# SSH - Secure Shell

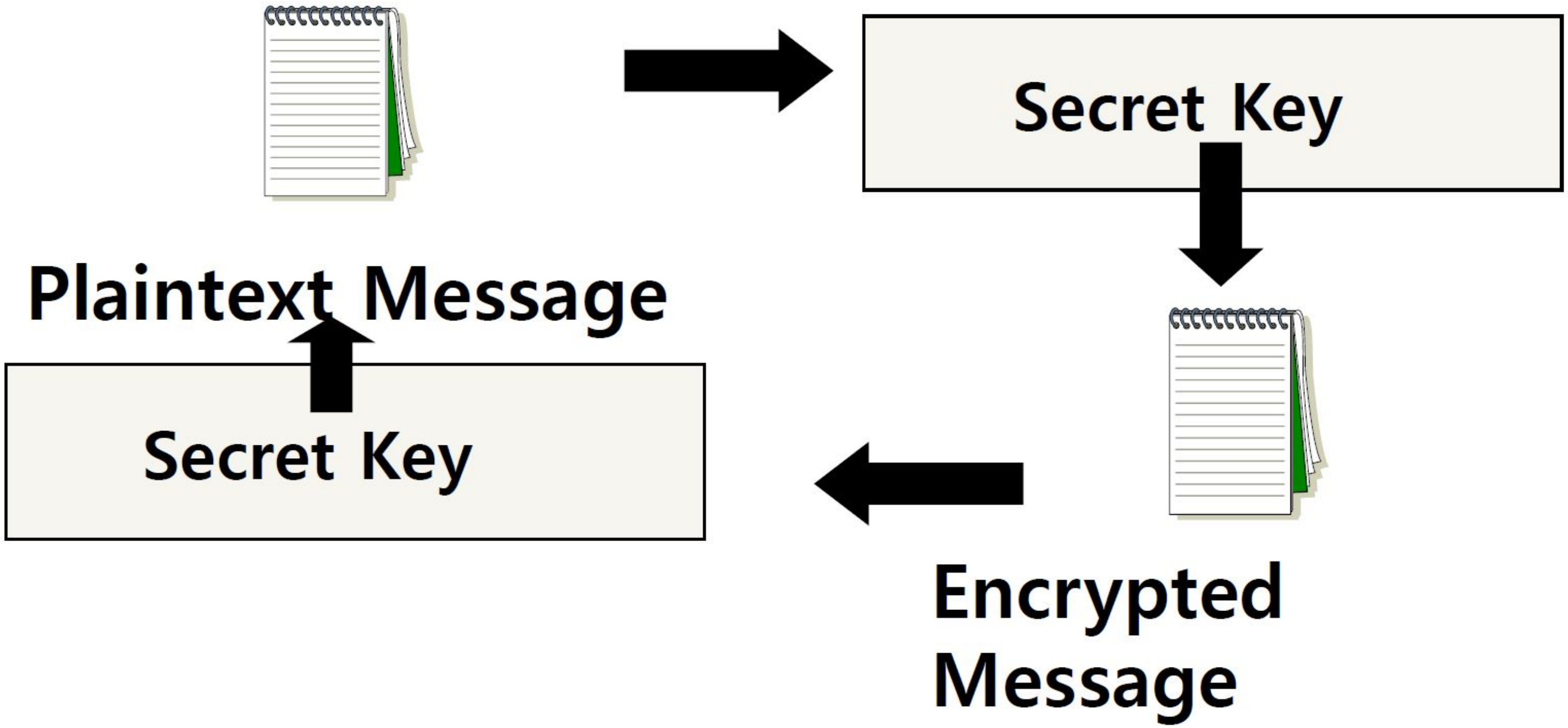
CS 35L
Spring 2018 - Lab 3

# Reminder: Communication Over the Internet

- What type of guarantees do we want?
  - **Confidentiality**
    - Message secrecy
  - **Data integrity**
    - Message consistency
  - **Authentication**
    - Identity confirmation
  - **Authorization**
    - Specifying access rights to resources

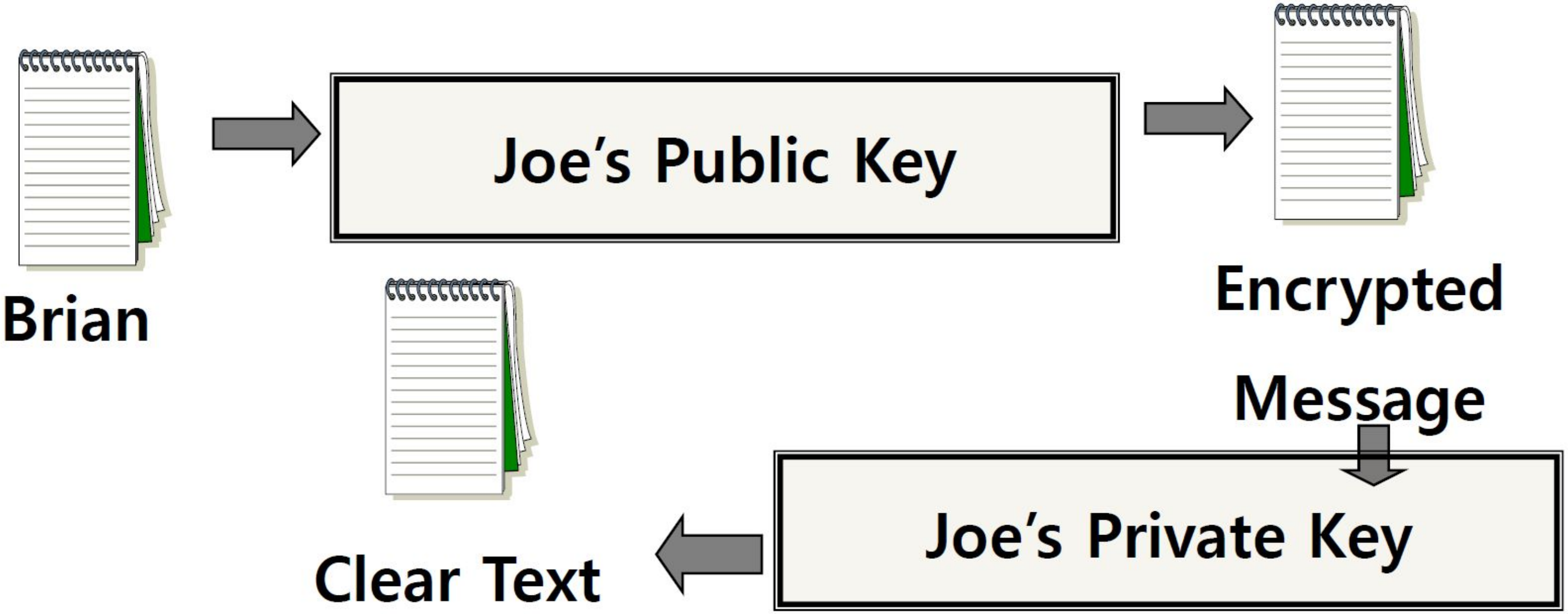
# Reminder: Secret Key (symmetric) Cryptography

- 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

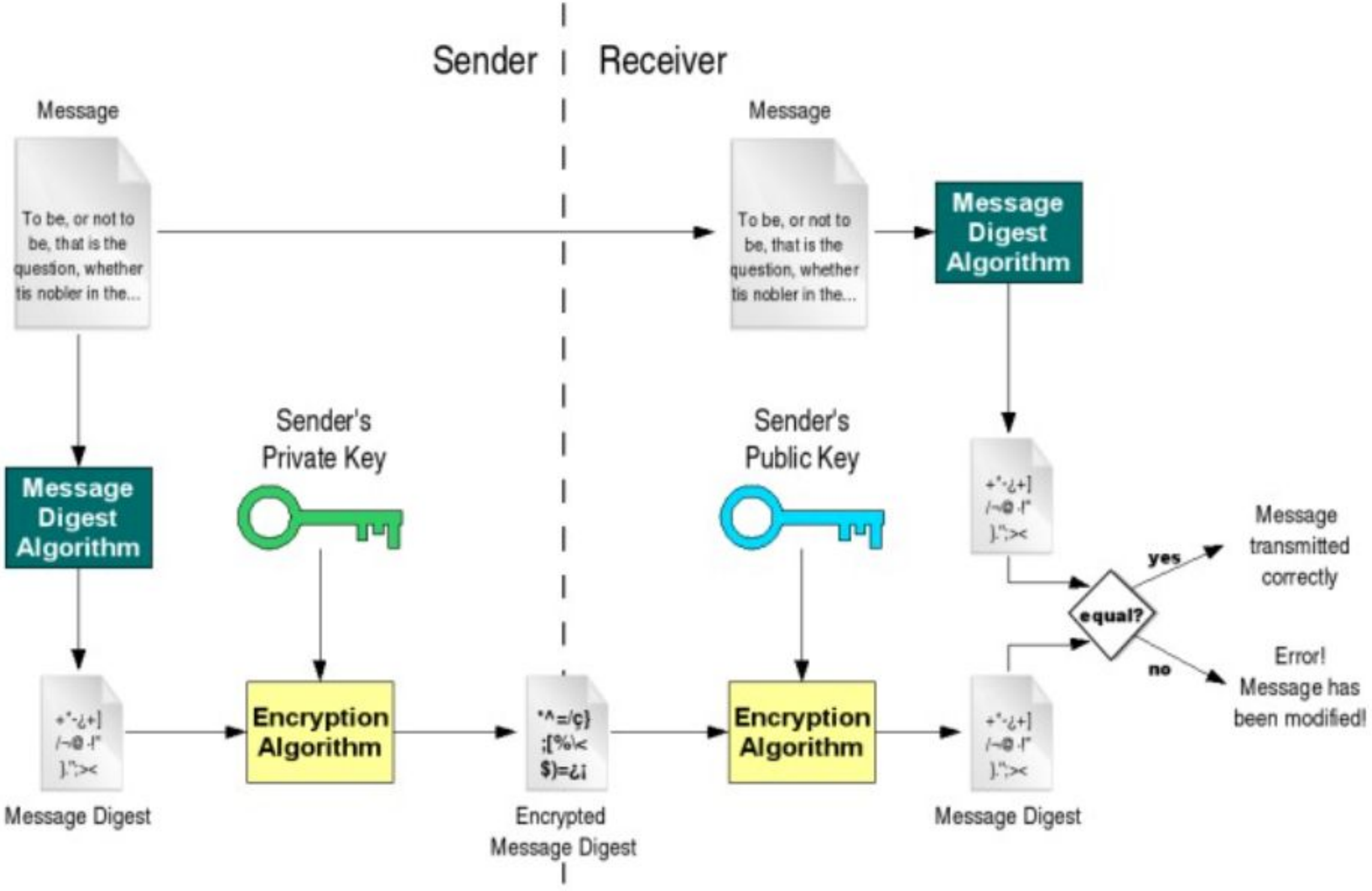
# Steps for Generating a Digital Signature

**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 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’
    ```
- Use the private key you created to make a detached clear signature `eeprom.sig` for `eeprom`

  -
    ```
    $ gpg --armor --output eeprom.sig --detach-sign
    eeprom
    ```
- Use given commands to verify signature and file formatting
  - These can be found at the end of the assignment spec