# CS35L Software Construction Laboratory

Lab 5: Sneha Shankar

Week 8; Lecture 2

# Digital Signature

- An electronic stamp or seal
  - almost exactly like a written signature, except it gives more guarantees!
- Is appended to a document
  - Or sent separately (detached signature)
- Ensures data integrity
  - document was not changed during transmission

# 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

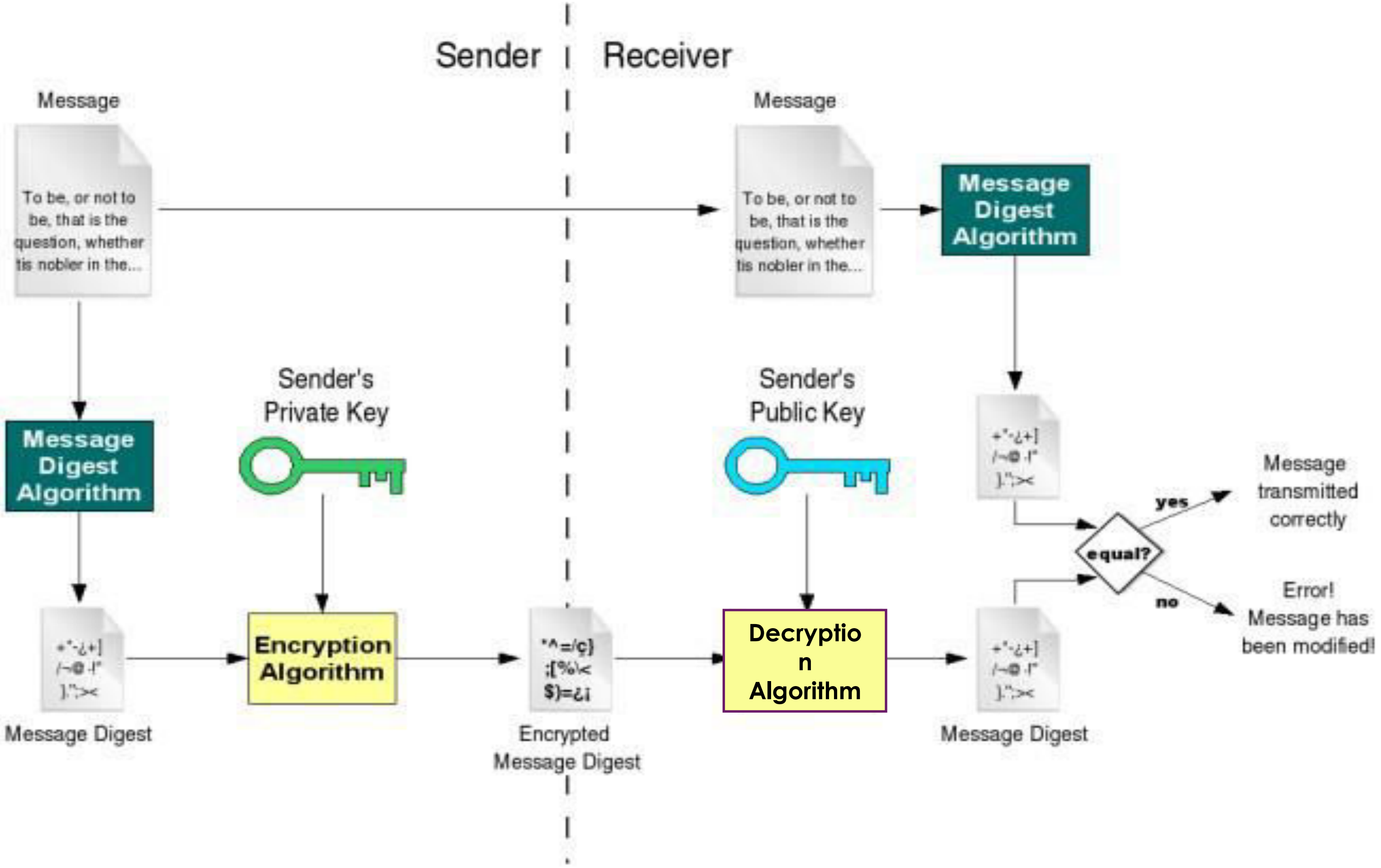
**RECEIVER:**

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



# 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.

# GNU Privacy Guard

- 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

# Homework 7

- 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