CS 35L

Software Construction Laboratory

Lecture 9.1

5th March, 2019

Logistics

- Hardware requirement for Week 8
  - Seeed Studio BeagleBone Green Wireless Development Board
- Presentations for Assignment 10
  - https://docs.google.com/spreadsheets/d/1o6r6CKCaB2du3klPflHiquymhBvbn7oP0wkHHMz_q1E/edit?usp=sharing
- Assignment 7 is due on 3rd March, 2018 at 11:55pm

Review - Previous Lab

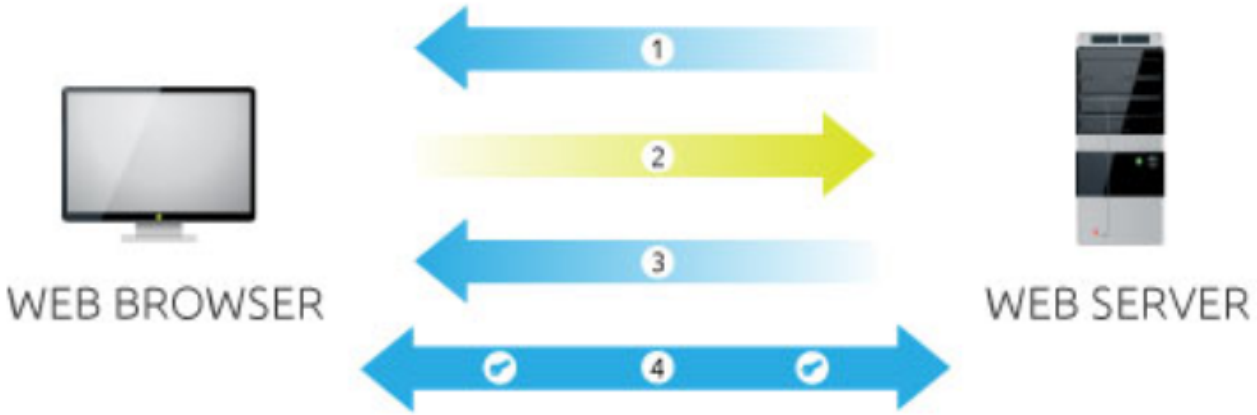
- SSH
  - Symmetric Key Encryption
  - Asymmetric Key Encryption
  - Server Validation
  - User Authentication

SSH

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

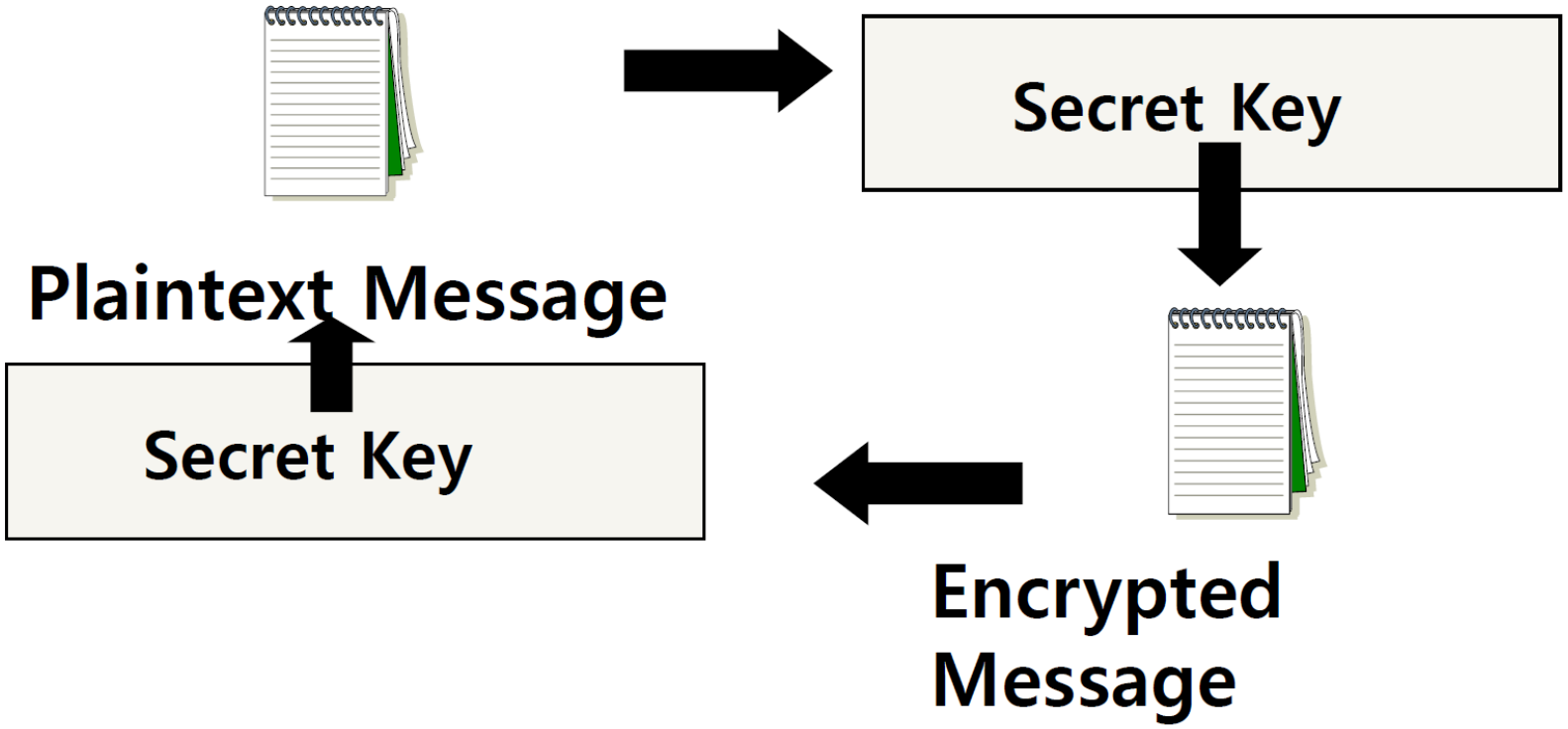
Session Encryption



1. **Server** sends a copy of its asymmetric public key.
2. **Browser** creates a symmetric session key and encrypts it with the server's asymmetric public key. Then sends it to the server.
3. **Server** decrypts the encrypted session key using its asymmetric private key to get the symmetric session key.
4. **Server** and **Browser** now encrypt and decrypt all transmitted data with the symmetric session key. This allows for a secure channel because only the browser and the server know the symmetric session key, and the session key is only used for that session. If the browser was to connect to the same server the next day, a new session key would be created.

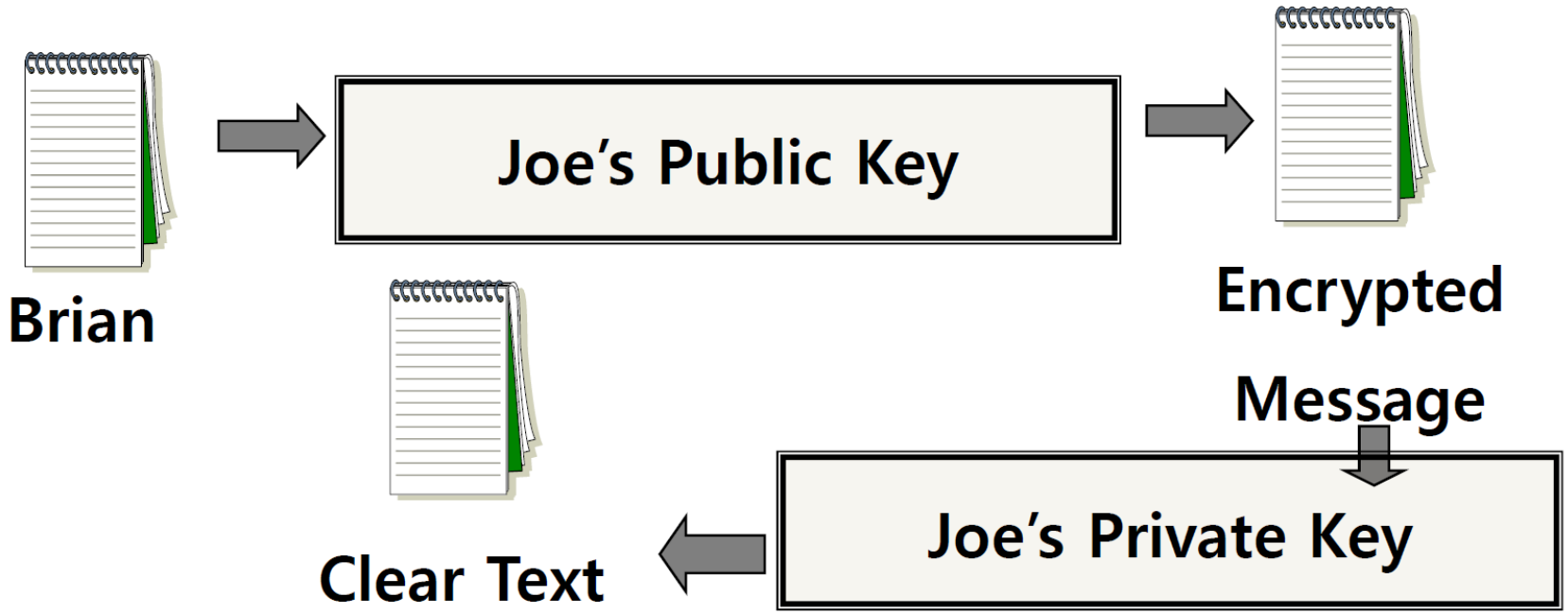
Secret Key (symmetric) Cryptography

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



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.



Digital Signature

- An electronic stamp or seal
  - almost exactly like a written signature, except more guarantees!
- Is appended to a document
  - Or sent separately (detached signature)
- Ensures data integrity
  - document was not changed during transmission
  - intended to solve the problem of tampering and impersonation in digital communications.
- Based on Public Key Cryptography
- Reference

Steps for Generating a Digital Signature

SENDER:

- Generate a Message Digest
  - The message digest is generated using a set of hashing algorithms
  - Even the slightest change in the message produces a different digest
- Create a Digital Signature
  - The message digest is encrypted using the sender's private key. The resulting encrypted message digest is the digital signature
- Attach digital signature to message and send to receiver

Steps for Generating a Digital Signature

RECEIVER:

- Recover the Message Digest
  - Decrypt the digital signature using the sender's public key to obtain the message digest generated by the sender
- Generate the Message Digest
  - Use the same message digest algorithm used by the sender to generate a message digest of the received message
- 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

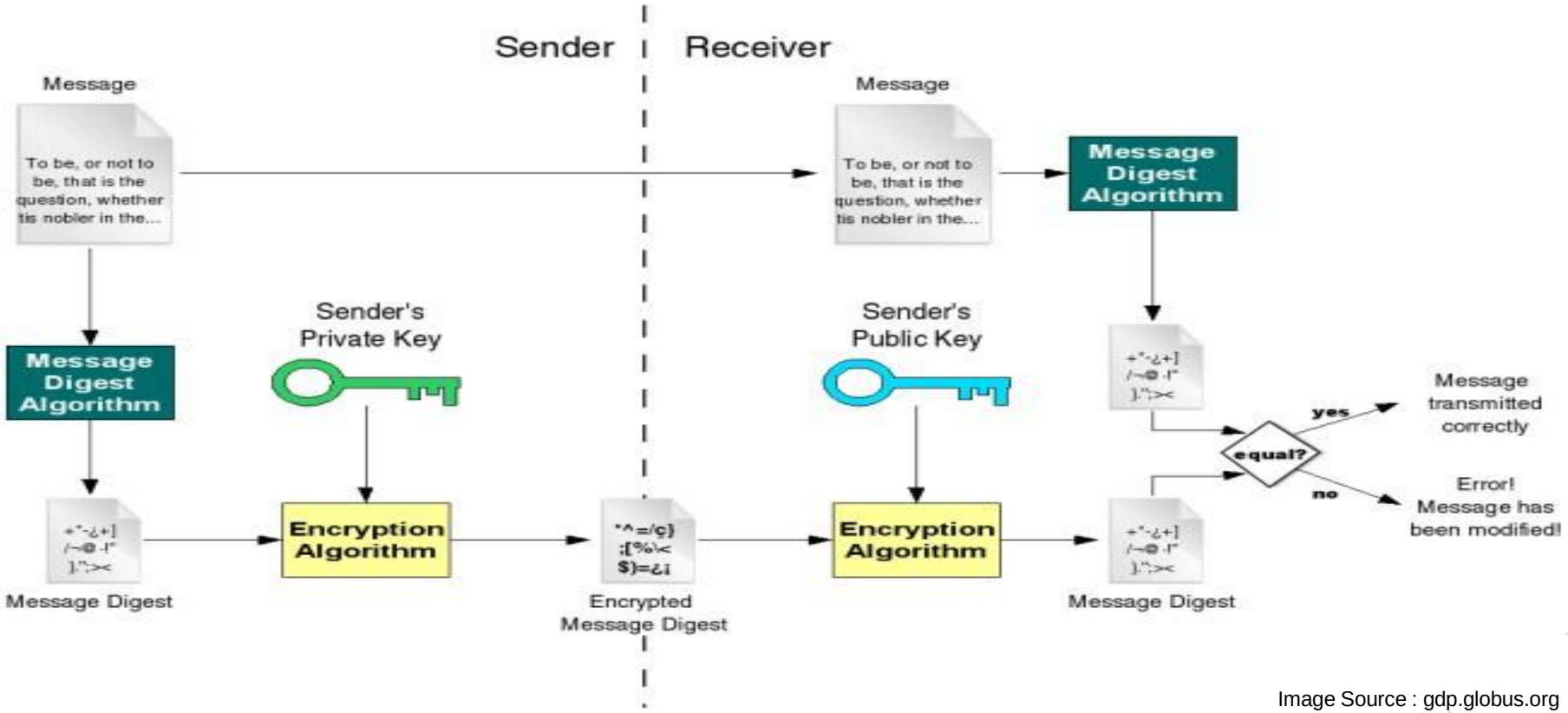


Image Source : gdp.globus.org

Detached Signature

- Digital signatures can either be attached to the message or detached
- A detached signature is stored and transmitted separately from the message it signs
- Commonly used to validate software distributed in compressed tar files
- You can't sign such a file internally without altering its contents, so the signature is created in a separate file

Assignment 7 - Laboratory

- 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

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 accunt for the client on the server
  - $ sudo useradd <username>
  - $ sudo passwd <username>
- Create .ssh directory for new user
  - $ 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 Address

- $ 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>(packet internet groper)
  - 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

Assignment 8 - Homework

- Answer 2 questions in the file hw.txt
- A file eeprom that is a copy of the file /sys/bus/i2c/devices/0-0050/eeprom on your BeagleBone.
- https://www.gnupg.org/gph/en/manual.html
- Generate a key pair with the GNU Privacy Guard's commands (choose default options when prompted)
- Export public key, in ASCII format, into hw-pubkey.asc
- Use the private key you created to make a detached clear signature eeprom.sig for eeprom
- Use given commands to verify signature and file formatting
  - These can be found at the end of the assignment spec

Assignment 8 - Homework

- GNU Privacy Guard (GnuPG)
  - 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

Assignment 8 - Homework

- 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

Presentations

- Today's Presentation:
  - Zhenghao Sun
  - Yu Yang
- Next up:
  - Junhong Wang
  - Don

Questions?