CS 35L- Software Construction Laboratory

Fall 18

TA: Guangyu Zhou

Digital Signature

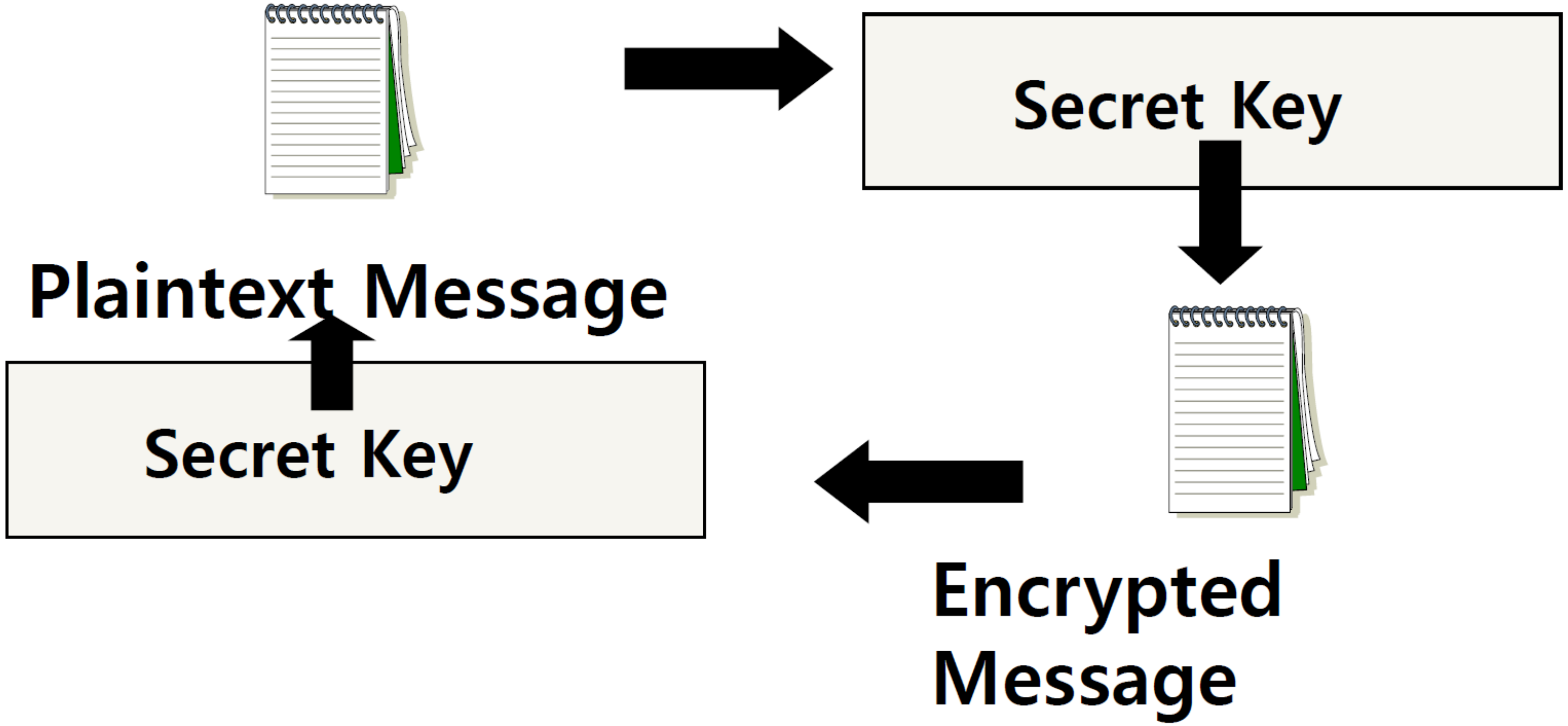
Week 9

Outline

- Review of Cryptography
- Digital Signature
- Hints for Assignment 8

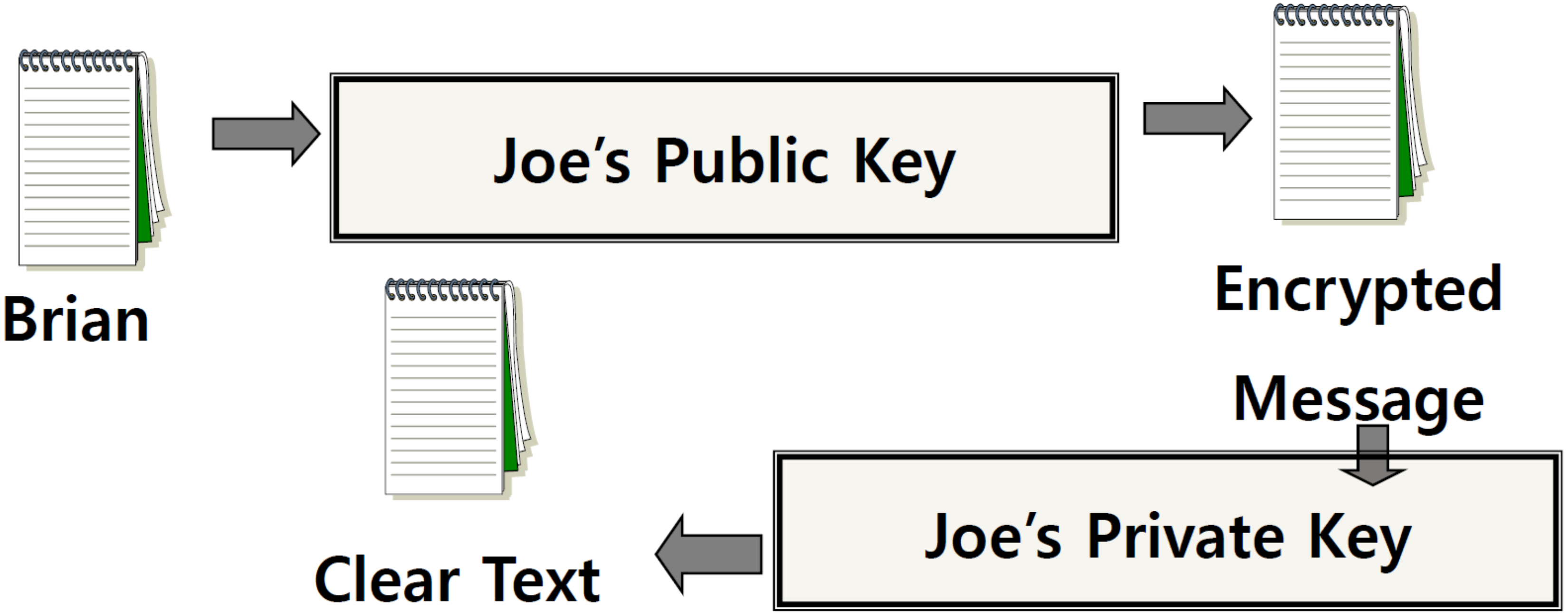
Review: Secret Key (symmetric) Cryptography

- A single key is used to both encrypt and decrypt a message



Review: Public Key (asymmetric) Cryptography

- Two keys are used: a public and a private key. If a message is encrypted with one key, it has to be decrypted with the other.



Review: Encryption Types Comparison

- 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!!!

Review: 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

- 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 and retrieve the private key from it automatically

X session forwarding

- **X** is the windowing system for GUI apps on Linux
- **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
  - i.e. you want to run such apps remotely, but the GUI should show up on the local machine
- Windowing system forms the basis for most GUIs on Unix
  - **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

  => An electronic stamp or seal, almost exactly like a written signature, except more guarantees!
- Digital signature is extra data attached to the document (or separately) 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**

Steps for Generating a Digital Signature

SENDER:

1) Generate a *Message Digest*
   - The message digest is generated using a set of hashing algorithms
   - A message digest is a 'summary' of the message we are going to transmit
   - Even the slightest change in the message produces a different digest
2) Create a Digital Signature
   - The message digest is encrypted using the sender's *private* key. The resulting encrypted message digest is the *digital signature*
3) Attach digital signature to message and send to receiver

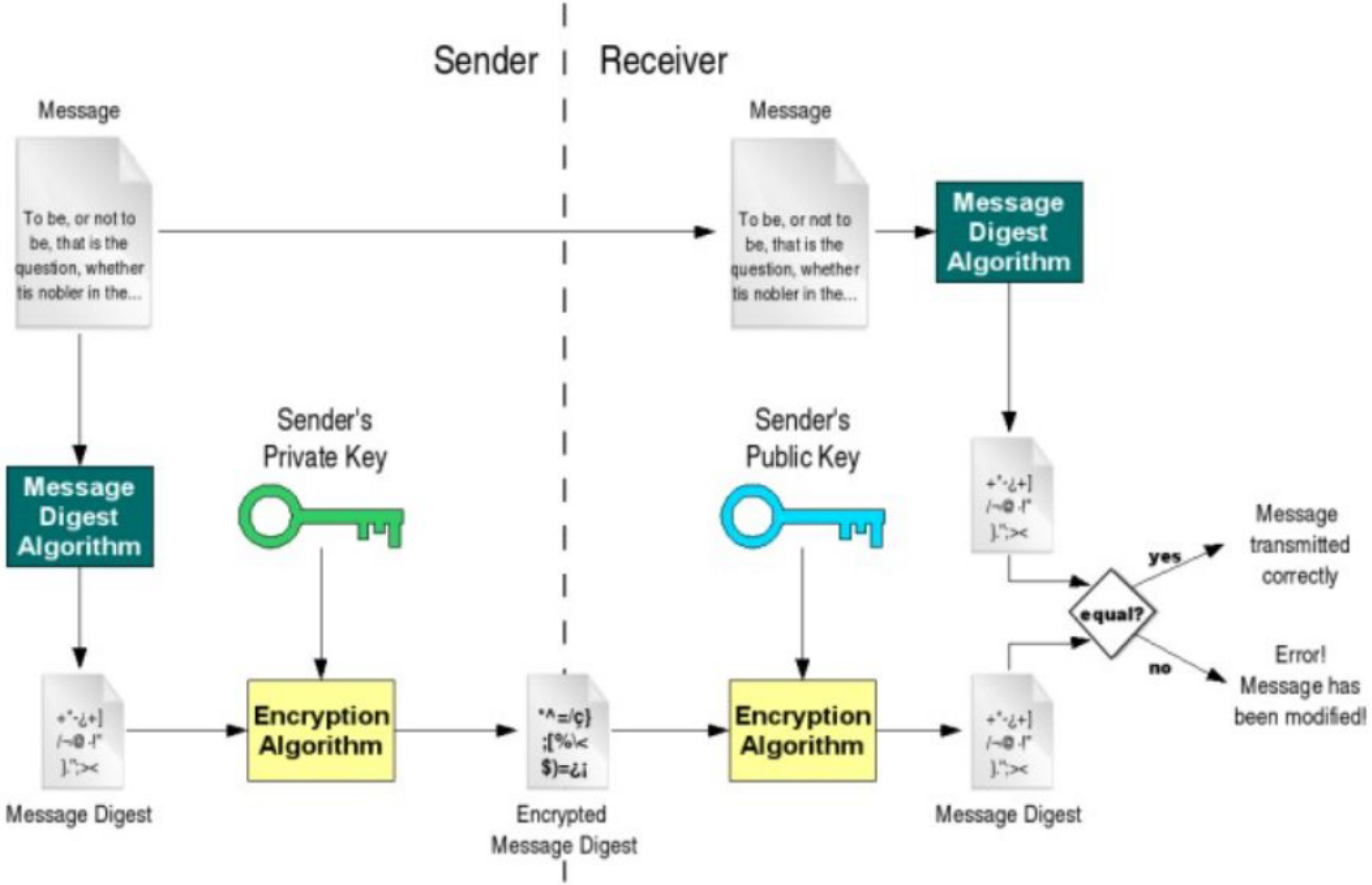
Steps for Generating a Digital Signature

RECEIVER:

1) Recover the *Message Digest*
   - Decrypt the digital signature using the sender’s public key to obtain the message digest generated by the sender
2) Generate the Message Digest
   - Use the same message digest algorithm used by the sender to generate a message digest of the received message
3) Compare digests (the one sent by the sender as a digital signature, and the one generated by the receiver)
   - If they are not *exactly the same* => the message has been tampered with by a third party
   - We can be sure that the digital signature was sent by the sender (and not by a malicious user) because *only* the sender's public key can decrypt the digital signature and that public key is proven to be the sender’s through the certificate. If decrypting using the public key renders a faulty message digest, this means that either the message or the message digest are not exactly what the sender sent.

Digital signature

Verifies document integrity, but does it prove origin? and who is the Certificate Authority?



What is *GNU privacy guard*

- GnuPG allows you to encrypt and sign your data and communications
- It features a versatile key management system, along with access modules for all kinds of public key directories.
- GnuPG, also known as *GPG*, is a command line tool with features for easy integration with other applications.
- Reference: https://gnupg.org/gph/en/manual.html#INTRO

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 someone’s key

Assignment 8 is available

- Visit:
  http://web.cs.ucla.edu/classes/fall18/cs35L/assign/assign7.html
  - Deadline: 11:55 PM, 12-01, Saturday.
  - Form a team of 2 (can be in another lab)
    - Report your and your team member's UID to log.txt
- BeagleBone setup instructions:
  - **https://piazza.com/class/jmgnuany1cl6gw?cid=288**
- **New submission requirement:**
  - A file **eeprom** that is a copy of the file /sys/bus/i2c/devices/0-0050/eeprom on your BeagleBone.
- Follow these instructions to **reset** a used board:
  - http://wiki.seeedstudio.com/BeagleBone_Green/#update-to-latest-software

On your PC:

- Make sure X11 forwarding is enabled:
  - Putty: Connection -> SSH -> X11 -> “Enable X11 forwarding” should be checked
  - SSH command (Mac/Linux): -X or -Y flag
- Make sure an X11 windowing tool is installed:
  - Windows: Xming
  - Mac: XQuartz
  - (U|Li)nix: No extra software necessary!

Lab Environment Setup

- On your board:
  - Make sure you have openssh-server and openssh-client installed
  - Check: *$ 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 (IP address, etc.)
- *$ 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