Lecture7- Security Programming

NET 445 – Internet Programming

Secure Communication

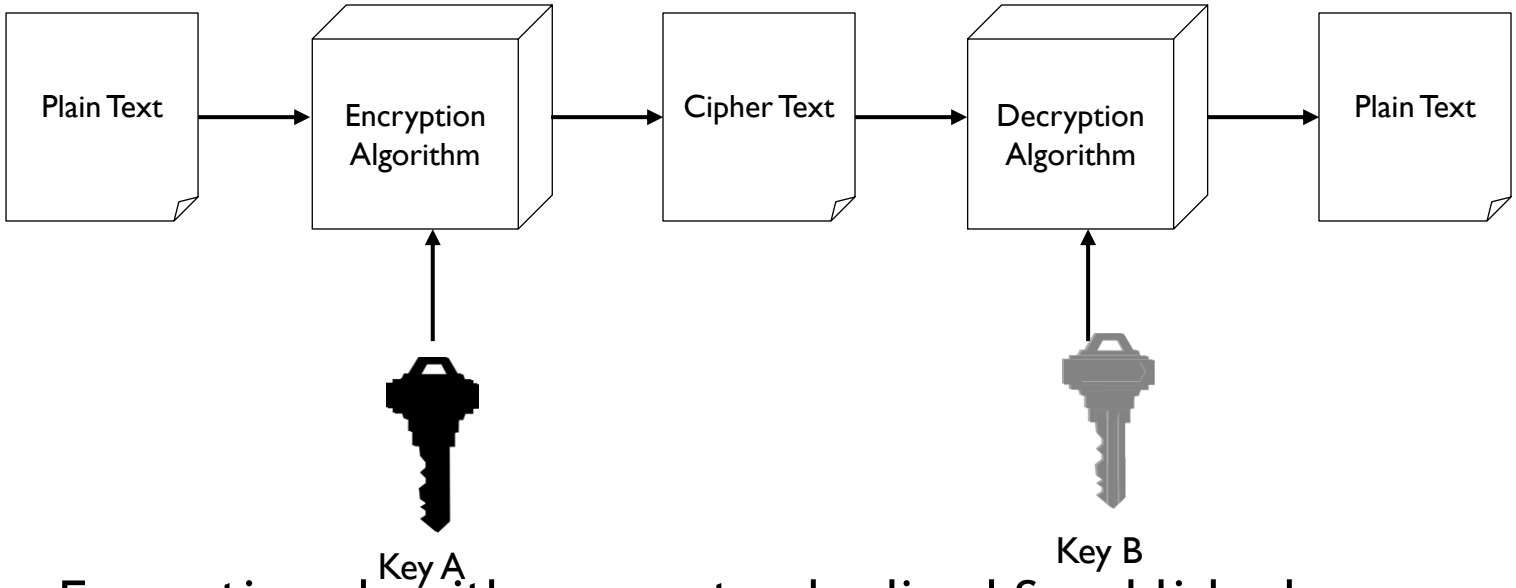
- Well established needs for secure communication
  - War time communication
  - Business transactions
- Requirements of secure communication
  1. Secrecy
     - Only intended receiver understands the message
  2. Authentication
     - Sender and receiver need to confirm each others identity
  3. Message Integrity
     - Ensure that their communication has not been altered, either maliciously or by accident during transmission

Cryptography

- Cryptography is the science of secret, or hidden writing
- It has two main Components:
  1. Encryption
     - Practice of hiding messages so that they can not be read by anyone other than the intended recipient
  2. Authentication
     - Ensuring that users of data/resources are the persons they claim to be and that a message has not been surreptitiously altered

Encryption - Cipher

- Cipher is a method for encrypting messages



- Encryption algorithms are standardized & published
- The key which is an input to the algorithm is secret
  - Key is a string of numbers or characters
  - If same key is used for encryption & decryption the algorithm is called symmetric
  - If different keys are used for encryption & decryption the algorithm is called asymmetric

Symmetric Encryption

- Algorithms in which the key for encryption and decryption are the same are Symmetric
  - Example: Caesar Cipher
- Types:
  1. Block Ciphers
     - Encrypt data one block at a time (typically 64 bits, or 128 bits)
     - Used for a single message
  2. Stream Ciphers
     - Encrypt data one bit or one byte at a time
     - Used if data is a constant stream of information

Symmetric Encryption – Key Strength

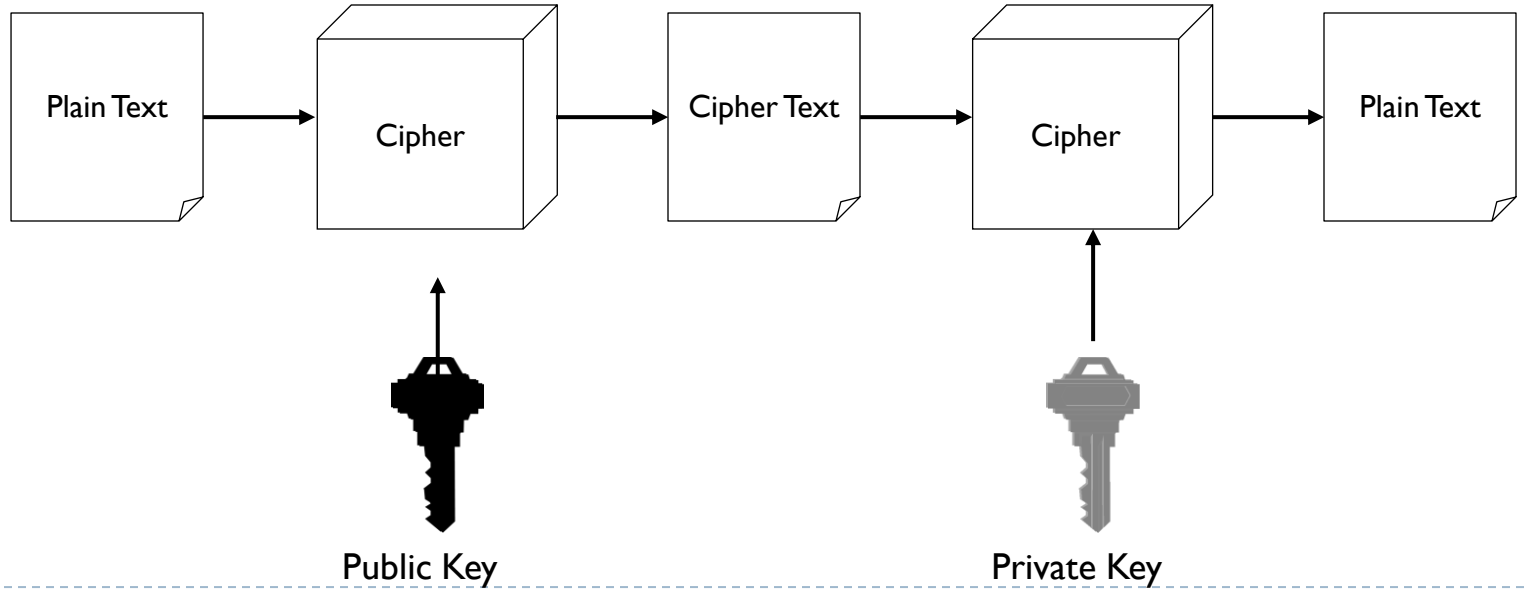
- Strength of algorithm is determined by the size of the key
  - The longer the key the more difficult it is to crack
- Key length is expressed in bits
  - Typical key sizes vary between 48bits and 448 bits
- Set of possible keys for a cipher is called key space
  - For 40-bit key there are 2^{40} possible keys
  - For 128-bit key there are 2^{128} possible keys
  - Each additional bit added to the key length doubles the security
- To crack the key the hacker has to use brute-force

  (i.e. try all the possible keys till a key that works is found)
  - Super Computer can crack a 56-bit key in 24 hours
  - It will take 2^{72} times longer to crack a 128-bit key

  (Longer than the age of the universe)

Asymmetric Encryption

- Uses a pair of keys for encryption
  - Public key for encryption
  - Private key for decryption
- Messages encoded using public key can only be decoded by the private key
  - Secret transmission of key for decryption is not required
  - Every entity can generate a key pair and release its public key



Asymmetric Encryption

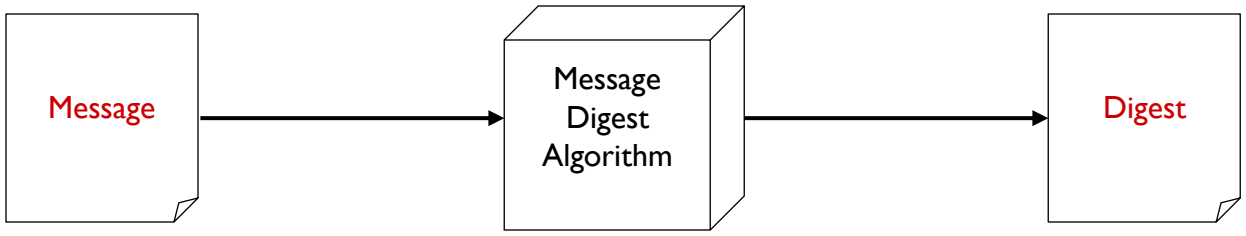
- Two most popular algorithms are RSA & El Gamal
  - RSA
    - Developed by Ron Rivest, Adi Shamir, Len Adelman
    - Both public and private key are interchangable
    - Variable Key Size (512, 1024, or 2048 buts)
    - Most popular public key algorithm
  - El Gamal
    - Developed by Taher ElGamal
    - Variable key size (512 or 1024 bits)
    - Less common than RSA, used in protocols like PGP

Asymmetric Encryption - Weaknesses

- Efficiency is lower than Symmetric Algorithms
  - A 1024-bit asymmetric key is equivalent to 128-bit symmetric key
- Potential for eavesdropping attack during transmission of key
- It is problematic to get the key pair generated for the encryption

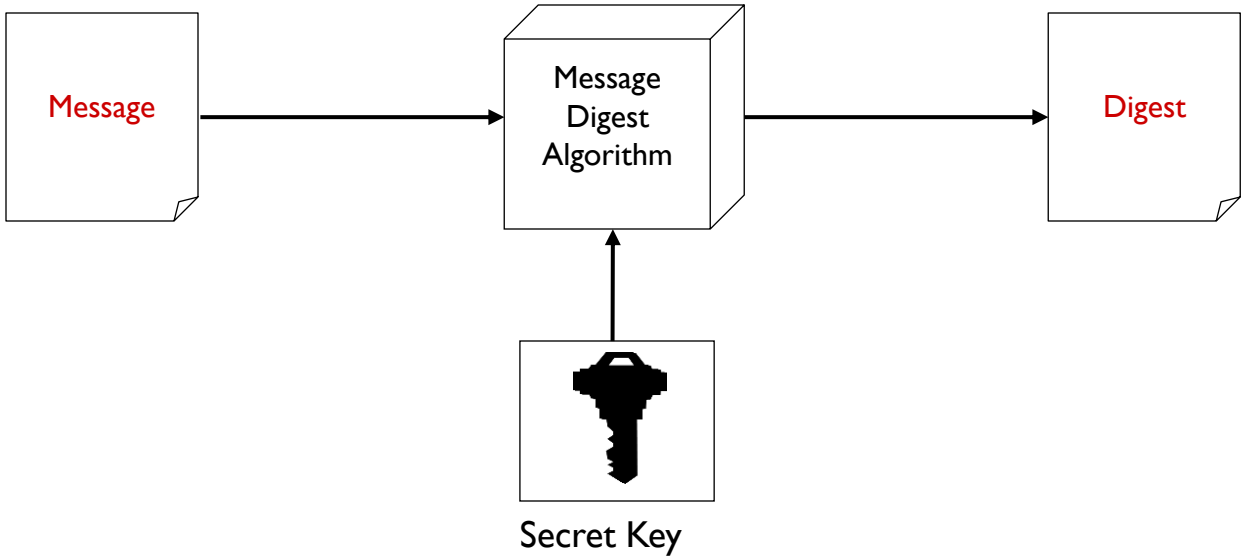
Authentication – Message Digests

- A message digest is a fingerprint for a document
- Purpose of the message digest is to provide proof that a document has not been tampered with.
- Hash functions used to generate message digests are one way functions that have following properties
  - It must be computationally infeasible to reverse the function
  - It must be computationally infeasible to construct two messages which which hash to the same digest
- Some of the commonly used hash algorithms are
  - MD5 – 128 bit hashing algorithm by Ron Rivest of RSA
  - SHA & SHA-1 – 162 bit hashing algorithm developed by NIST



Message Authentication Codes

- A message digest created with a key
- Creates security by requiring a secret key to be possesses by both parties in order to retrieve the message
- Some of the commonly used hash algorithms are
  - MD5 – 128 bit hashing algorithm by Ron Rivest of RSA
  - SHA & SHA-1 – 162 bit hashing algorithm developed by NIST



Install Security Libraries

```
pip3 install setuptools_rust

pip3 install cryptography

pip3 install pycryptodome
```

Simple Symmetric Encryption Using AES

- Using AES symmetric encryption

```
from cryptography.fernet import Fernet
key = Fernet.generate_key()
print ("Key is ", key)
cipher_suite = Fernet(key)
cipher_text = cipher_suite.encrypt(b"A really secret message. Not for
prying eyes.")
print ("The cipher message is ", cipher_text)
plain_text = cipher_suite.decrypt(cipher_text)
print ("The plain message is ", plain_text)
```

Simple Symmetric Encryption Using RSA

- Public and private key generations
- Encryption and decryption

```
from Crypto.PublicKey import RSA
from Crypto.Cipher import PKCS1_OAEP
import binascii
keyPair = RSA.generate(1024)

pubKey = keyPair.publickey()
print(f"Public key:  (n={hex(pubKey.n)}, e={hex(pubKey.e)})")
pubKeyPEM = pubKey.exportKey()
print(pubKeyPEM.decode('ascii'))

print(f"Private key: (n={hex(pubKey.n)}, d={hex(keyPair.d)})")
privKeyPEM = keyPair.exportKey()
print(privKeyPEM.decode('ascii'))

msg = b'A message for encryption'
encryptor = PKCS1_OAEP.new(pubKey)
encrypted = encryptor.encrypt(msg)
print("Encrypted:", binascii.hexlify(encrypted))

decryptor = PKCS1_OAEP.new(keyPair)
decrypted = decryptor.decrypt(encrypted)
print('Decrypted:', decrypted)
```

Message Authentication using Sha2 hash

- Message Authentication
- Using Sha2 hash

```
# import the library module
import hashlib

# initialize a string
str = "NET445"

# encode the string
encoded_str = str.encode()

# create sha-2 hash objects initialized with the encoded string
hash_obj_sha224 = hashlib.sha224(encoded_str)   # SHA224
hash_obj_sha256 = hashlib.sha256(encoded_str)   # SHA256
hash_obj_sha384 = hashlib.sha384(encoded_str)   # SHA384
hash_obj_sha512 = hashlib.sha512(encoded_str)   # SHA512

# print
print("\nSHA224 Hash: ", hash_obj_sha224.hexdigest())
print("\nSHA256 Hash: ", hash_obj_sha256.hexdigest())
print("\nSHA384 Hash: ", hash_obj_sha384.hexdigest())
print("\nSHA512 Hash: ", hash_obj_sha512.hexdigest())
```

References:

- Foundations of Python Network Programming Third Edition by Brandon Rhodes (2014)
- James F. Kurose, and Keith W Ross, Computer Networking: A Top-Down Approach, 6th Edition
- Python 3 documentation
- https://wiki.python.org/moin/UdpCommunication
- https://www.w3schools.com/python/
- https://www.tutorialspoint.com/python/