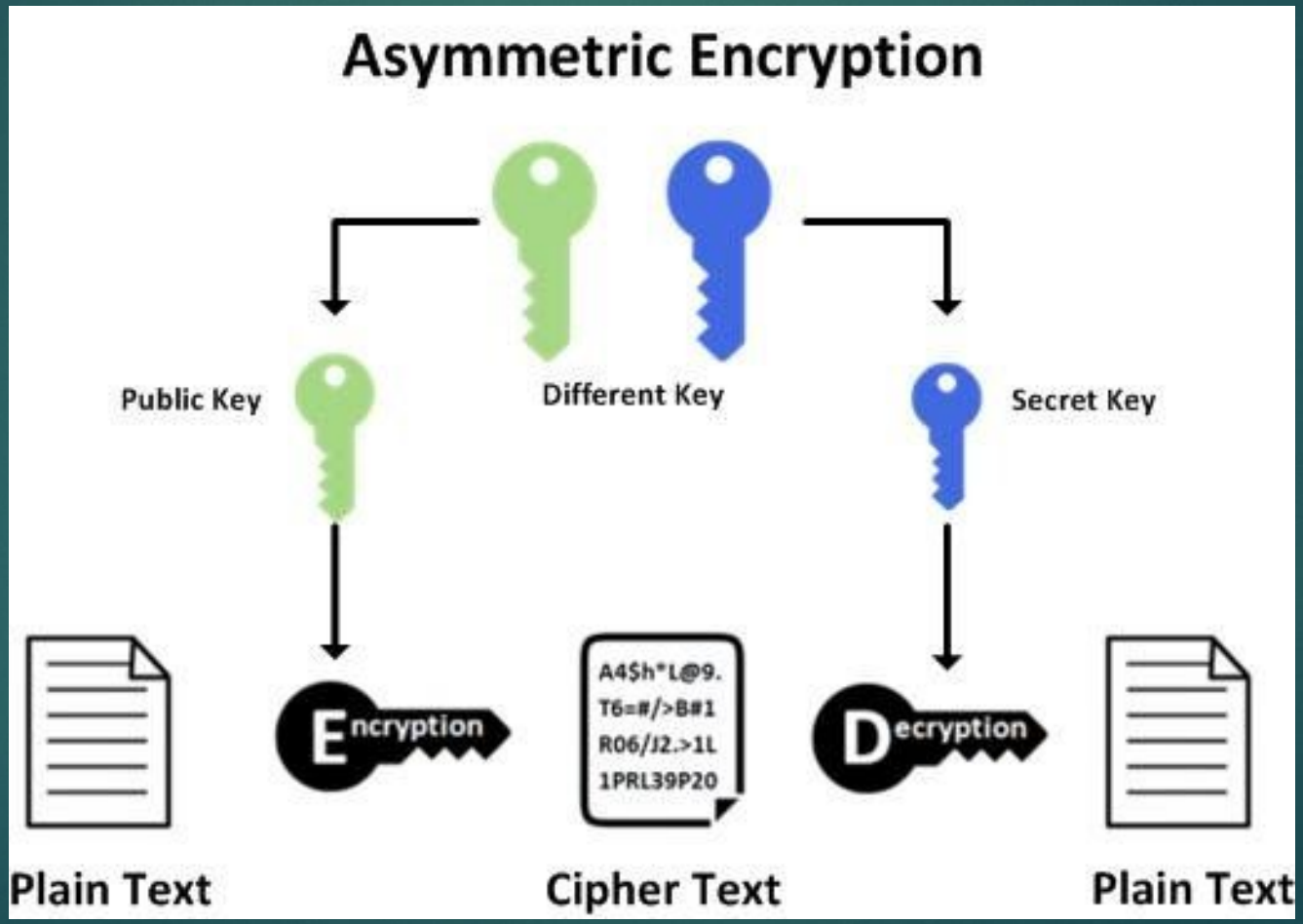
Network Security Laboratory Session 3

SSL AND TLS

Asymmetric Cryptography



Secure Socket Layer (SSL) and Transport Layer Security (TLS)

- Cryptographic protocols designed to provide security over network
- TLS: provides privacy and data integrity between hosts
- The connection is **SECURE** since data are encrypted with symmetric cryptography
- The **IDENTITY** of hosts is authenticated with public key cryptography
- The connection is **RELIABLE** since messages includes a message integrity check using a Message Authentication Code to prevent manipulation during transmission

How does it work?

- The client contacts the server using a secure URL (HTTPS://)
- The server sends the client its certificate and public key
- The client verifies this with a Trusted Root Certification Authority to ensure the certificate is legitimate
- The client and server negotiate the strongest type of encryption that each can support
- The client encrypts a session (secret) key with the server's public key, and sends it back to the server
- The server decrypts the client communication with its private key, and the session is established
- The session key (symmetric encryption) is now used to encrypt and decrypt data transmitted between the client and server

Certificates

- It is an electronic document used to verify the owner's identity
- It includes information about owner identity, the public key and the signature of the entity who verified the certificate's content
- In a common Public Key Infrastructure (PKI), certificates are released by a Certificate Authority (CA)
- Based on X509 protocol

Certificate Validation

- When a connection is setup, the server sends to the client its certificate and the client checks if it is valid

- In order to do that, the client has to check if:

1. The subject of the certificate matches the hostname (i.e., domain name) to which the client is trying to connect
2. The certificate is signed by a trusted certificate authority

- A TLS server may be configured with a self-signed certificate
- In this case clients will generally be unable to verify the certificate thus, communication will end (unless the certificate checking is disabled)

Self Signed Certificates

- We could generate self-signed certificates
- In order to create a self-signed certificate we must create a custom Certificate Authority
- This type of certificate could be used for testing purposes
- They are not valid since self-signed certificates are “NOT TRUSTED” by a Certificate Authority

ES 01 - Test SSL/TLS Stream

- In order to create a simple connection between 2 hosts using SSL/TLS protocol, we could use openssl
- Server Side: openssl s_server –key [key] –cert [cert] -accept <<port>>
- Client side: openssl s_client <<host>>:<<port>>
- On wireshark we could see handshake and how message are encrypted

ES 02 - WebServer

- Create your own web server (in a host of your choice in the GNS3 laboratory)
- Install a new self-signed certificate
- Generate a valid certificate using letsencrypt
  - Letsencrypt is a free Certificate Authority
- Try using a simple curl from another host in the lab with both self-signed and letsencrypt certificates. What difference do you notice?

ES 03 - TLS Socket with python

- create a python script that connects to an **https** site and prints its contents to stdout
- create a TLS client that connects to an encrypted server and prints debugging information on encryption (see exercise text on github for more details)
- create a TLS client and a TLS server that communicate with each other securely (uses certificates obtained with letsencrypt)