SSH – Secure Shell

Week 6

Cryptography

- **Plaintext** – Actual message
- **Ciphertext** – Encrypted message (unreadable gibberish)
- **Encryption** – Going from plaintext to ciphertext
- **Decryption** – Going from ciphertext to plaintext
- **Secret key**
  - Part of the mathematical function used to encrypt/decrypt.
  - Good key makes it hard to get back plaintext from ciphertext

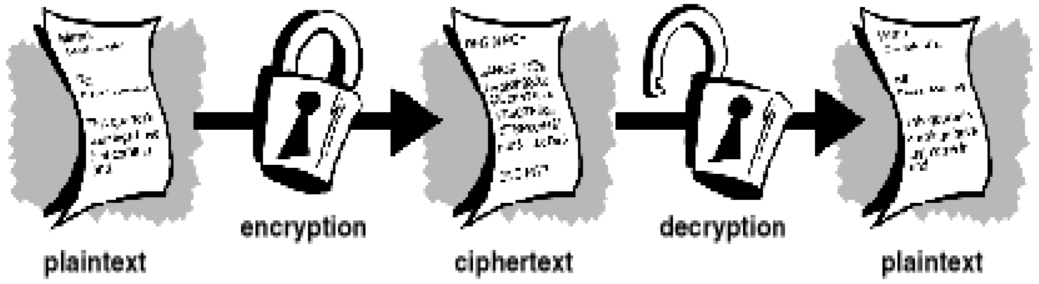


Image Source: gpgtools.org

Symmetric-key Encrption

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

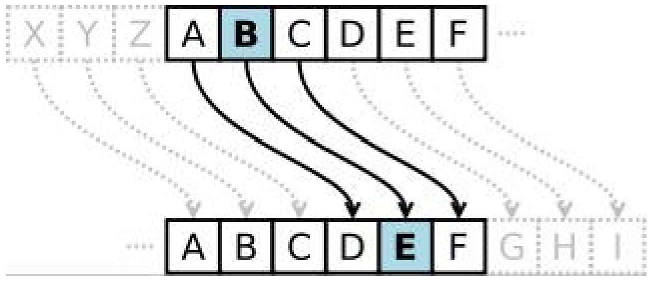


Image Source: wikipedia

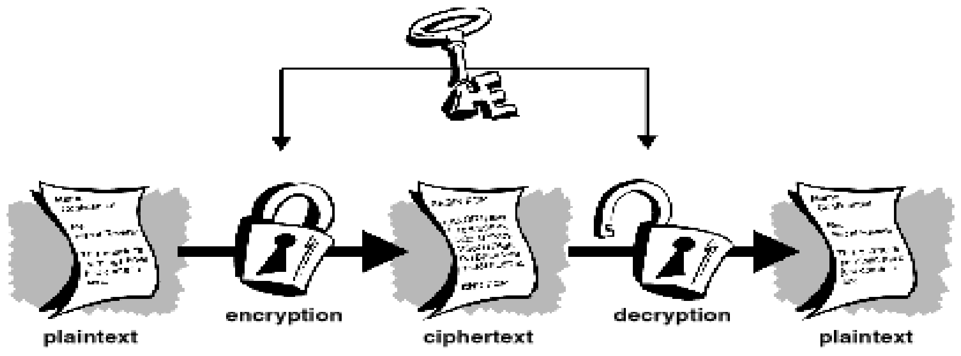


Image Source: gpgtools.org

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

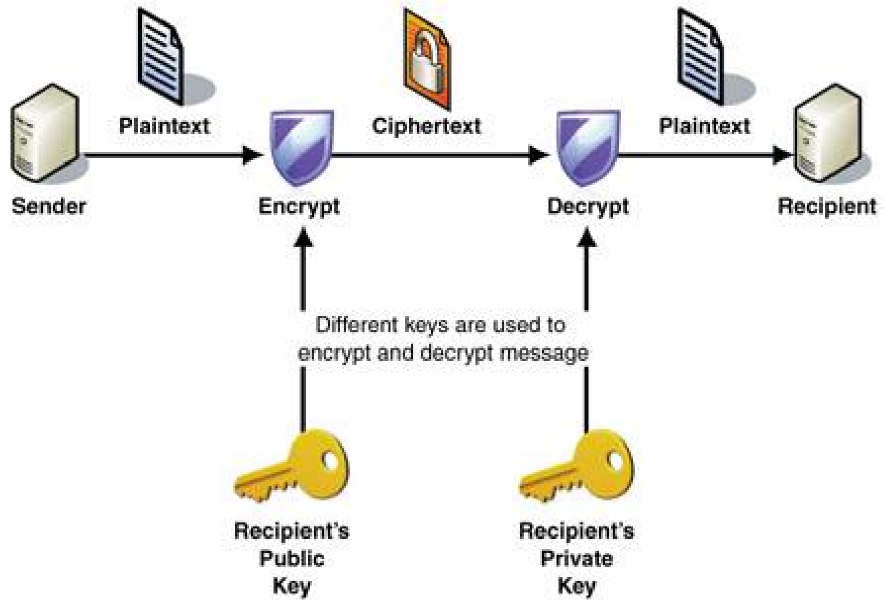
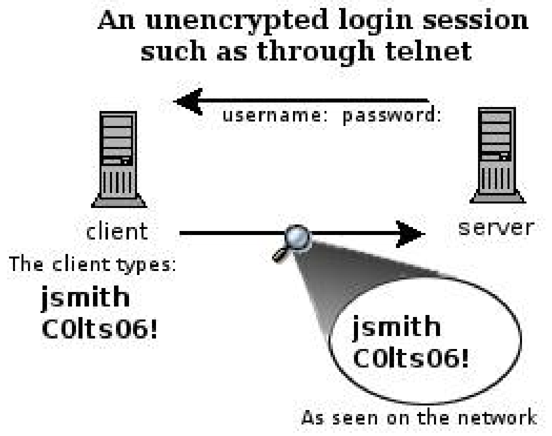


Image Source: MSDN

Secure Shell (SSH)

- Telnet
  - Remote access
  - Not encrypted
  - Packet sniffers can intercept sensitive information (username/password)
- SSH
  - Run processes remotely
  - Encrypted session
  - **Session key** (secret key) used for encryption during the session



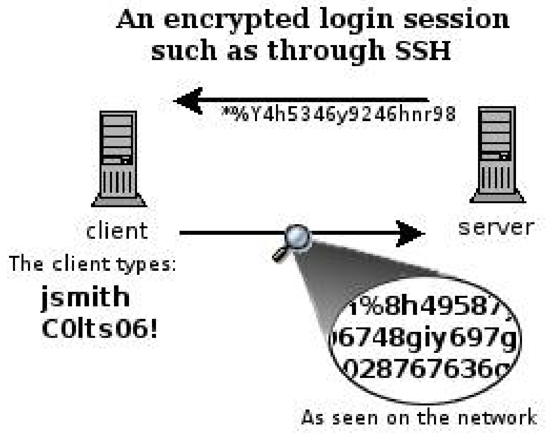


Image Source: suso.com

Secure Shell (SSH) – Client Authentication

- **Password** login
  - `ssh username@ugrad.seas.ucla.edu`
- **Passwordless** login with keys
  - Use private/public keys for authentication **(Server and Client authentication)**
  - `ssh-keygen`
    - Passphrase (longer version of a password / more secure)
    - Passphrase for protecting the private key
    - Passphrase needed whenever the keys are accessed
  - `ssh-copy-id username@ugrad.seas.ucla.edu`
    - Copies the public key to the server (`~/.ssh/authorized_keys`)
  - Login without password
    - `ssh username@ugrad.seas.ucla.edu`
    - Run scripts/commands on the remote machine
      - `ssh username@ugrad.seas.ucla.edu ls`
    - But you need to provide the passphrase to use the private key

Secure Shell (SSH) – Client Authentication

- **Passphrase-less** authentication
  - **`ssh-agent -`** Authentication agent
  - Manages private key identities for SSH
  - To avoid entering the passphrase whenever the key is used
  - **`ssh-add`**
    - Registers the private key with the agent
    - Passphrase asked only once
    - `ssh` will ask the `ssh-agent` whenever the private keys are needed

Encryption Schemes for SSH

- **Session encryption**
  - Symmetric encryption
  - Exchange secret key (Example – Diffie-Hellman)
- **Host/Client Validation**
  - Public-key Encryption
  - Challenge-Response
    - Host sends a “challenge” that has to be answered by the client.
    - Similarly, client sends a “challenge” that has to be answered by the host.

Account Administration

- Install OpenSSH (Should be done on both server and client)
  - `$ sudo apt-get update`
  - `$ sudo apt-get install openssh-server`
  - `$ sudo apt-get install openssh-client`
- Server
  - `$ sudo useradd -d /home/<username> -m <UserName>`
  - `$ sudo passwd <username>`
  - `$ cd /home/<username>`
  - `$ sudo mkdir .ssh`
  - `$ sudo chown -R <username> .ssh`
  - `$ sudo chmod 700 .ssh`
  - `$ ifconfig` (This will give you the IP address of the server. Give this to your partner.)
  - `$ ps aux | grep ssh` (This should show a process named 'sshd' – the ssh daemon/server)
- Client
  - Password login
    - `$ ping server_ip_addr (Just to check if the server responds)`
    - `$ ssh <username>@server_ip_addr`
  - Password-less login
    - `$ ssh-keygen`
    - `$ ssh-copy-id -i <username>@server_ip_addr`
    - `$ ssh <username>@server_ip_addr` [Should not ask for login password]
  - Passphrase-less login
    - `$ ssh-add`
    - `$ ssh <username>@server_ip_addr` [Should not ask for key's passphrase]
  - X Session forwarding – Running programs with GUI
    - `$ ssh -X <UserName>@server_ip_addr`
    - `$ xterm`
    - `$ firefox`

X Session forwarding

- X is the windowing system for GUI apps on Linux
- You want to run such apps remotely, but the GUI should show up on the local machine
  - `ssh -X username@ugrad.seas.ucla.edu`
  - `gedit`
  - `gimp`

Secure copy (scp)

- Based on Secure Shell (ssh)
- Used for transferring files between hosts in a secure way (encrypted)
- Usage similar to `cp`
  - `scp [source] [destination]`
- Transferring to remote host
  - `scp /home/username/doc.txt username@ugrad.seas.ucla.edu:/home/user/docs/`
- Transferring from remote host
  - `scp username@ugrad.seas.ucla.edu:/home/user/docs/foo.txt /home/username`

Digital Signature

- Protect **integrity** of the documents
  - Receiver received the document that the sender intended
- Digital signature is extra data attached to the document that can be used to check **tampering**
- **Message digest**
  - **Shorter** version of the document
  - Generated using **hashing** algorithms
  - Even a slight change in the original document will change the message digest with **high probability**

Digital Signature

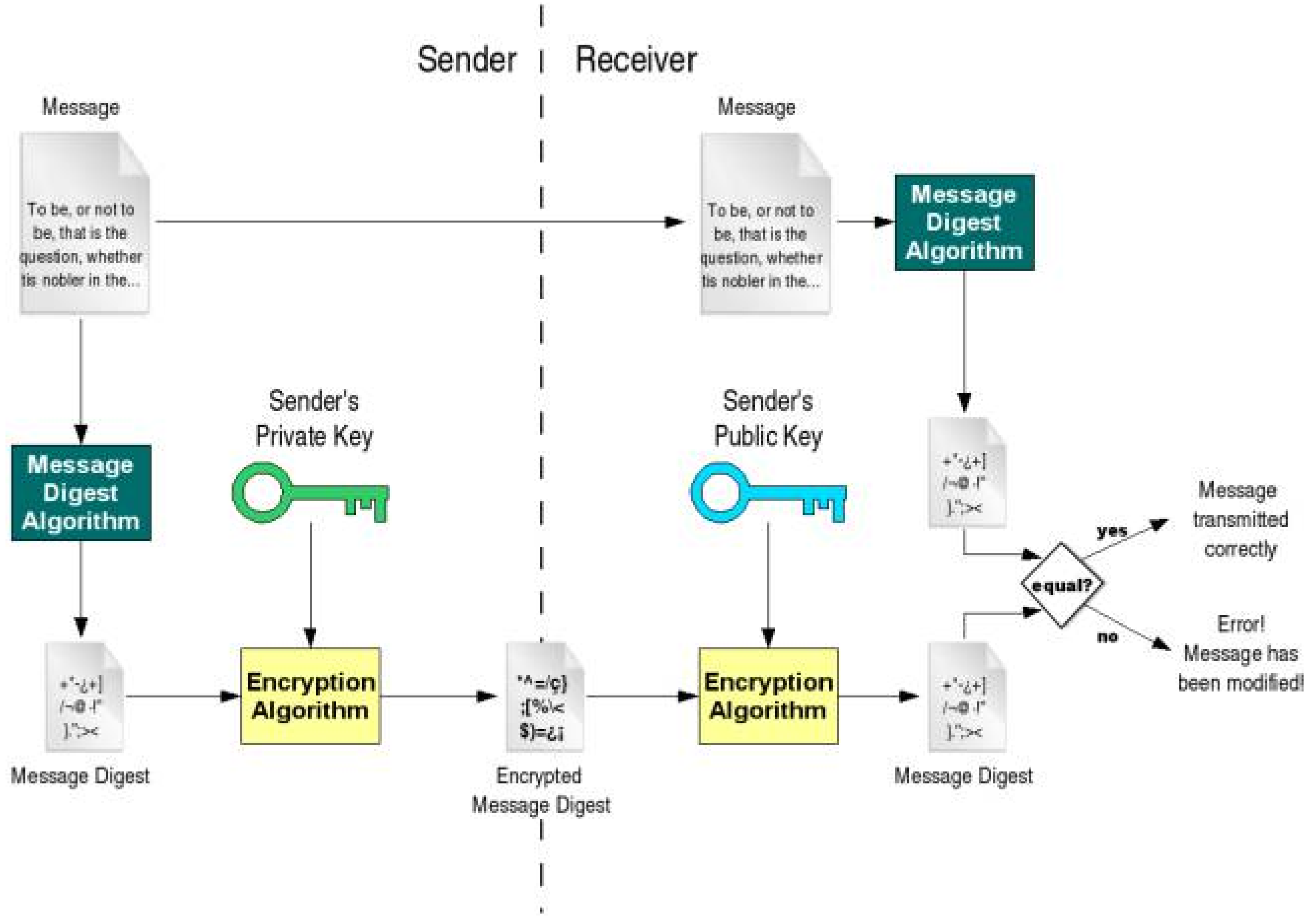


Image Source : gdp.globus.org

- Verifies document integrity
- Does it prove origin?
- Who is Certificate Authority (CA) ?

Certificate Authority

- A Certificate authority (CA) is a trusted third party (Verisign ,Symantec, etc)
- Issues digital certificates (online identity of persons, companies) that map public keys to an entity
- Browsers have a list of trusted CAs that can be referred to, whenever the user needs to know the owner of a public key (verify a Digital Certificate)
- More details of the verification process at - https://sites.google.com/site/ddmwsst/digital-certificates

GNU Privacy Guard

- gpg [option]

--gen-key	(Generating new keys)
--armor	(ASCII format)
--export	(Exporting public key)
--import	(Import public key)
--detach-sign	(Creates a file with just the signature)
--verify	(Verify signature with a public key)
--encrypt	(Encrypt document)
--decrypt	(Decrypt document)
--list-keys	(List all keys in the keyring)
--send-keys	(Register key with a public server / –keyserver option)
--search-keys	(Search for a someone's key)