CS35L – Spring 2019

Slide set:	8.1
Slide topics:	SSH
Assignment:	8

Communication Over the Internet

What type of guarantees do we want?

- **Confidentiality**
  - Message secrecy
- **Data integrity**
  - Message consistency
- **Authentication**
  - Identity confirmation
- **Authorization**
  - Specifying access rights to resources

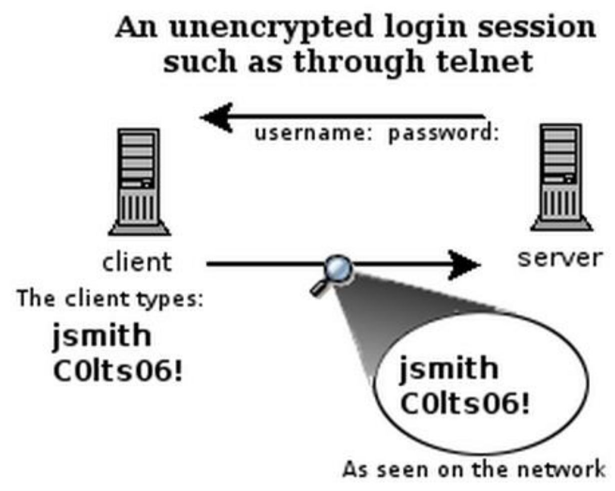
What is SSH?

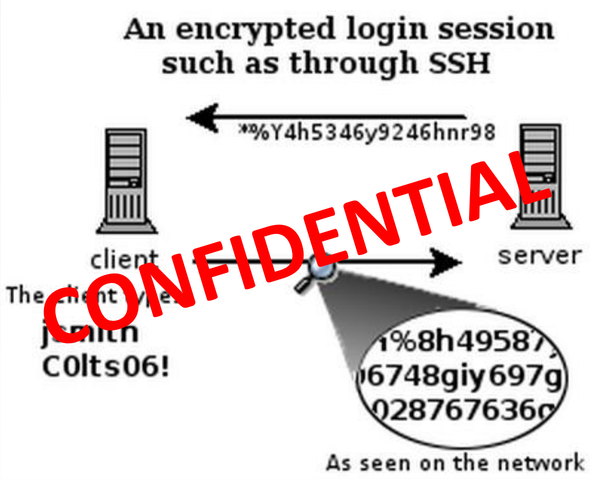
Secure **Sh**ell

Used to remotely access shell

Successor of telnet

Encrypted and better authenticated session





Encryption Types

Symmetric Key Encryption

- a.k.a shared/secret key
- Key used to encrypt is the same as key used to decrypt

Asymmetric Key Encryption: Public/Private

- 2 different (but related) keys: public and private
  - Only creator knows the relation. Private key cannot be derived from public key
- Data encrypted with public key can only be decrypted by private key and vice versa
- Public key can be seen by anyone
- **Never** publish private key!!!

Symmetric-key Encryption

- Same secret key used for encryption and decryption
- **Example** : Data Encryption Standard (**DES**)
- **Caesar's cipher**
  - Map the alphabet to a shifted version
    - ABCDEFGHIJKLMNOPQRSTUVWXYZ
    - DEFGHIJKLMNOPQRSTUVWXYZABC
  - Plaintext – SECRET. Ciphertext – VHFUHW
  - Key is 3 (number of shifts of the alphabet)

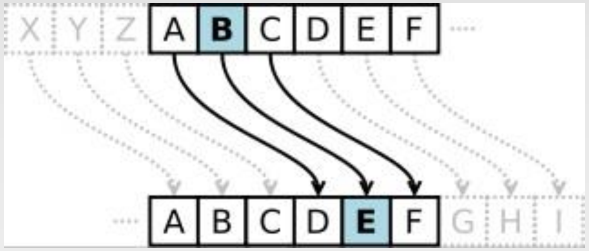


Image Source: wikipedia

- **Key distribution** is a problem
  - The secret key has to be delivered in a safe way to the recipient
  - Chance of key being compromised

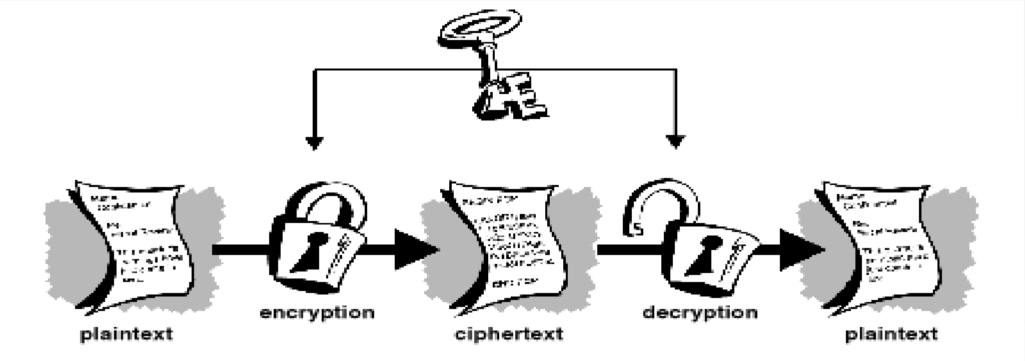


Image Source: gpgtools.org

Public-key Encryption (Asymmetric)

- Uses a pair of keys for encryption
  - **Public key** – Published and known to everyone
  - **Private key** – Secret key known only to the owner
- **Encryption**
  - Use public key to encrypt messages
  - Anyone can encrypt message, but they cannot decrypt the ciphertext
- **Decryption**
  - Use private key to decrypt messages
- **Example** : **RSA** – Rivest, Shamir & Adleman
  - Property used - **Difficulty of factoring** large integers to prime numbers
  - N = p * q (3233 = 61 * 53)
  - N is a large integer and p, q are prime numbers
  - N is part of the public key
  - http://en.wikipedia.org/wiki/RSA_Factoring_Challenge

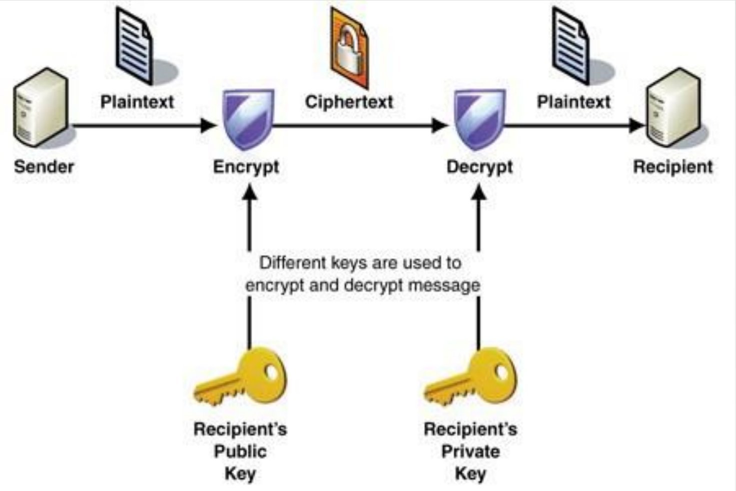


Image Source: MSDN

High-Level SSH Protocol

Client ssh’s to remote server

- `$ ssh username@somehost`
- If first time talking to server -> host validation

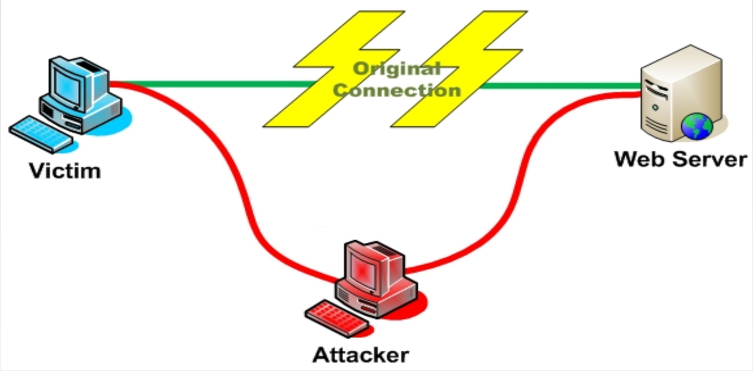
```
The authenticity of host 'somehost (192.168.1.1)' can't be established.
RSA key fingerprint is 90:9c:46:ab:03:1d:30:2c:5c:87:c5:c7:d9:13:5d:75.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'somehost' (RSA) to the list of known hosts.
```

- ssh doesn't know about this host yet
- shows hostname, IP address and fingerprint of the server’s public key, so you can be sure you're talking to the correct computer
- After accepting, public key is saved in ~/.ssh/known_hosts

Host Validation

Next time client connects to server

- Check host’s public key against saved public key
- If they don’t match



```
$ ssh username@somehost
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
@    WARNING: REMOTE HOST IDENTIFICATION HAS CHANGED!     @
@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
IT IS POSSIBLE THAT SOMEONE IS DOING SOMETHING NASTY!
Someone could be eavesdropping on you right now (man-in-the-middle attack)!
It is also possible that the RSA host key has just been changed.
The fingerprint for the RSA key sent by the remote host is
90:9c:46:ab:03:1d:30:2c:5c:87:c5:c7:d9:13:5d:75.
Please contact your system administrator.
Add correct host key in /home/user/.ssh/known_hosts to get rid of this message.
Offending key in /home/user/.ssh/known_hosts:1
Password authentication is disabled to avoid man-in-the-middle attacks.
Agent forwarding is disabled to avoid man-in-the-middle attacks.
X11 forwarding is disabled to avoid man-in-the-middle attacks.
Permission denied (publickey,password,keyboard-interactive).
```

Host Validation (cont’d)

Client asks server to prove that it is the owner of the public key using **asymmetric encryption**

- Encrypt a message with public key
- If server is true owner, it can decrypt the message with private key

AUTHENTICATION

If everything works, host is successfully validated

Session Encryption

Client and server agree on a **symmetric encryption key** (session key)

All messages sent between client and server

- encrypted at the sender with session key
- decrypted at the receiver with session key

anybody who doesn't know the session key (hopefully, no one but client and server) doesn't know any of the contents of those messages

User Authentication

Password-based authentication

- Prompt for password on remote server
- If username specified exists and remote password for it is correct then the system lets you in

Key-based authentication

- Generate a key pair on the client
- Copy the public key to the server (~/.ssh/authorized_keys)
- Server authenticates client if it can demonstrate that it has the private key
- The private key can be protected with a passphrase
- Every time you ssh to a host, you will be asked for the passphrase (inconvenient!)

ssh-agent (passphrase-less ssh)

A program used with OpenSSH that provides a secure way of storing the private key

`ssh-add` prompts user for the passphrase once and adds it to the list maintained by `ssh-agent`

Once passphrase is added to `ssh-agent`, the user will not be prompted for it again when using SSH

OpenSSH will talk to the local ssh-agent daemon and retrieve the private key from it automatically

X Window System

Windowing system that forms the basis for most GUIs on UNIX

X is a network-based system. It is based upon a network protocol such that a program can run on one computer but be displayed on another (X Session Forwarding)

Lab 8

Securely log in to each others' computers

- Use ssh (OpenSSH)

Use key-based authentication

- Generate key pairs

Make logins convenient

- type your passphrase once and be able to use ssh to connect to any other host without typing any passwords or passphrases

Use port forwarding to run a command on a remote host that displays on your host

Lab Environment Setup

Debian

- Make sure you have openssh-server and openssh-client installed
- `$dpkg --get-selections | grep openssh` should output:
    - openssh-server   install
    - openssh-client install
- If not:
    - `$sudo apt-get install openssh-server`
    - `$sudo apt-get install openssh-client`

Server Steps

- **Generate public and private keys**
  - $ `ssh-keygen` (by default saved to ~/.ssh/is_rsa and id_rsa.pub) – don’t change the default location
- **Create an account for the client on the server**
  - $ `sudo useradd -d /home/<homedir_name> -m <username>`
  - $ `sudo passwd <username>`
- **Create .ssh directory for new user**
  - $ `cd /home/<homedir_name>`
  - $ `sudo mkdir .ssh`
- **Change ownership and permission on .ssh directory**
  - $ `sudo chown -R username .ssh`
  - $ `sudo chmod 700 .ssh`

Client Steps

- **Generate public and private keys**
  - $ `ssh-keygen`
- **Copy your public key to the server for key-based authentication (~/.ssh/authorized_keys)**
  - $ `ssh-copy-id -i UserName@server_ip_addr`
- **Add private key to authentication agent (ssh-agent)**
  - $ `ssh-add`
- **SSH to server**
  - $ `ssh UserName@server_ip_addr`
  - $ `ssh -X UserName@server_ip_addr` (X11 session forwarding)
- **Run a command on the remote host**
  - $ `xterm`, $ `gedit`, $ `firefox`, etc.

How to Check IP Addresses

`$ ifconfig`

- configure or display the current network interface configuration information (IP address, etc.)

`$ hostname -I`

- gives the IP address of your machine directly

`$ ping <ip_addr>`(**p**acket **in**ternet **g**roper)

- Test the reachability of a host on an IP network
- measure round-trip time for messages sent from a source to a destination computer
- Example: $ ping 192.168.0.1, $ ping google.com