CA.

3.1.4. Remote Access Solution Policy

SSH is the only remote access solution that has been allowed in the system for the technical suppliers. By default SSH server is protected against attacks that include:

- IP spoofing, where a remote host sends out packets which pretend to come from another, trusted host. SSH even protects against a spoofer on the local network, who can pretend he/she is the router to the outside.
- IP source routing, where a host can pretend that an IP packet comes from another trusted host.
- DNS spoofing, where an attacker forges name server records.
- Interception of cleartext passwords and other data by intermediate hosts.
- Manipulation of data by the staff in control of intermediate hosts.
- Attacks based on listening to X authentication data and spoofed connection to the X11 server.

In other words, SSH never trusts the net somebody hostile has taken over the network can only force SSH to disconnect but cannot decrypts or plays back the traffic or hijacks the connection.

SSH will not help users with anything that compromises the host security by some means. Once an attacker has gained the root access to a machine, the SSH be subverted. Also if the malevolent has access to user's home directory, then security is nonexistent. This is similar to the case if a home directory is exported via Network file service (NFS). SSH is very fixable to be modified it to add more restriction policy. In this research the following restriction features will be enabled:

- Restricts communication to version 2 of SSH.
- Forbids root user from connection 2th version using SSH. This forces the use of 'su' to gain root access to the machine.
- Denies access to accounts with empty passwords.
- Reduces the amount of time that SSH server allows the clients to presents their credentials.
- Using Public key identifier to authenticate rather than passwords.

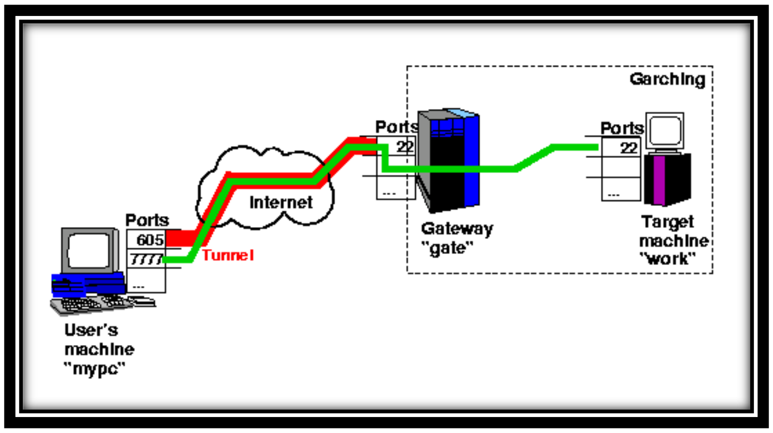


Figure 3.2 SSH Port Forwarding (SSH tunnel) [13]

We have CA server contain basic information included public key of any client in network to authorize users of system in this project passwords in authentication has been replaced by public /private key cryptography, client send id to the other pair in two parts one of them is encrypted and the other is in plain mode, intercepted one is by private key, server decrypt encrypted id then compare to the original one, if they match then user is authorized and if not server well refuse connection.

3.2. System Analysis

Unified Modeling Language (UML) is an industry-standard graphical language for analyzing, describing and documenting the components of an object-oriented system under development and Use graphical notation to communicate more clearly than natural language which is imprecise and coding which is too detailed.

UML was used to analyze and describe the system.

3.3. UML Language

Unified Modeling Language (UML) is a standardized general-purpose modeling language .includes a set of graphic notation techniques to create visual models of object-oriented software-intensive systems. The proposed system analyzed using use case, Sequence and Activity diagrams to emphasize what must happen in the system being modeled. These behavior diagrams illustrate the behavior of a system; they are used extensively to describe the functionality of software systems.

The Use case diagrams: Use case diagrams are behavior diagrams used to describe the interactions that take place between actors and the systems during the execution processes. A use case represents a part of the functionality of the system and enables the user (modeled as an actor) to access this functionality.

The sequence diagram: Shows how objects communicate with each other in terms of a sequence of messages. Also indicates the lifespan of objects relative to those messages. It is a construct of a message sequence classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. A sequence diagram4.1shows. Sequence diagrams typically are associated with use case realizations in the Logical View of the system under development.

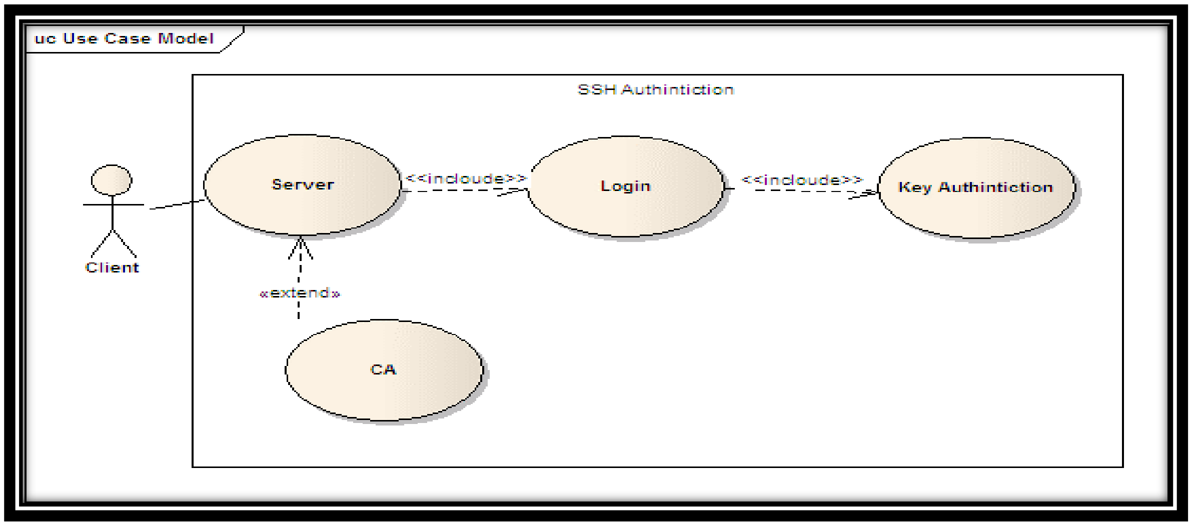


Figure 3.3 User Case Diagram of SSH Authentication

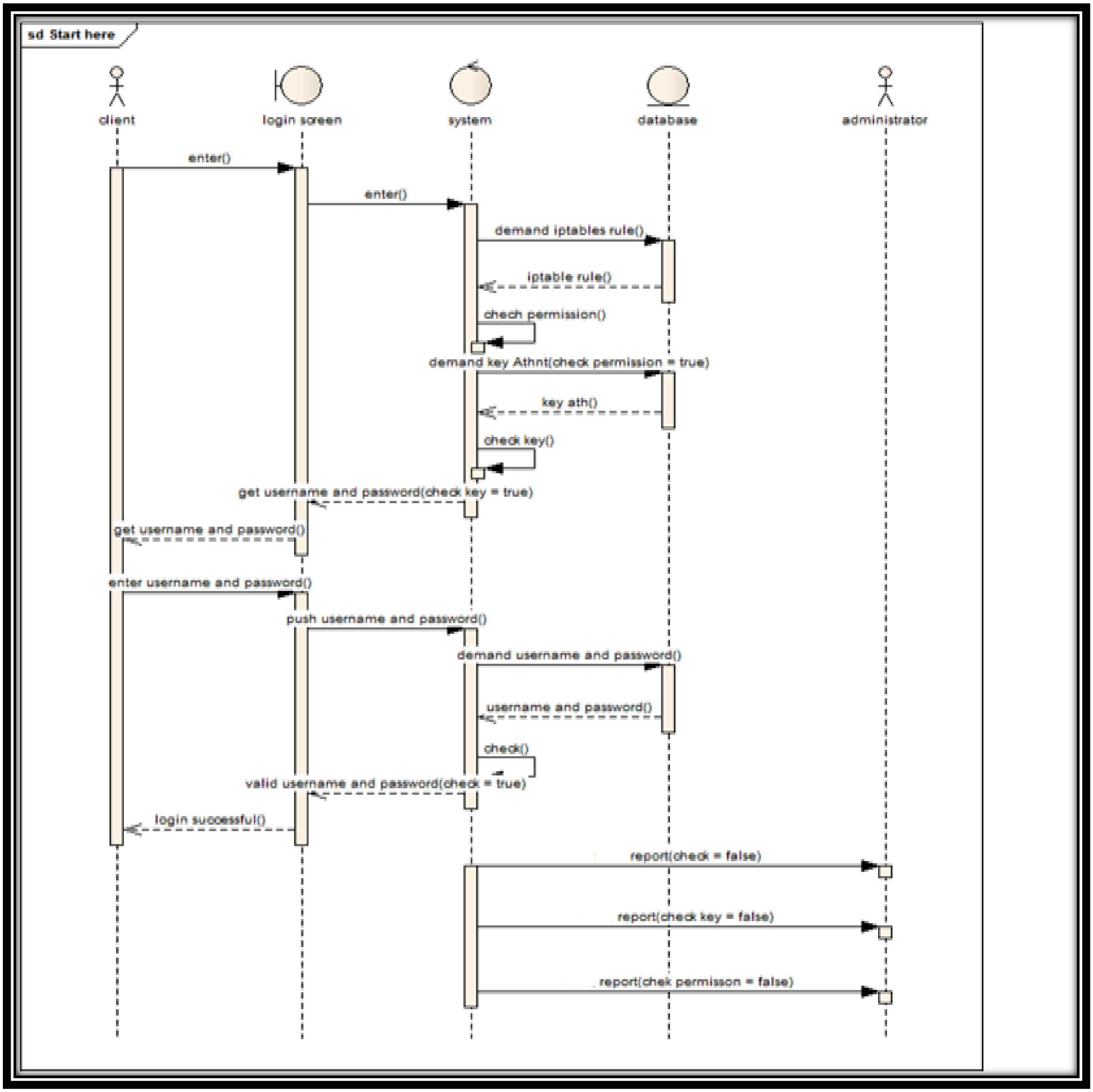

sd Start here
client
login screen
system
database
administrator
enter()
enter()
demand iptables rule()
iptable rule()
chech permission()
demand key Athnt(check permission = true)
key ath()
check key()
get username and password(check key = true)
get username and password()
enter username and password()
push username and password()
demand username and password()
username and password()
check()
valid username and password(check = true)
login successful()
report(check = false)
report(check key = false)
report(chek permisson = false)

Figure 3.4 Sequence Diagram SSH Authentication

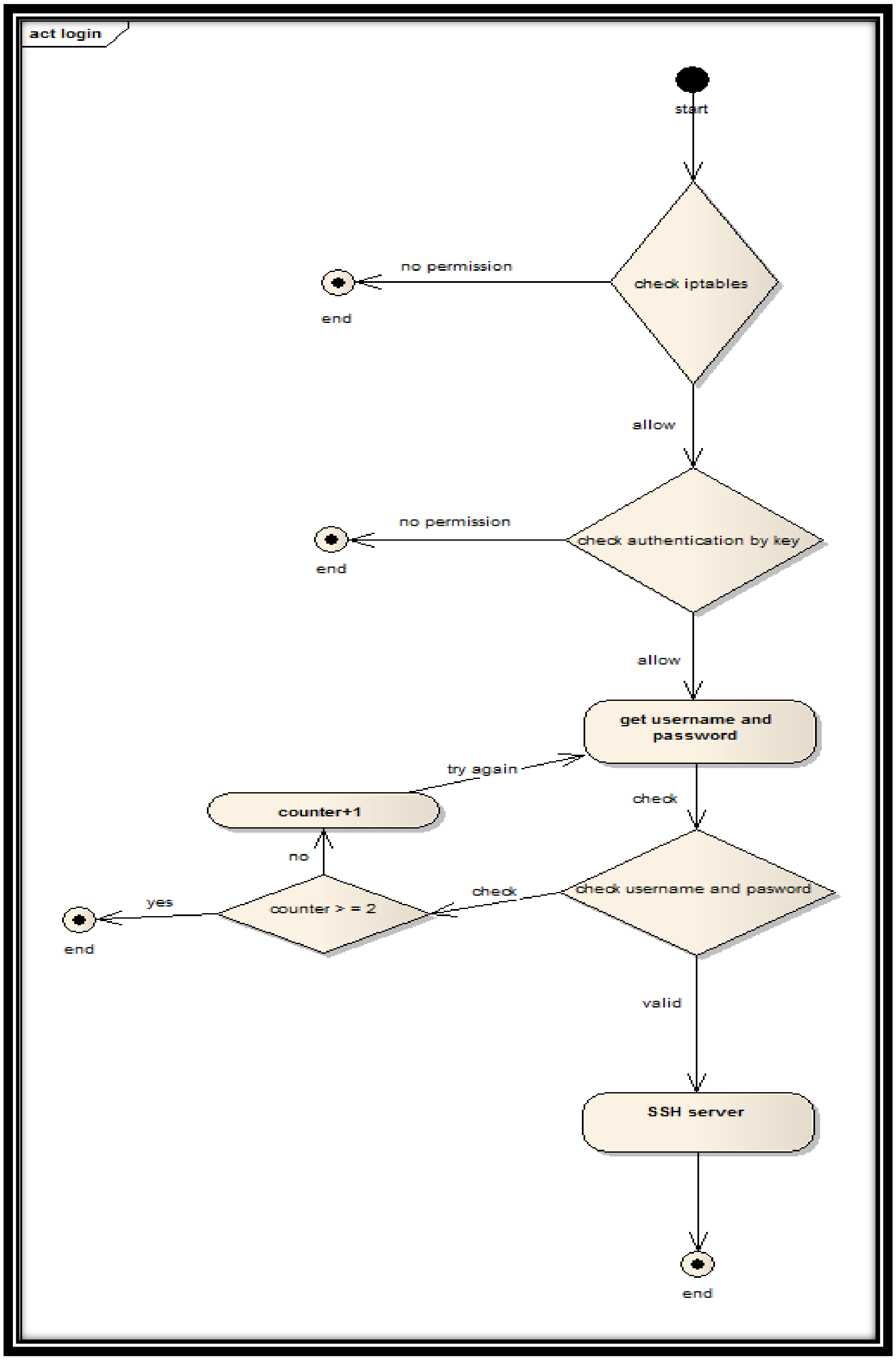


Figure 3.5 Activity Diagram of SSH Authentication

3.4. System Implementation

This part showing in details how all the system components has been implemented in steps supported by screenshots.

3.4.1. Research Environment Description

Many application software and packages has been used, and it is summarized in the following manner.

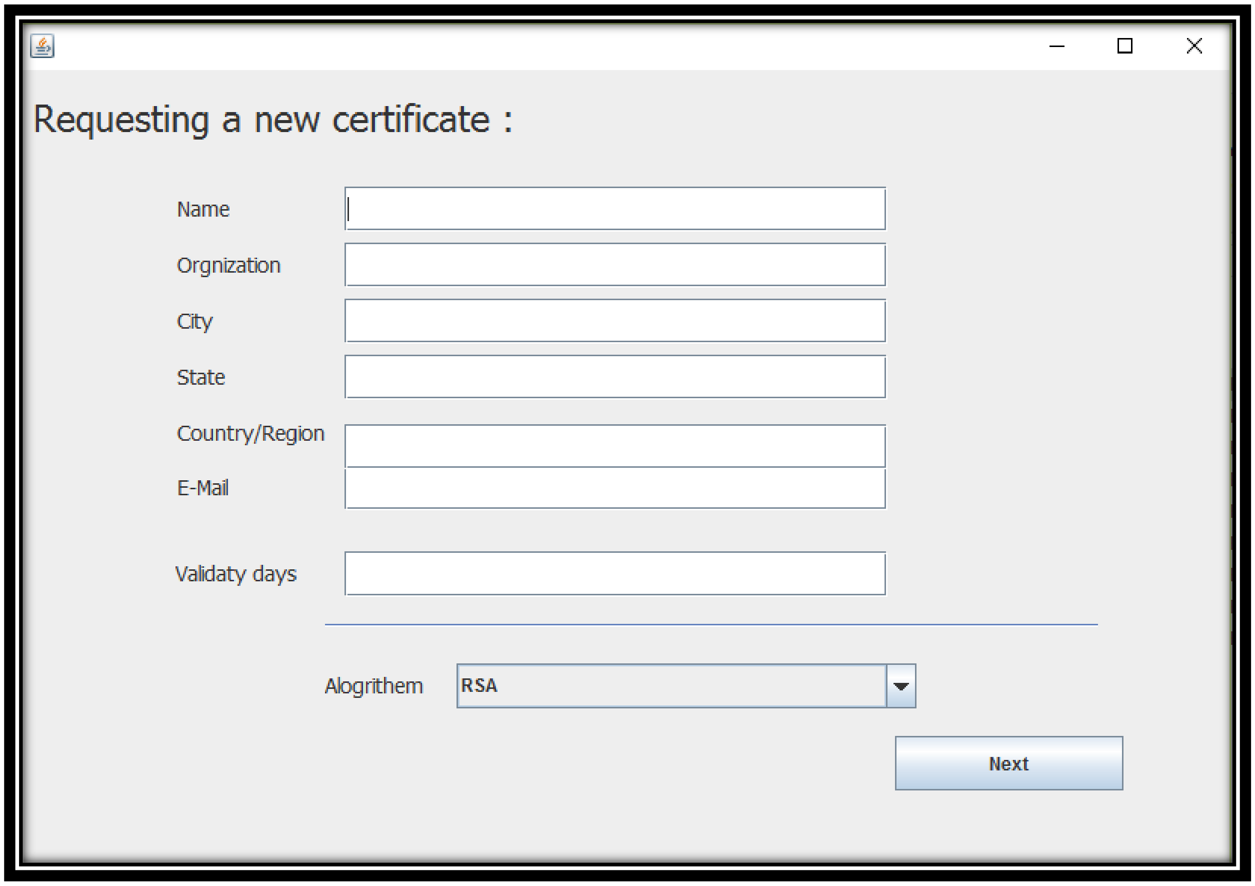
Operating system	: Linux CentOS 6.3
Secure shell (SSH)	:OpenSSH_4.3
Java language	: version 7
C language	:version 4.1+
Net beans	:6.0
Enterprise architect	:7.0.0.4

3.4.2 Certificate Authority (Java Program)

The program asks for basic information to create a certification by public key, the following are type for information in program :

- Name: is refer to Device name .
- Organization: or company the user or client is belong to.
- City: location information about user.
- State: also for location information.
- Country/Region: location information.
- E-mail: for verification and contact with user.
- Validation Date: is the time between that certification is valid.

- Encryption algorithm: to choose which algorithm to encrypt (recommended to add more algorithms).
- Public key: which is used instead of password for authentication.


Requesting a new certificate :

Name	
Orgnization	
City	
State	
Country/Region	
E-Mail	
Validaty days	
Alogrithem	RSA

Figure 3.6 CA Form

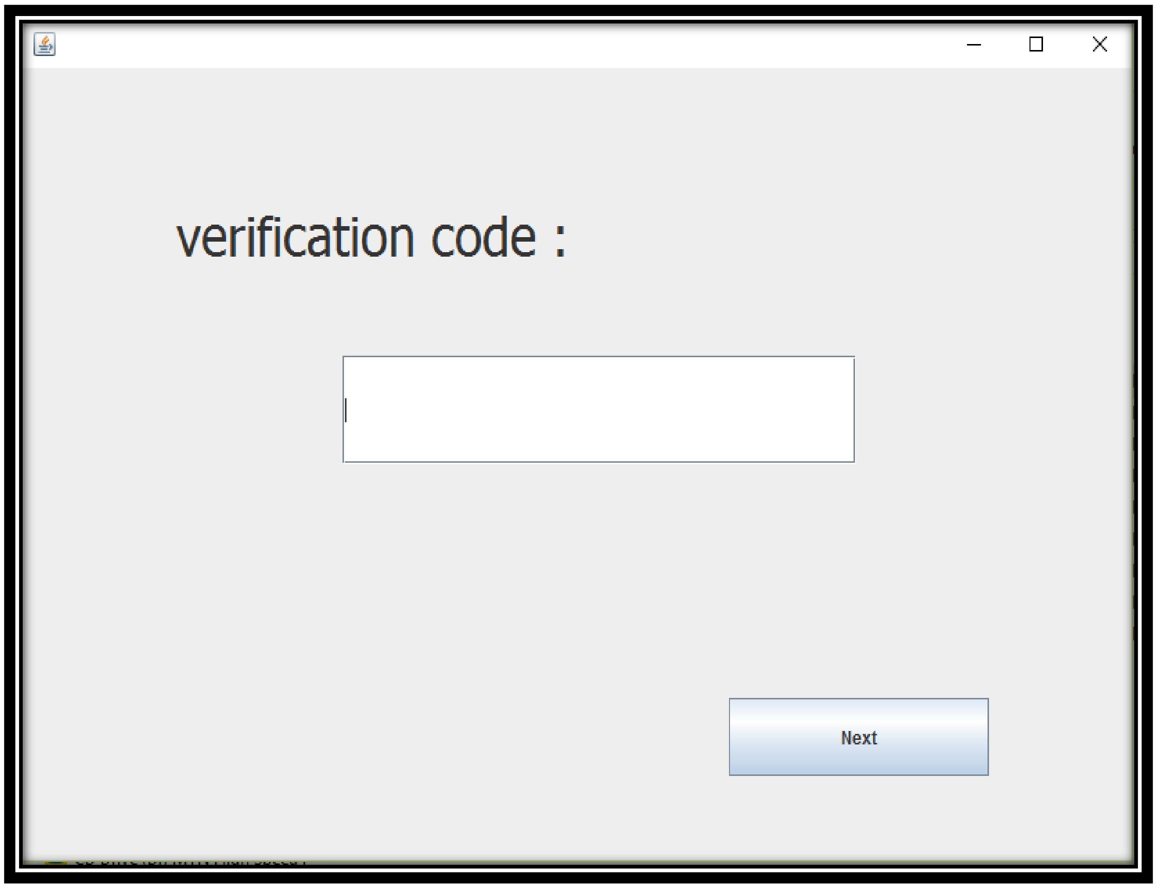
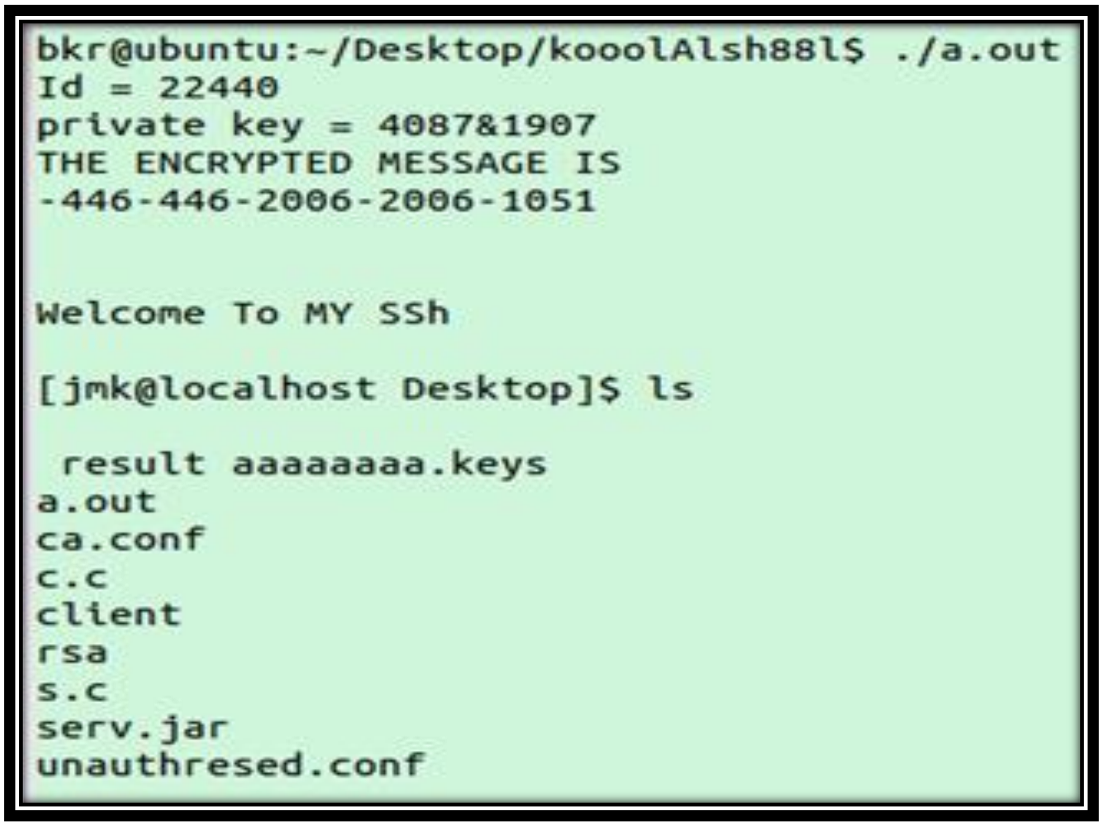


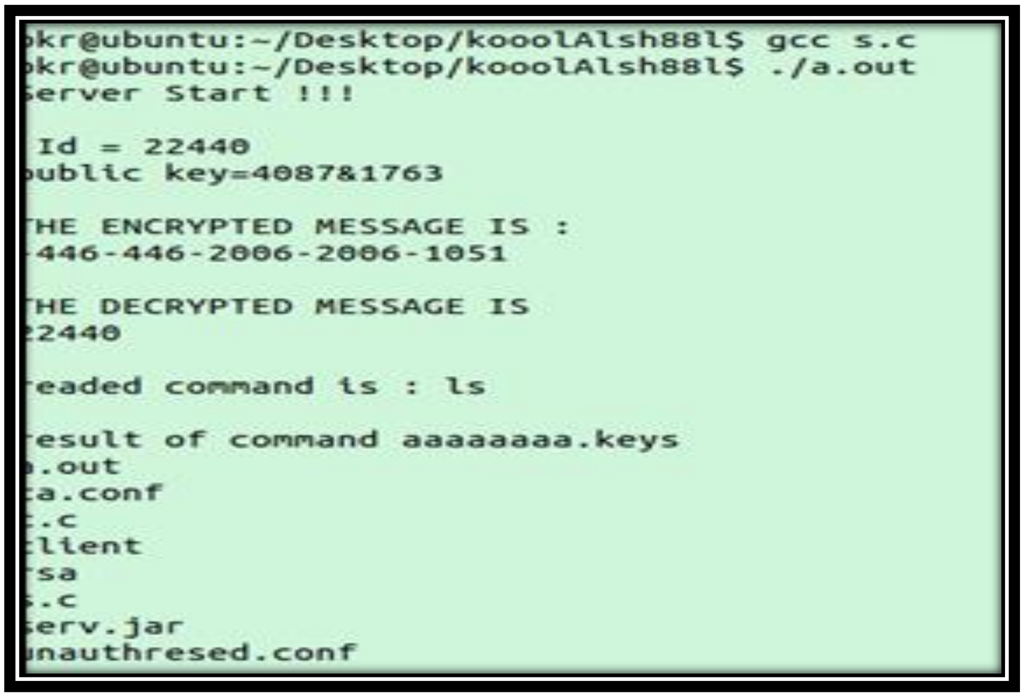
Figure 3.7 CA Validation Field

After filling the form and press ok the server will verify your information by sending a number code to clients email, client should type the same code to verify its citification, server will check the code; if its match server will register the client to a file in clients device; if code does not match the server will reject the operation within error message.



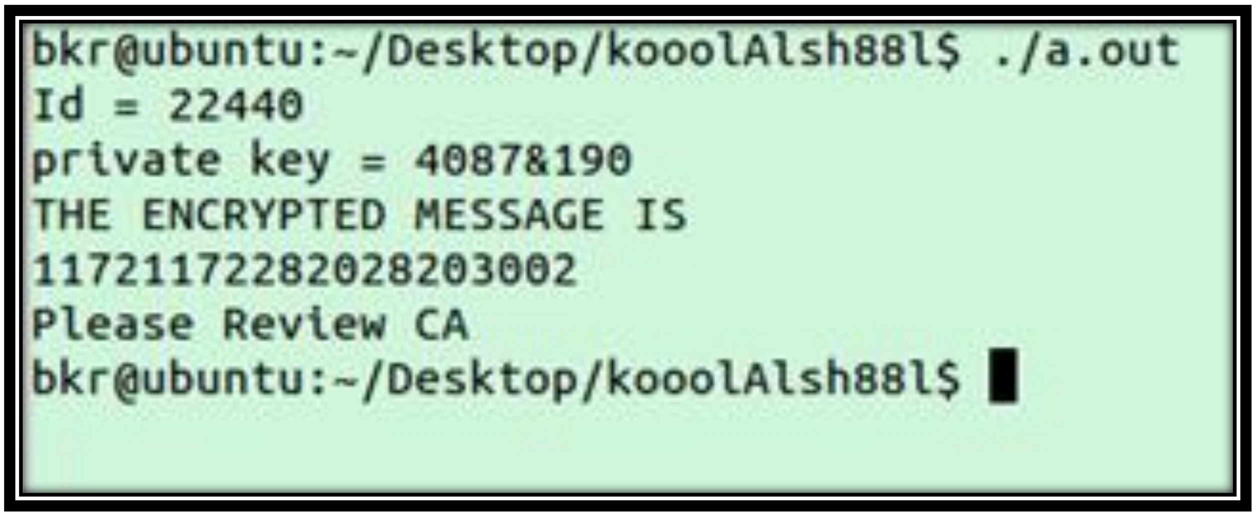
```
bkr@ubuntu:~/Desktop/kooolAlsh88l$ ./a.out
Id = 22440
private key = 4087&1907
THE ENCRYPTED MESSAGE IS
-446-446-2006-2006-1051


Welcome To MY SSh

[jmk@localhost Desktop]$ ls

 result aaaaaaa.keys
a.out
ca.conf
c.c
client
rsa
s.c
serv.jar
unauthresed.conf
```

Figure 3.8 Execute Command Client



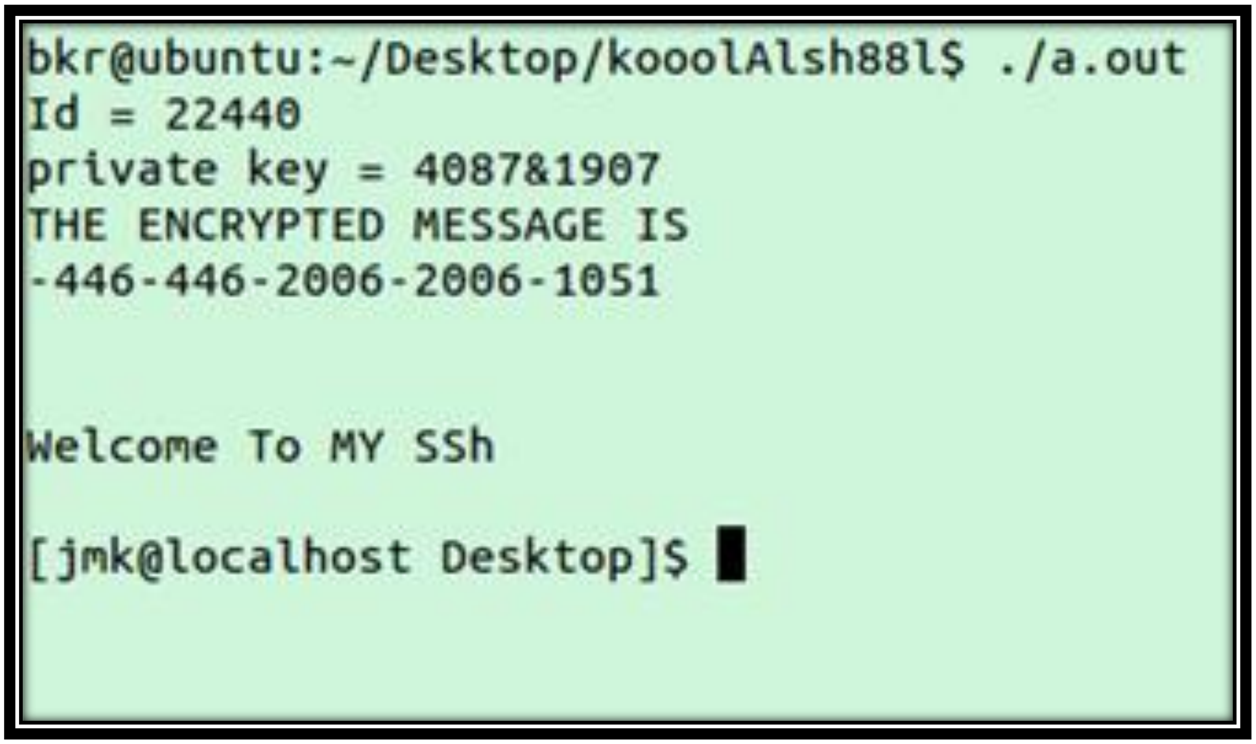
```
kr@ubuntu:~/Desktop/kooolAlsh88l$ gcc s.c
kr@ubuntu:~/Desktop/kooolAlsh88l$ ./a.out
erver Start !!!

 Id = 22440
ublic key=4087&1763

HE ENCRYPTED MESSAGE IS :
446-446-2006-2006-1051

HE DECRYPTED MESSAGE IS
2440

eaded command is : ls

esult of command aaaaaaa.keys
.out
a.conf
.c
lient
sa
.c
erv.jar
nauthresed.conf
```

Figure 3.9 Execute Command Server



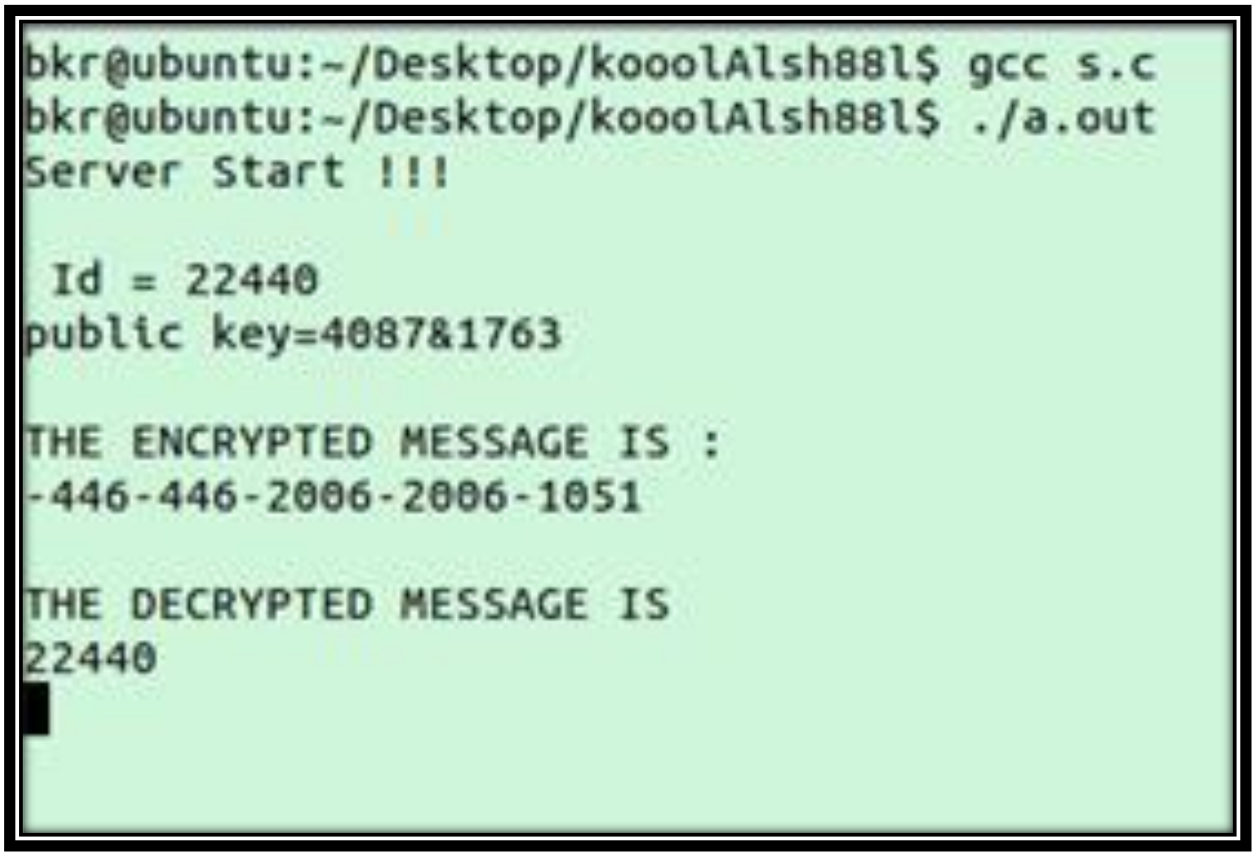
```
bkr@ubuntu:~/Desktop/koooolAlsh88l$ ./a.out
Id = 22440
private key = 4087&190
THE ENCRYPTED MESSAGE IS
117211722820282030O2
Please Review CA
bkr@ubuntu:~/Desktop/koooolAlsh88l$ █
```

Figure 3.10 Login Failed Client



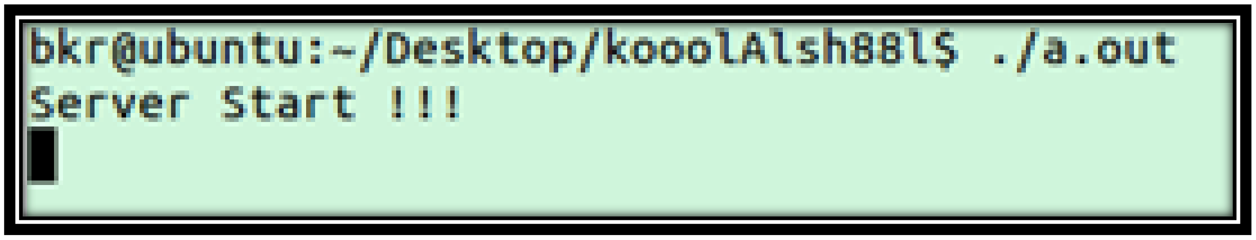
```
bkr@ubuntu:~/Desktop/koooolAlsh88l$ ./a.out
Id = 22440
private key = 4087&1907
THE ENCRYPTED MESSAGE IS
-446-446-2006-2006-1051


Welcome To MY SSh

[jmk@localhost Desktop]$ █
```

Figure 3.11 Login Success Client



```
bkr@ubuntu:~/Desktop/kooolAlsh88l$ gcc s.c
bkr@ubuntu:~/Desktop/kooolAlsh88l$ ./a.out
Server Start !!!

 Id = 22440
public key=4087&1763

THE ENCRYPTED MESSAGE IS :
-446-446-2006-2006-1051

THE DECRYPTED MESSAGE IS
22440
```

Figure 3.12 Login Success Server



```
bkr@ubuntu:~/Desktop/kooolAlsh88l$ ./a.out
Server Start !!!
```

Figure 3.13 Run Server

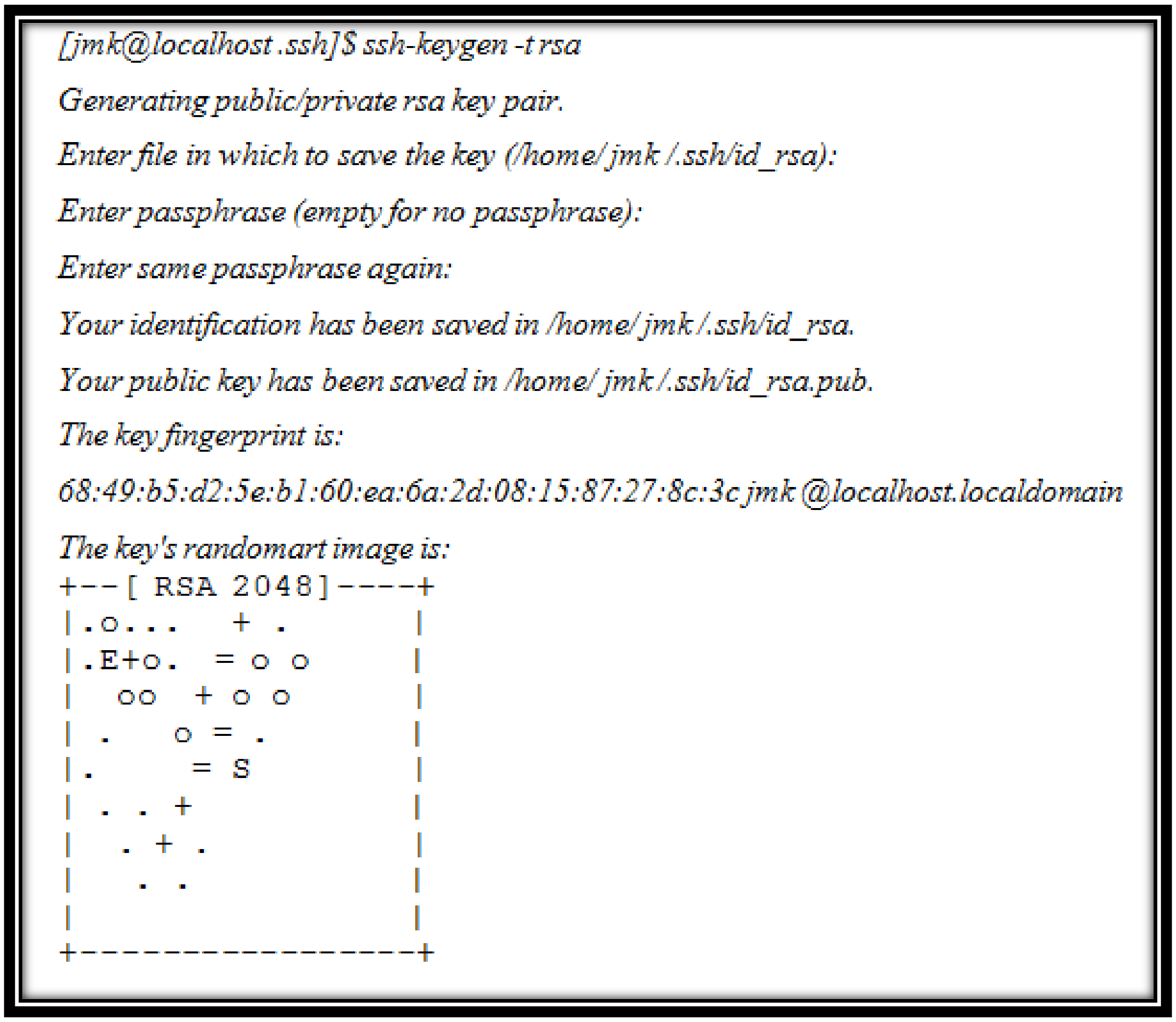
3.4.3 SSH Configuration

This part shows the steps to configure the server and the clients to use PKI (Public Key Identifier) authentication rather than password.

3.4.3.1 CLIENT SIDE (CONFIGURE PKI "PUBLIC KEY IDENTIFIER")

The configuration steps bellow suppose the user name "coif_sifsia-fsts" is already created in the server and it will be used as dedicated user for the SSH access.

- **Generating public RSA key at the client.**

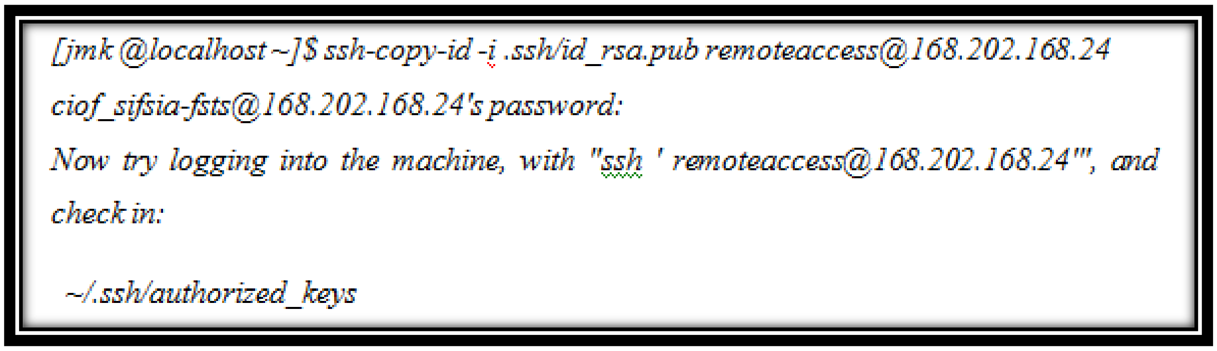


```
[jmk@localhost .ssh]$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ jmk /.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ jmk /.ssh/id_rsa.
Your public key has been saved in /home/ jmk /.ssh/id_rsa.pub.
The key fingerprint is:
68:49:b5:d2:5e:b1:60:ea:6a:2d:08:15:87:27:8c:3c jmk @localhost.localdomain
The key's randomart image is:
+--[ RSA 2048]----+
|.o...    + .     |
|.E+o.  = o o     |
|  oo  + o o      |
| .    o = .      |
|.      = S       |
| . . +           |
|  . + .          |
|   . .           |
|                 |
+-----------------+
```

Figure 3.14 RSA Mechanisms

- **Copying the client public key to the remote server.**

as showed in the following figure :



```
[jmk @localhost ~]$ ssh-copy-id -i .ssh/id_rsa.pub remoteaccess@168.202.168.24
ciof_sifsia-fsts@168.202.168.24's password:
Now try logging into the machine, with "ssh ' remoteaccess@168.202.168.24'", and
check in:

  ~/.ssh/authorized_keys
```

Figure 3.15 Public Key Coping Mechanisms in CA

To make sure we haven't added extra keys that you weren't expecting.

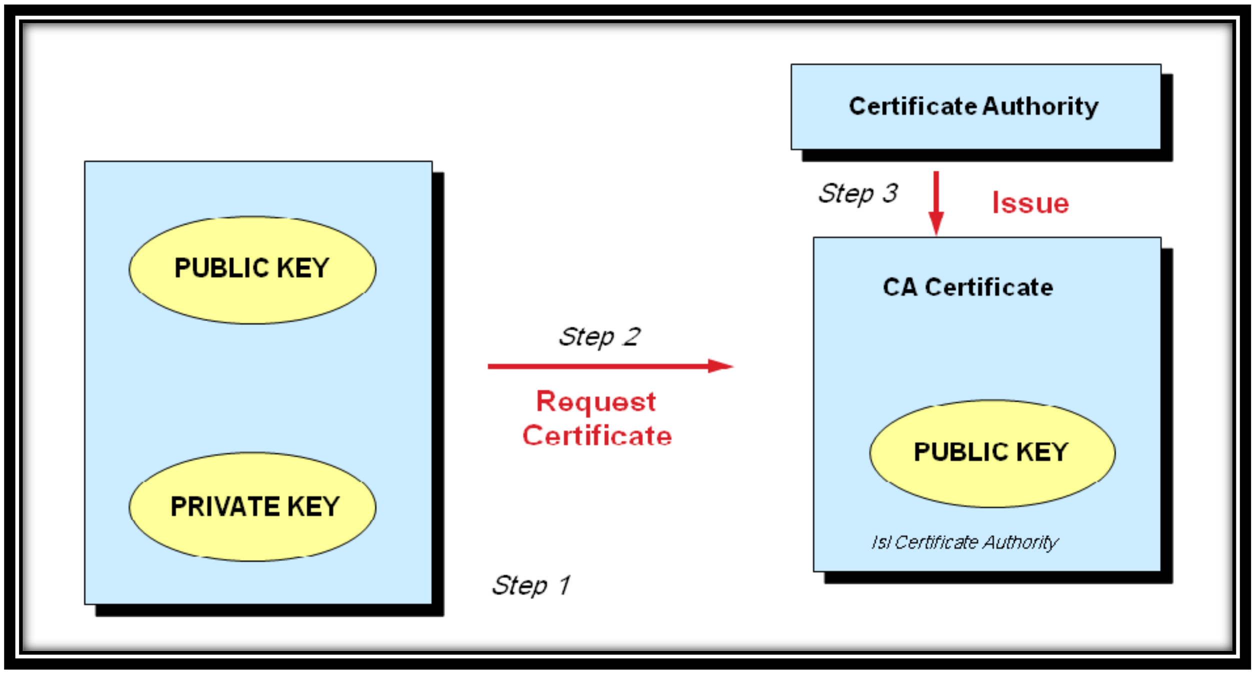


Figure 3.16 RSA Mechanisms in CA

We have CA server contain basic information included public key of any client in network to authorize users of system in this project passwords in authentication has been replaced by public /private key cryptography, client sanded to the other pair in two parts one of them is encrypted and the other is in plain mode, intercepted one is by private key, server decrypt encrypted id then compare to the original one, if they match then user is authorized and if not server well refuse connection.

CHAPTER 4

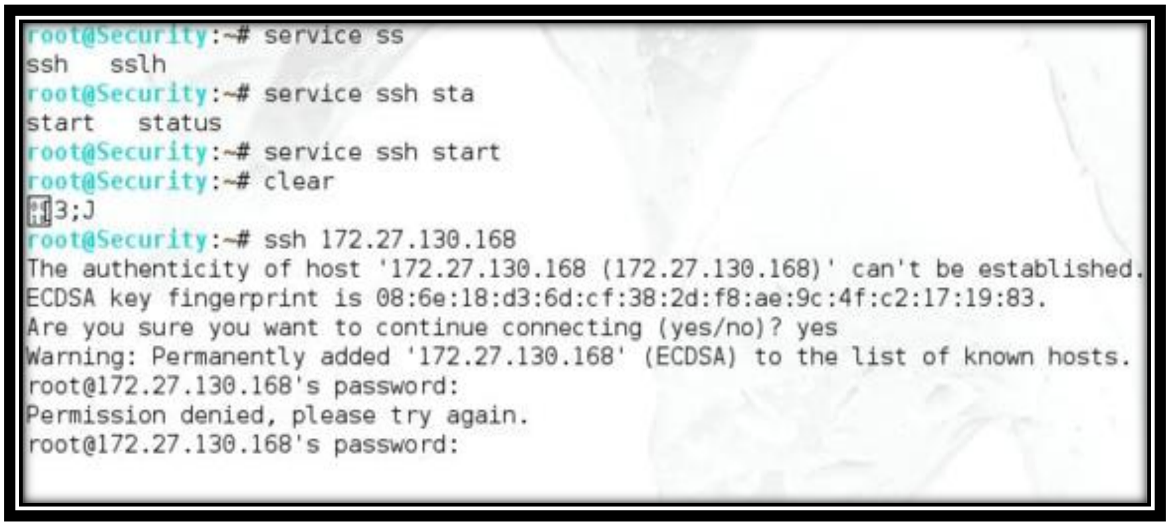
RESULTS AND DISCUSSIONS

4.1 Remote Access Process

Figures 4.1, 4.2 and 4.3 show the attempts of login process to the system before and after the SSH service has been modified using public key instead of passwords.

4.1.1 Remote Access Test

The following figure show how regular remote access fail login to server and explained next to figure:



```
root@Security:~# service ss
ssh   sslh
root@Security:~# service ssh sta
start   status
root@Security:~# service ssh start
root@Security:~# clear
3;J
root@Security:~# ssh 172.27.130.168
The authenticity of host '172.27.130.168 (172.27.130.168)' can't be established.
ECDSA key fingerprint is 08:6e:18:d3:6d:cf:38:2d:f8:ae:9c:4f:c2:17:19:83.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '172.27.130.168' (ECDSA) to the list of known hosts.
root@172.27.130.168's password:
Permission denied, please try again.
root@172.27.130.168's password:
```

Figure 4.1Failed SSH Login

4.1.2. Discussions of Regular SSH Server

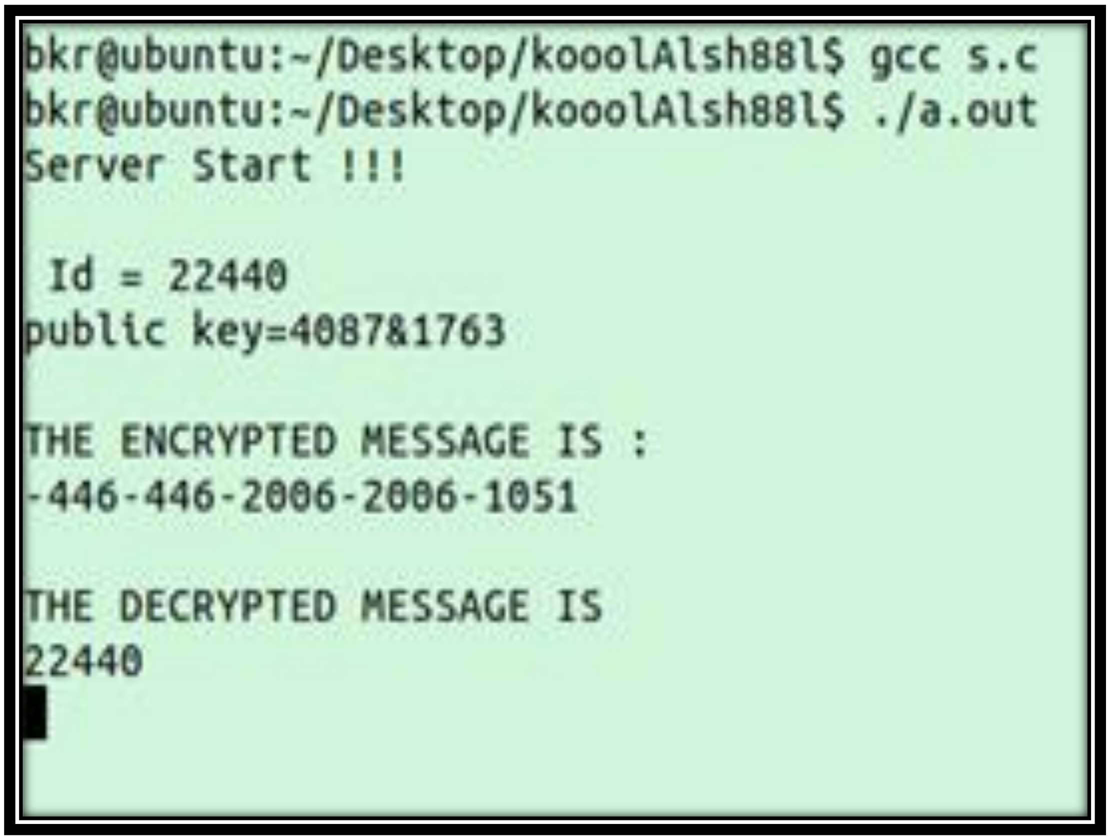
In this section we will discuss difference between regular SSH access failure and modified SSH in this points:

- Figure 4.1 shows how to gain root access before SSH service has been modified. By default SSH service asks for passwords after giving the name of the remote device and its IP address
- Figure 4.2 shows the response in case that user did not enter correct password, as showed in Figure 4.2 SSH service will ask user to try again which give any kind of attackers another chance to try guessing

the password of remote device, these other chances could be up to 3 (or more if configured), that will lead to different kinds of attacks which can break the authentication of clients.

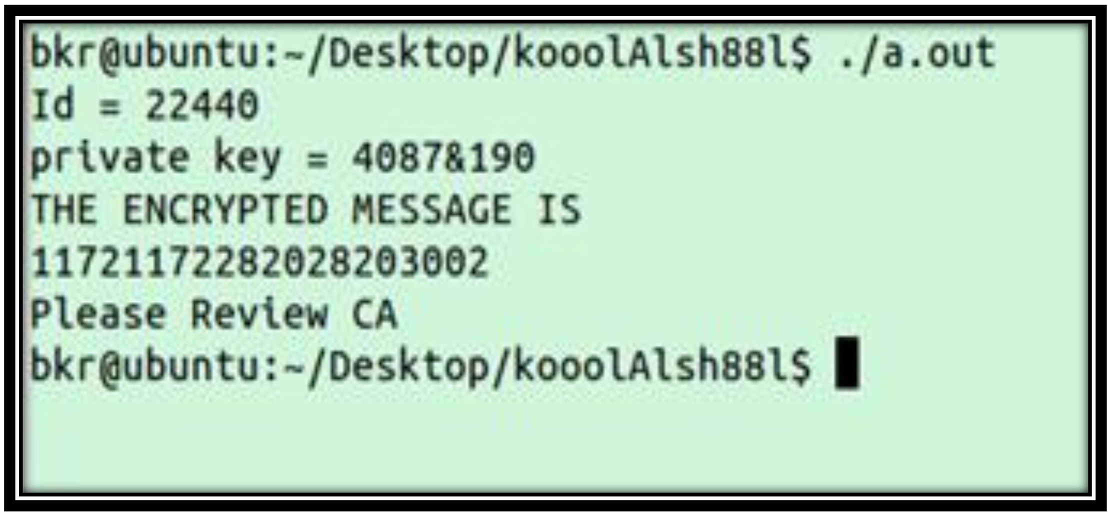
4.2 Modified SSH Remote Access:

The following figure show how modified remote access success login to server and explained next to figure:



```
bkr@ubuntu:~/Desktop/kooolAlsh88l$ gcc s.c
bkr@ubuntu:~/Desktop/kooolAlsh88l$ ./a.out
Server Start !!!

 Id = 22440
public key=4087&1763

THE ENCRYPTED MESSAGE IS :
-446-446-2006-2006-1051

THE DECRYPTED MESSAGE IS
22440
```

Figure 3.10 Login Success Server



```
bkr@ubuntu:~/Desktop/koootAlsh88l$ ./a.out
Id = 22440
private key = 40878190
THE ENCRYPTED MESSAGE IS
117211722820282030O2
Please Review CA
bkr@ubuntu:~/Desktop/koootAlsh88l$ 
```

Figure 3.9 Login Failed Client

4.2.2. Discussions Of Modified SSH Server

In this section we will discuss difference between regular SSH access failure and modified SSH in this following points:

- When a client try to access remote device by modified SSH service as we show in chapter 3, attacker don't have any kind of chances to try guessing pass words because all authorized devices are already registered in CA server
- If any unregistered device try to access the remote device, server will reject the request and ask to return to certificate authority to solve the problem if any.
- Attacker will not return to the CA because he knows that he does not have any authority, so there is that kind of second chances to try another password, only authorized clients have access to remote devices.

CHAPTER 5

CONCLUSION AND RECOMMENDATIONS

5.1 Conclusion

The SSH has been successfully designed and implemented as a remote access solution for the proposed aperture system design. We hope that this project can provided high security features. With new security threats cropping up every day, network managers are understandably protective of their computing assets. Enhanced security measures, however, can inflict significant hardships on legitimate users and can lead to frustration, productivity losses, and dangerous attempts at circumvention of restrictions. Equipping yourself with proper tools for connectivity can make a task easier while still maintaining network security and integrity. Restricted remote access is obtained by replacing SSH password by public key cryptography.

5.2. Recommendations

Security is not finite in particular limit, this project should provide security protection and can be even more enhanced by add these recommendation following:

- Certificate authority have many fields, we recommend adding full CA standard for more functionality.
- Our project is running as a program; add it to kernel and running as a service well make it more useful.
- add more algorithms for encryption to RSA.

References

[1] Yasir Ali, Adding Public Key Security to SSH, DARTMOUTH COLLEGE, Hanover, New Hampshire Feb, 20th, 2003.Accessed on 13/10/2015 at 19:00 pm

[2] FarukDabak, Sameer Panjwani, Tunneling TCP over TCP – A Study Of A Real System, Department of Computer Engineering, Chalmers University of Technology, Gothenburg 2006.Accessed on 13/10/2015 at 19:00 pm

[3] MD. AHASAN HABIB, SSH Over UDP, Department of computer science and engineering, University of Gothenburg, Chalmers University ofTechnology, Sweden, July 2010, Access on 13/10/2015 at 22:00pm

[4] Karen Scarfone, Paul Hoffman, MurugiahSouppaya, Guide to Enterprise Telework and Remote Access Security, National Institute of Standards and Technology-Department of Commerce-United States of America, Jun 2009, Accessed on 01/10/2015 at 19:00 pm

[5] openssh. www.openssh.com. Accessed on 01/10/2015 at 19:00 pm

[6]Wikipedia. www.wikipedia.org. Accessed on 01/10/2015 at 19:00 pm

Appendix

SSH (Secure Shell) Configuration in port and permeation

```
#	$OpenBSD: sshd_config,v 1.73 2005/12/06 22:38:28 reykExp $

# This is the sshd server system-wide configuration file.  See
# sshd_config(5) for more information.

# This sshd was compiled with PATH=/usr/local/bin:/bin:/usr/bin

# The strategy used for options in the default sshd_config shipped with
# OpenSSH is to specify options with their default value where
# possible, but leave them commented.  Uncommented options change a
# default value.

#Port 22
Port 8888
#Protocol 2,1
Protocol 2
#AddressFamily any
#ListenAddress 0.0.0.0
#ListenAddress ::

# HostKey for protocol version 1
#HostKey /etc/ssh/ssh_host_key
# HostKeys for protocol version 2
#HostKey /etc/ssh/ssh_host_rsa_key
#HostKey /etc/ssh/ssh_host_dsa_key

# Lifetime and size of ephemeral version 1 server key
#KeyRegenerationInterval 1h
#ServerKeyBits 768

# Logging
# obsoletes QuietMode and FascistLogging
#SyslogFacility AUTH
SyslogFacility AUTHPRIV
#LogLevel INFO
```

```
# Authentication:

#LoginGraceTime 2m
LoginGraceTime 1m
#PermitRootLogin yes
PermitRootLogin no

#StrictModes yes
#StrictModes yes
#MaxAuthTries 1
MaxAuthTries 1

#RSAAuthentication yes
#PubkeyAuthentication yes
#AuthorizedKeysFile         .ssh/authorized_keys

# For this to work you will also need host keys in /etc/ssh/ssh_known_hosts
#RhostsRSAAuthentication no
RhostsRSAAuthentication no
# similar for protocol version 2
#HostbasedAuthentication no
# Change to yes if you don't trust ~/.ssh/known_hosts for
# RhostsRSAAuthentication and HostbasedAuthentication
#IgnoreUserKnownHosts no
# Don't read the user's ~/.rhosts and ~/.shosts files
#IgnoreRhosts yes

# To disable tunneled clear text passwords, change to no here!
#PasswordAuthentication yes
#PermitEmptyPasswords no
PermitEmptyPasswords no

PasswordAuthentication no

# Change to no to disable s/key passwords
#ChallengeResponseAuthentication yes
ChallengeResponseAuthentication no

# Kerberos options
#KerberosAuthentication no
#KerberosOrLocalPasswd yes
#KerberosTicketCleanup yes
```

```
#KerberosGetAFSToken no

# GSSAPI options
#GSSAPIAuthentication no
GSSAPIAuthentication yes
#GSSAPICleanupCredentials yes
GSSAPICleanupCredentials yes

# Set this to 'yes' to enable PAM authentication, account processing,
# and session processing. If this is enabled, PAM authentication will
# be allowed through the ChallengeResponseAuthentication mechanism.
# Depending on your PAM configuration, this may bypass the setting of
# PasswordAuthentication, PermitEmptyPasswords, and
# "PermitRootLogin without-password". If you just want the PAM account and
# session checks to run without PAM authentication, then enable this but set
# ChallengeResponseAuthentication=no
#UsePAM no
UsePAM yes

# Accept locale-related environment variables
AcceptEnv LANG LC_CTYPE LC_NUMERIC LC_TIME LC_COLLATE
LC_MONETARY LC_MESSAGES
AcceptEnv LC_PAPER LC_NAME LC_ADDRESS LC_TELEPHONE
LC_MEASUREMENT
AcceptEnv LC_IDENTIFICATION LC_ALL
#AllowTcpForwarding yes
AllowUsersremoteaccess
#GatewayPorts no
#X11Forwarding no
X11Forwarding yes
#X11DisplayOffset 10
#X11UseLocalhost yes
#PrintMotd yes
PrintMotd yes
#PrintLastLog yes
#TCPKeepAlive yes
#UseLogin no
#UsePrivilegeSeparation yes
UsePrivilegeSeparation yes
#PermitUserEnvironment no
PermitUserEnvironment no
#Compression delayed
#ClientAliveInterval 0
```

```
#ClientAliveCountMax 3
#ShowPatchLevel no
#UseDNS yes
#PidFile /var/run/sshd.pid
#MaxStartups 10
#PermitTunnel no
#ChrootDirectory none

# no default banner path
#Banner /some/path

# override default of no subsystems
Subsystem     sftp     /usr/libexec/openssh/sftp-server
```