CS 35L

Week 6

TA: Tomer Weiss

Feb-11-2016

goo.gl/HR4YTq

Slides

Announcements

- Student presentations today:
  - Machine Translation
  - To Make AI More Human, Teach It to Chitchat

web.cs.ucla.edu/classes/winter16/cs35L/assign/assign10.html

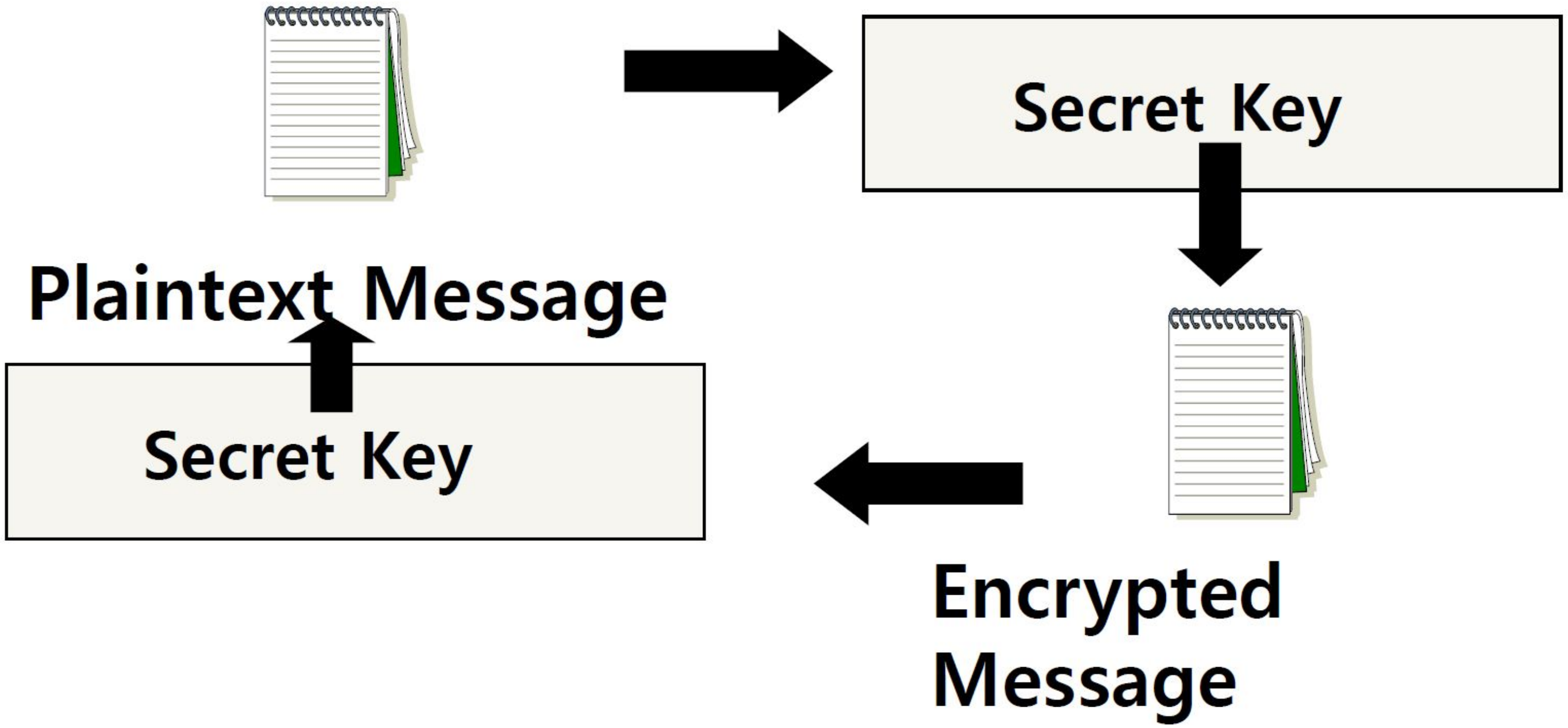
- Next week:
  - Write your topic here
  - Not registering you topic beforehand may result in rescheduling of your presentation
  - For reference on presentation, grading, please refer to this rubric.

SSH - Secure Shell

Week 6

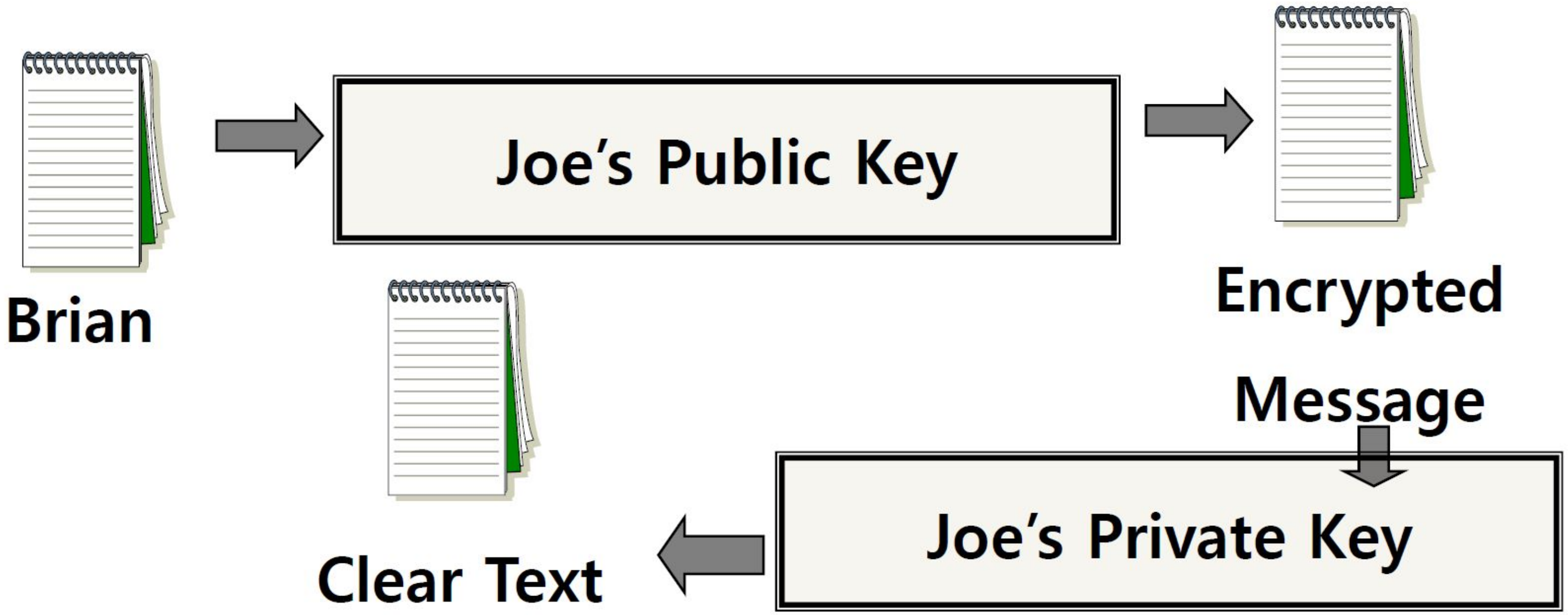
Reminder: Secret Key (symmetric) Cryptography

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



Reminder: 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.



Homework

Digital signature

- An electronic stamp\seal
- Digital signature is extra data attached to the document
  - Can be used to check **tampering**
  - Ensures **integrity** of the documents
  - Receiver received the document that the sender intended
- 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

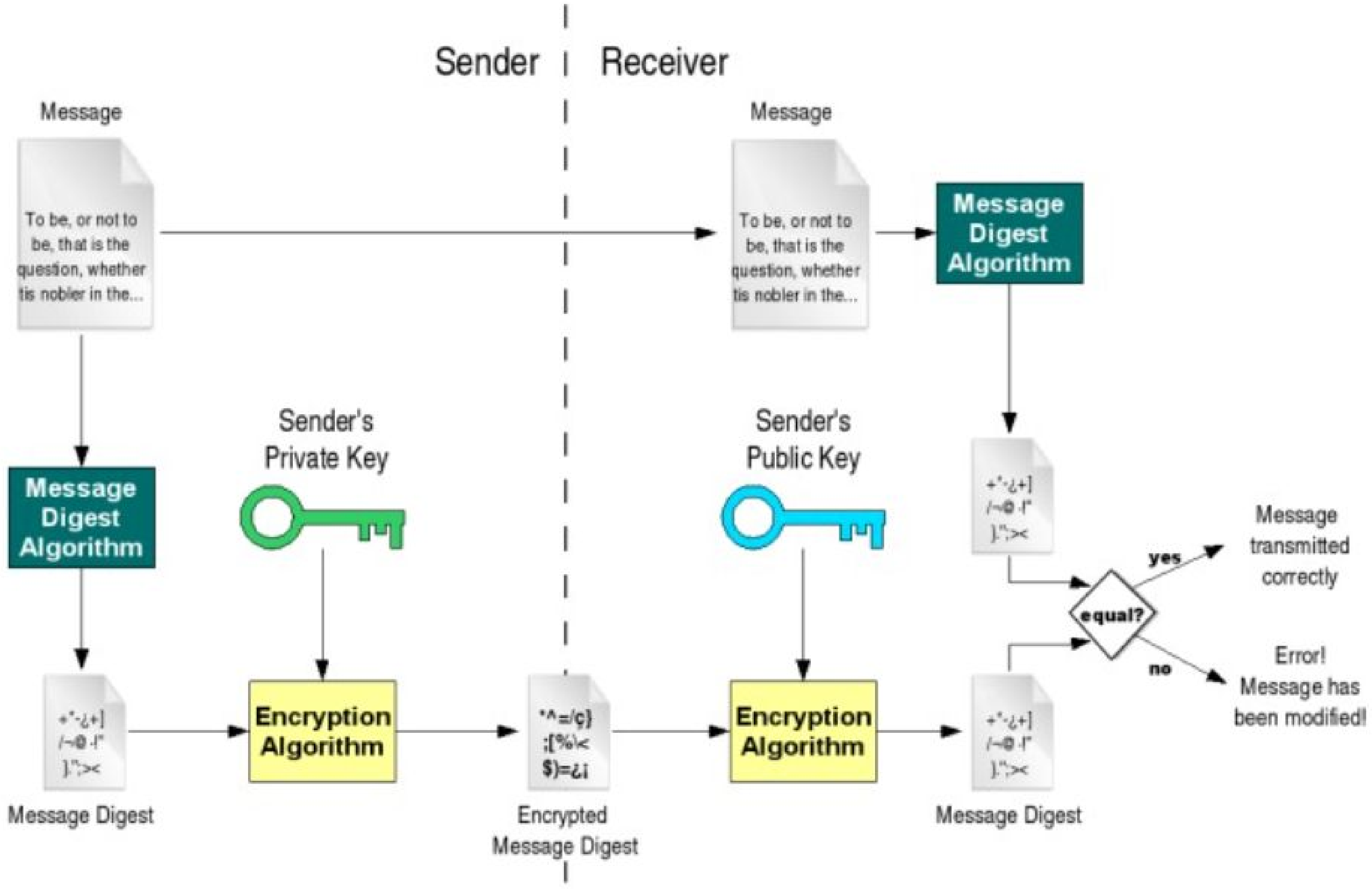
Steps for Generating a Digital Signature

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?



> gpg [option]

GNU privacy guard

--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

Homework 6

- Answer 2 questions in the file **`hw.txt`**
- Generate a key pair with the GNU Privacy Guard's commands
  - $ `gpg --gen-key` (choose default options)
- Export public key, in ASCII format, into **`hw-pubkey.asc`**
  - $ `gpg --armor --output hw-pubkey.asc --export 'Your Name'`
- Make a tarball of the above files + **`log.txt`** and zip it with gzip to produce `hw.tar.gz`
  - `$ tar –cf hw.tar <files>`
  - `$ gzip hw.tar` -> creates hw.tar.gz
- Use the private key you created to make a detached clear signature `hw.tar.gz.sig` for `hw.tar.gz`
  - `$ gpg --armor --output hw.tar.gz.sig --detach-sign hw.tar.gz`
- Use given commands to verify signature and file formatting
  - These can be found at the end of the assignment spec

Lab 6

- **Securely login 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

- Ubuntu
  - 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`

Client 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

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`
- **Optional: disable password-based authentication**
  - `$ emcas /etc/ssh/sshd_config`
  - change PasswordAuthentication option to no

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.)
- $ `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

Lab

web.cs.ucla.edu/classes/winter16/cs35L/assign/assign6.html