Encryption & SSL Certificates

Securing web traffic 101

What is Encryption?

Encryption is the process of encoding a message or information in such a way that only authorized parties can access it and those who are not authorized cannot

The message in its readable form is known as plain-text and after it has been encrypted it is known as cipher-text

Encryption is the basis for several digital security approaches and we have mentioned using it in previous lectures without covering how it actually works

How does encryption work?

- Encryption is accomplished by scrambling the bits, characters, words, phrases, etc in the original message to produce an unintelligible ciphertext.
- It performs the scrambling of information by using a combination of Transposition and Substitution

- Transposition - changes the order of bits, characters, etc without modifying the content itself.
  - For example, “Hello World” could be transposed into “eHll ooWlrd”

- Substitution - changes the content of the bits, characters, etc without changing their position/ordering
  - For example, “Hello World” after substitution may appear as “aNSSbUxbeSo”

How does encryption work?

While substitution and transposition do a decent job of scrambling data, what is the inherent flaw with just using them on the input data stream?

- It means the process of scrambling the data must be kept secret!
- If the process is secret, ensuring it is available on the systems you want to use it on (any only those systems) is very difficult

So what is the solution here if the process for scrambling the data must be public?

- Introduce entropy into the input stream!
- This comes in the form of a “Key”
- If you have the “Key” you can use the public process to perform Encryption/Decryption

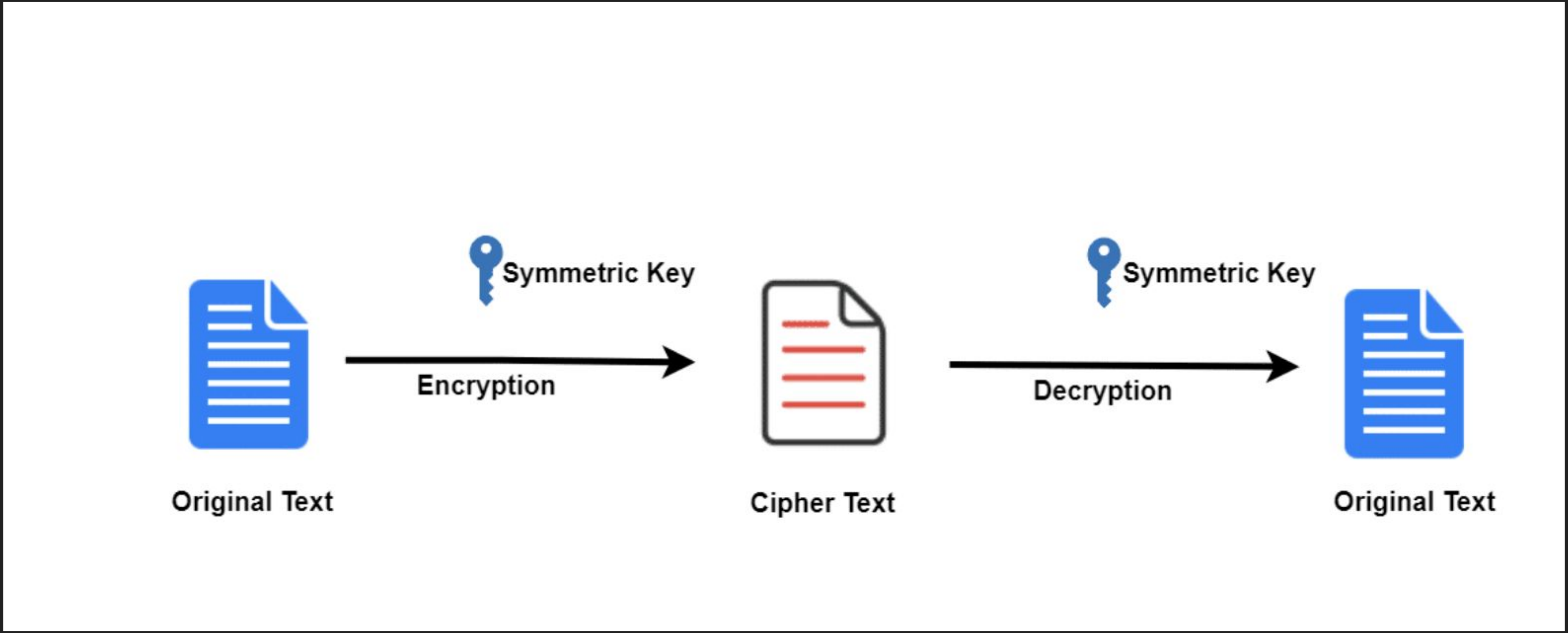
Types of Encryption

Encryption usually falls into two categories - Symmetric Key and Asymmetric Key cryptography.

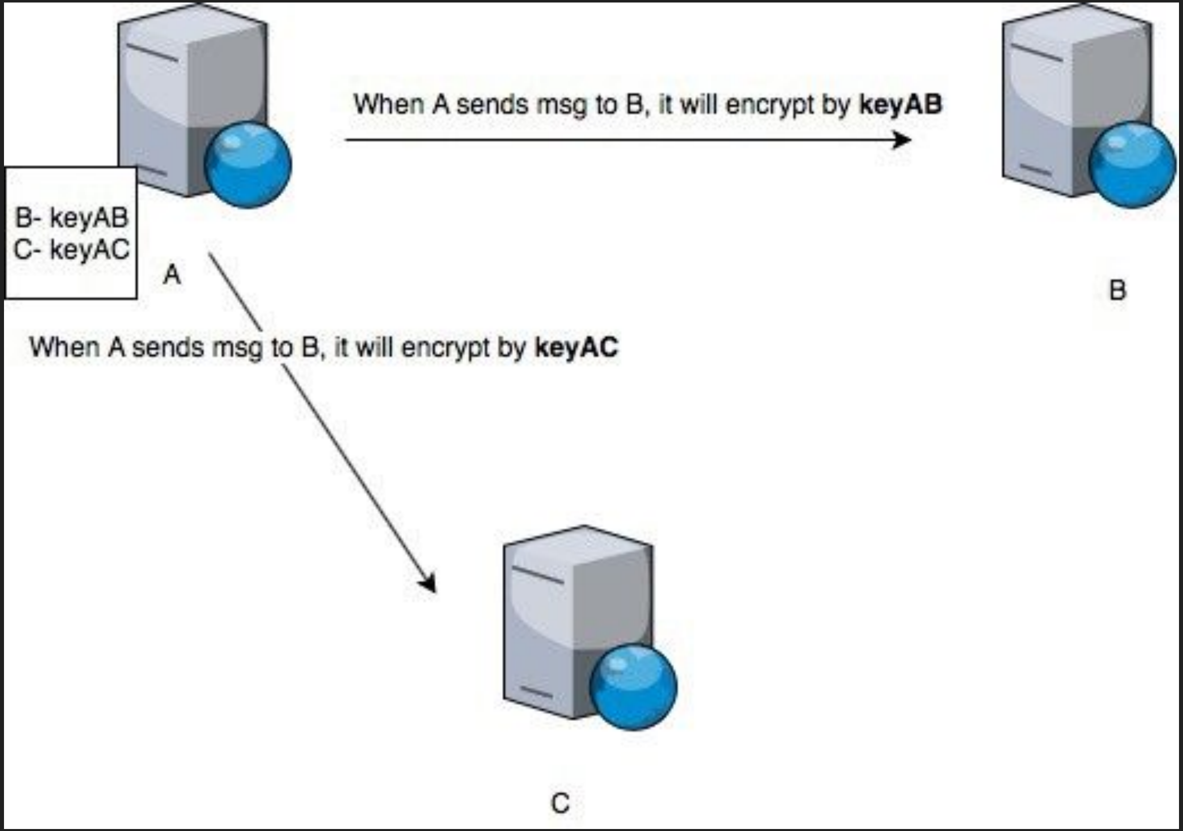
- Symmetric Key
    - Algorithm uses one key that is shared by sending and receiving parties
    - Key is assumed to be transferred over a secure means
    - Not very useful for internet communication
    - General fast execution of algorithms

- Asymmetric Key
    - Algorithms use multiple keys that are shared by sending and receiving parties
    - Keys are assumed to be transferred over a insecure means
    - General slower execution of algorithms
    - Keys can only decode text encoded with its pair (Public – Private)

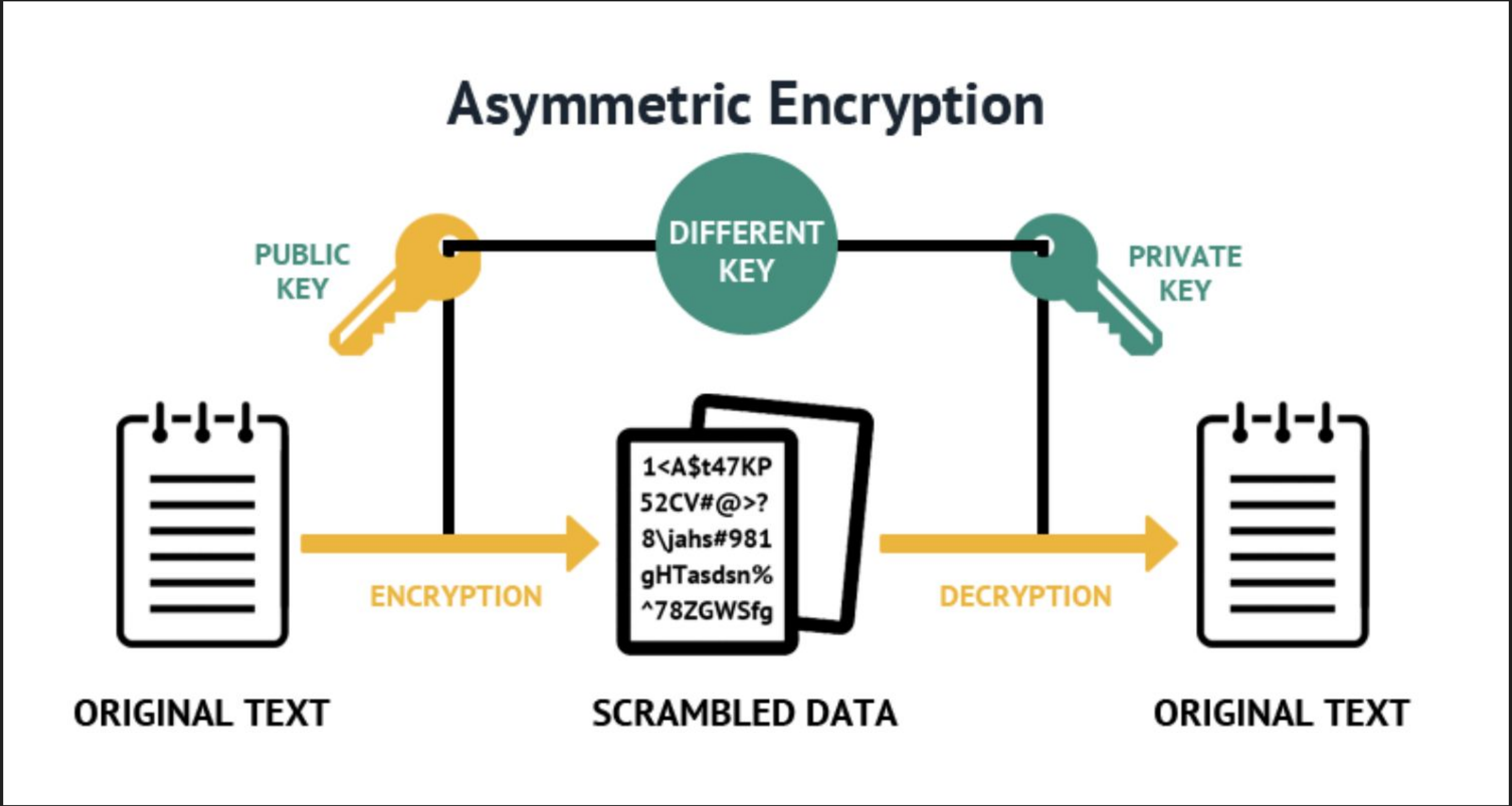
Types of Encryption



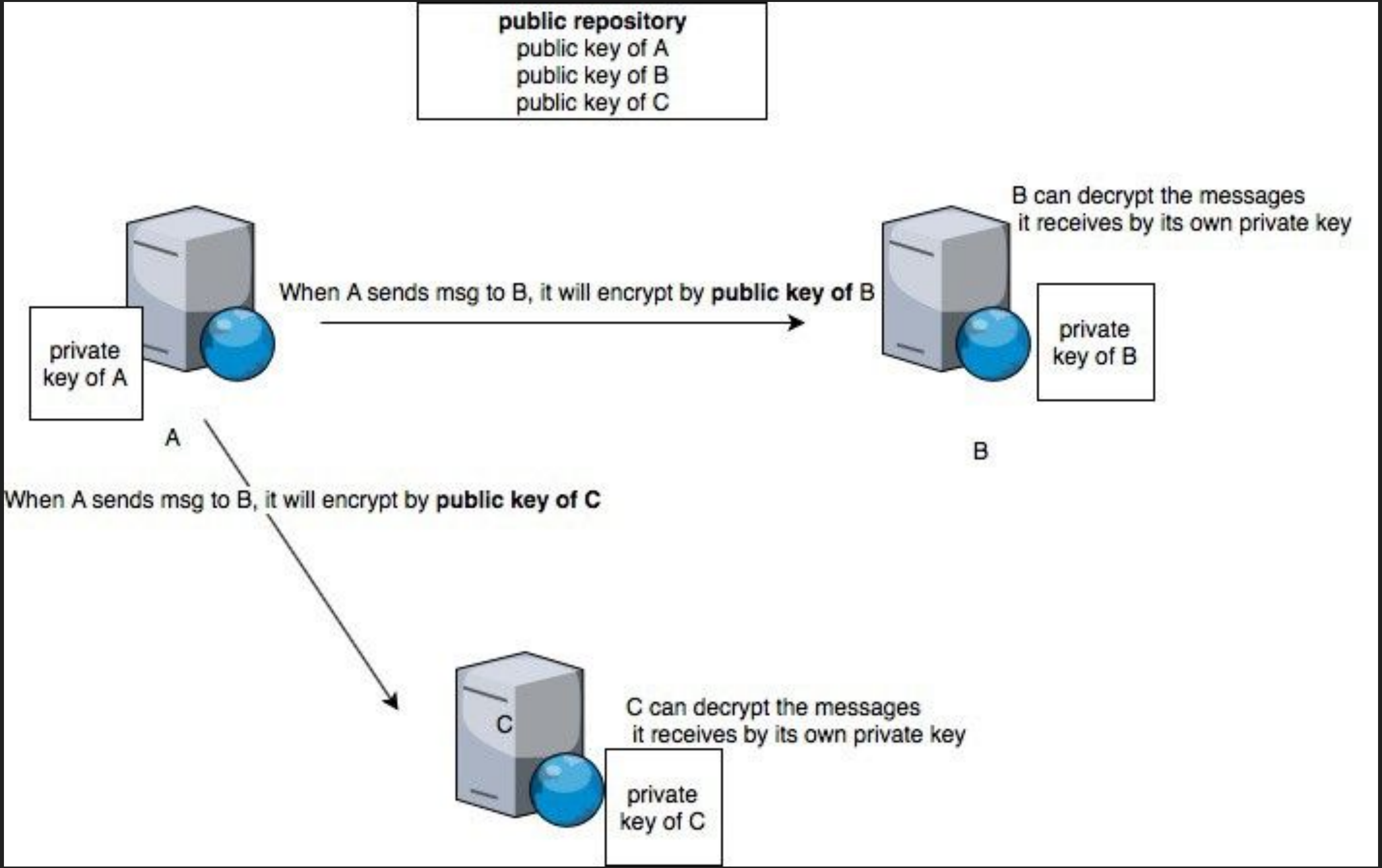
Types of Encryption


When A sends msg to B, it will encrypt by **keyAB**
B- keyAB
C- keyAC
A
B
When A sends msg to B, it will encrypt by **keyAC**
C

Types of Encryption



Types of Encryption


public repository
public key of A
public key of B
public key of C
B can decrypt the messages it receives by its own private key
When A sends msg to B, it will encrypt by **public key of** B
private key of A
A
private key of B
B
When A sends msg to B, it will encrypt by **public key of C**
C
C can decrypt the messages it receives by its own private key
private key of C

Signing and Verification

In addition to encrypting data itself, encryption can also be used for signing and verification of a document to ensure it is not modified in transit as well as for signifying approval

Signing allows for authentication of a user in that they were the original author of the message

So how does signing work?

Signing

- When Bob wants to sends a message to Alice, he takes the message and ‘hashes’ the message using a digital signature generation algorithm (i.e SHA)

- Bob will then encrypt the hashed ‘signature’ with his private key

- Then Bob will combine the original message and hashed message and encrypt that using the public key of Alice

- Now Bob sends the message to Alice

Verification

- Now Alice receives the message

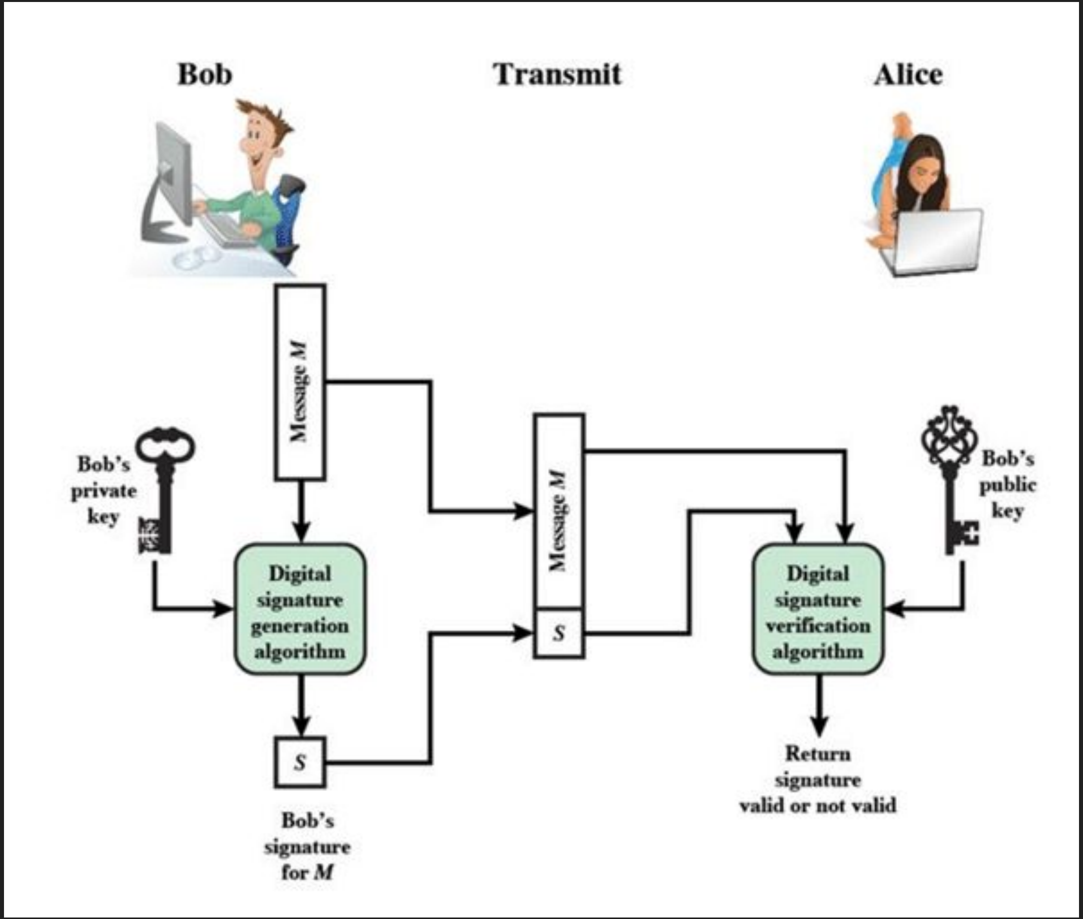
- Alice will decrypt the entire message using their private key

- Inside the message, there are two components: The original message, and the hashed text which is encrypted by Bob’s private key

- Then Alice will hash the original message using the hashing algorithm used by Bob. Now Alice has the hashed text of the original message

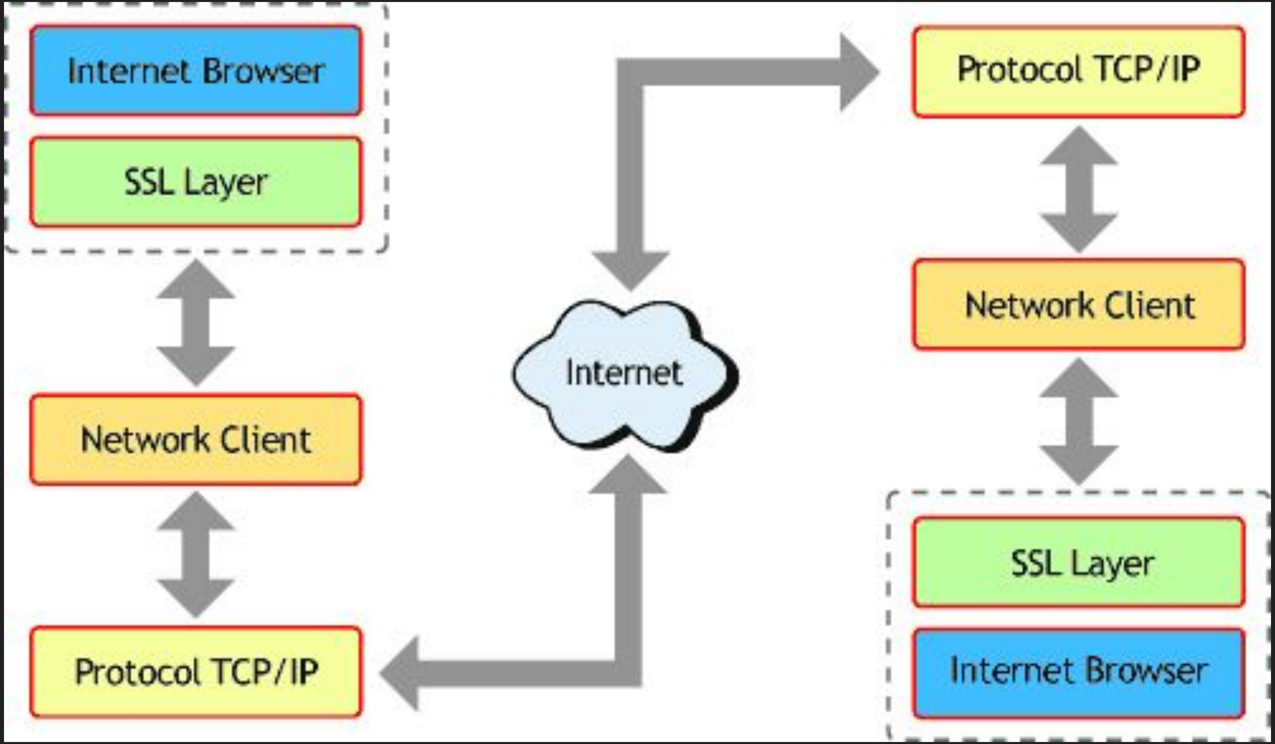
- Then Alice will decrypt the hashed text sent by Bob using the public key of Bob and compare it with the original message hashed by Alice. Now Alice can verify that the message is sent by Bob

Signing and Verification


Bob
Transmit
Alice
Message M
Bob's private key
Digital signature generation algorithm
S
Bob's signature for M
Message M
S
Digital signature verification algorithm
Bob's public key
Return signature valid or not valid

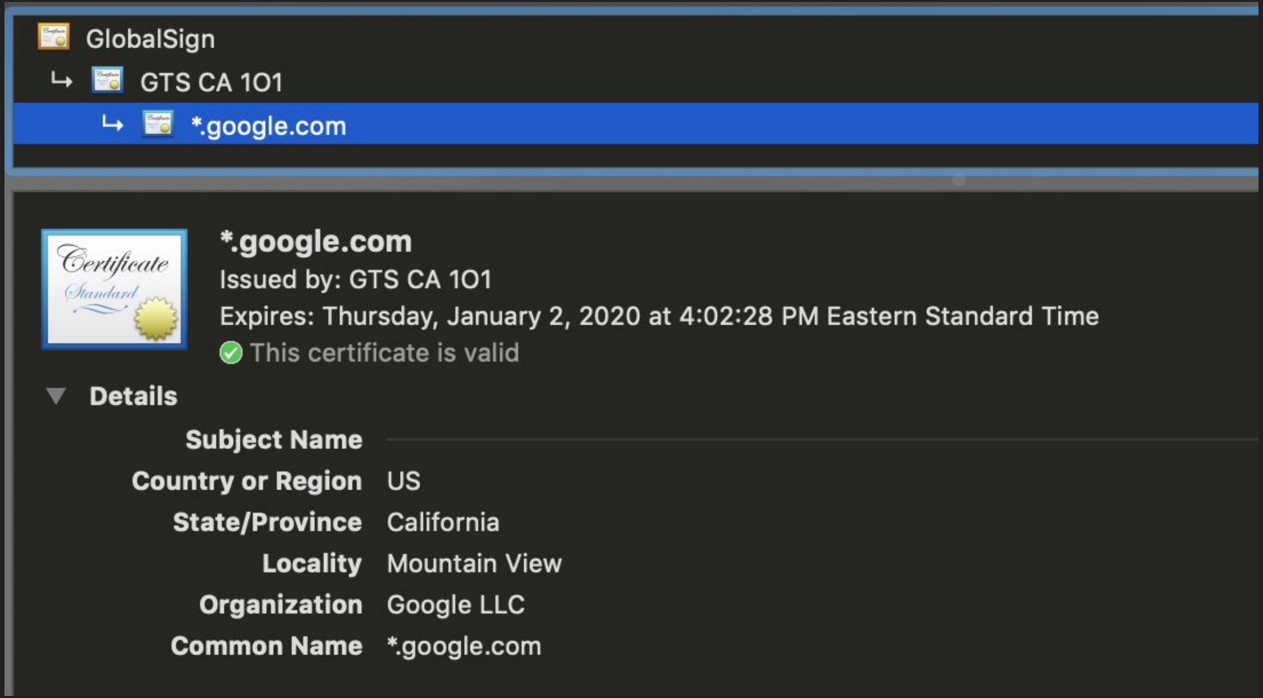
What about HTTPS?

When referring to HTTPS, what we are actually talking about is HTTP + Secure Socket Layer (SSL) now replaced by Transport Layer Security (TLS)



X.509 Certificates

HTTPS leverages X.509 Certificates for Encryption and Verification when communicating with web-servers on the internet



GlobalSign
↳ GTS CA 1O1
↳ *.google.com

***.google.com**
Issued by: GTS CA 1O1
Expires: Thursday, January 2, 2020 at 4:02:28 PM Eastern Standard Time
✓ This certificate is valid

▼ **Details**

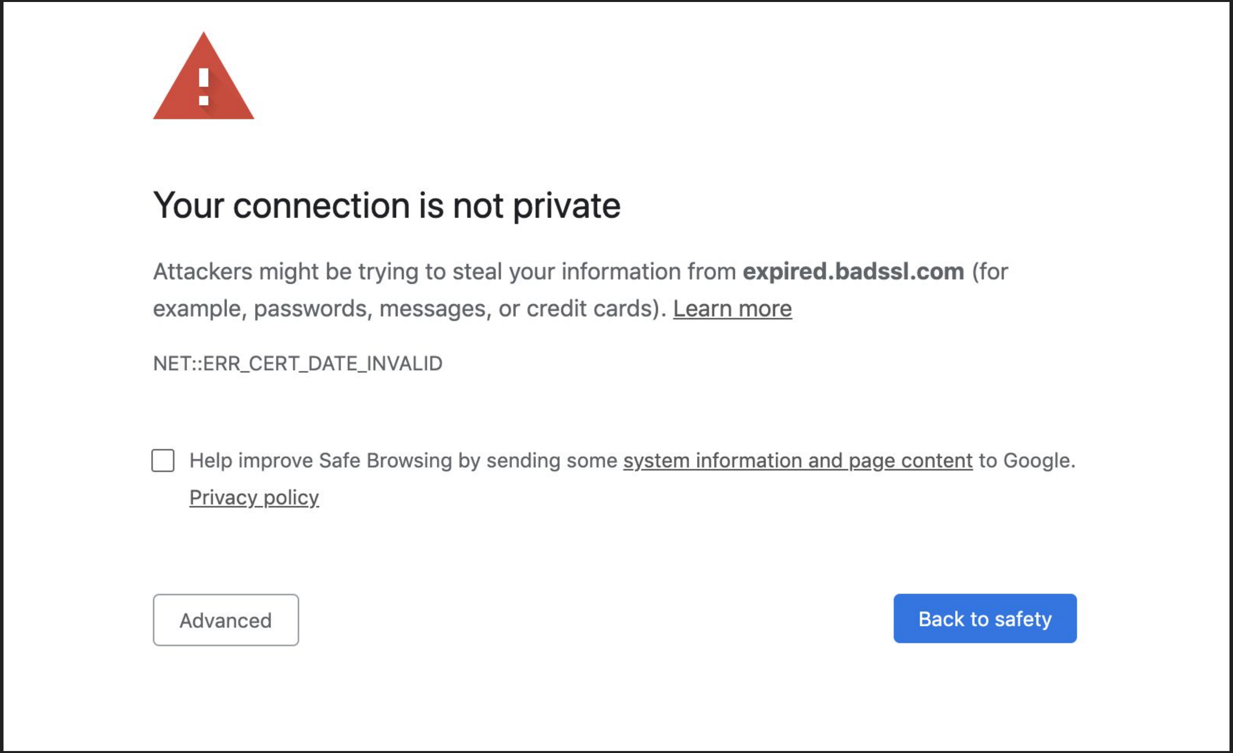
Subject Name	
Country or Region	US
State/Province	California
Locality	Mountain View
Organization	Google LLC
Common Name	*.google.com

Anatomy of X.509 Certificates

- **Subject** - Provides the name of the computer, user, network device, or service that the CA issues the certificate to
- **Issuer** - Provides a distinguished name for the CA that issued the certificate. For a root CA, the Issuer and Subject are identical. For all other CA certificates and for end entity certificates, the Subject and Issuer will be different
- **Valid From** - Provides the date and time when the certificate becomes valid
- **Valid To** - Provides the date and time when the certificate is no longer considered valid
- **Subject Alternative Name** - A subject can be presented in many different formats, used to provide additional subjects that this certificate is valid for.
- **Public Key** - Contains the public key of the key pair that is associated with the certificate.

Certificate Security

There are several instances where a certificate may not be considered as valid. Who has seen something like this before? What did you do?



Your connection is not private

Attackers might be trying to steal your information from **expired.badssl.com** (for example, passwords, messages, or credit cards). Learn more

NET::ERR_CERT_DATE_INVALID

☐ Help improve Safe Browsing by sending some system information and page content to Google. Privacy policy

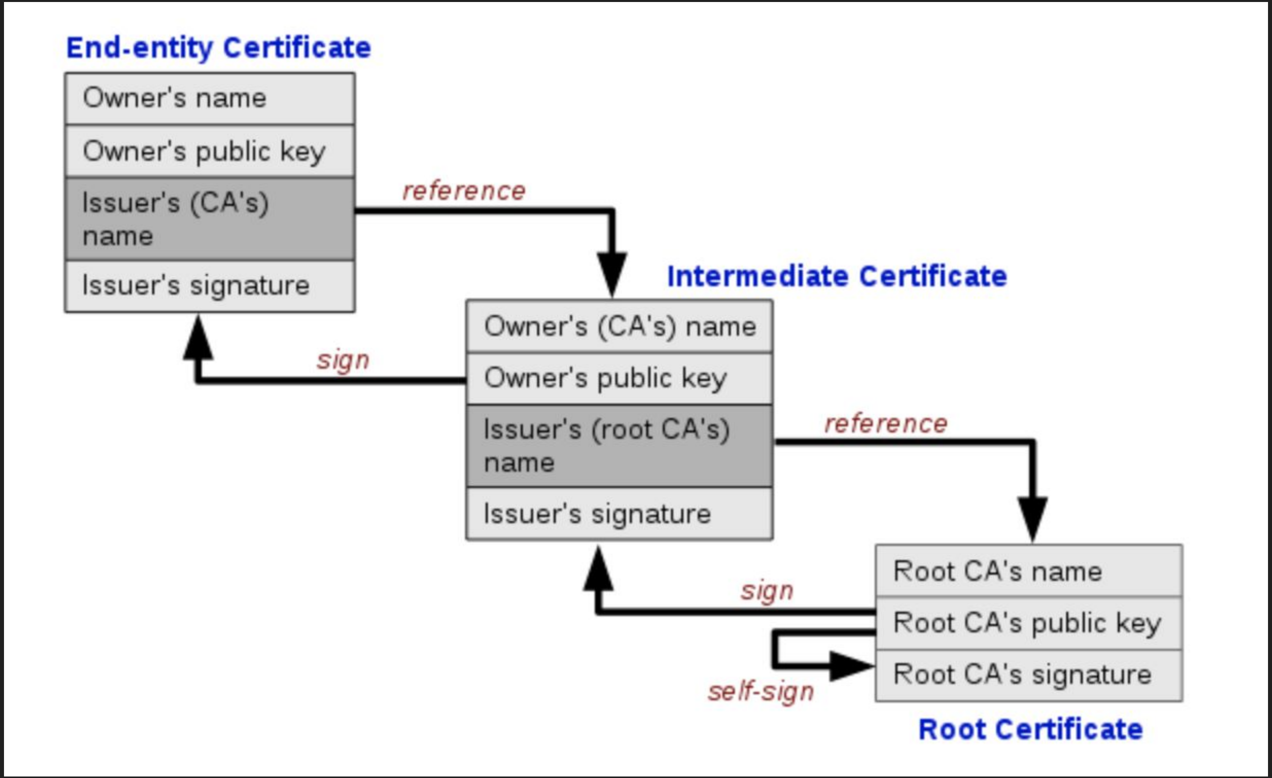
Certificate Security

- So why do certificates have expiration dates?
    - Helps mitigate vulnerabilities stemming from evolving security standards and changing ownership and control of companies and domain names over time

- What happens if a certificate key is compromised? (i.e. Someone pushed it to a public git repo…)
    - We use a Certificate Revocation List! A CRL is a list of digital certificates that have been revoked by the issuing Certificate Authority (CA) before their scheduled expiration date and should no longer be trusted

- So how is a certificate validated? How do we know which certs to trust and which we should not?

Certificate Verification



Don’t forget, the components of each cert in the chain also need to be validated!

Certificate Verification

- So now we can verify a certificate all the way up the hierarchy up to the Root Certificate Authority

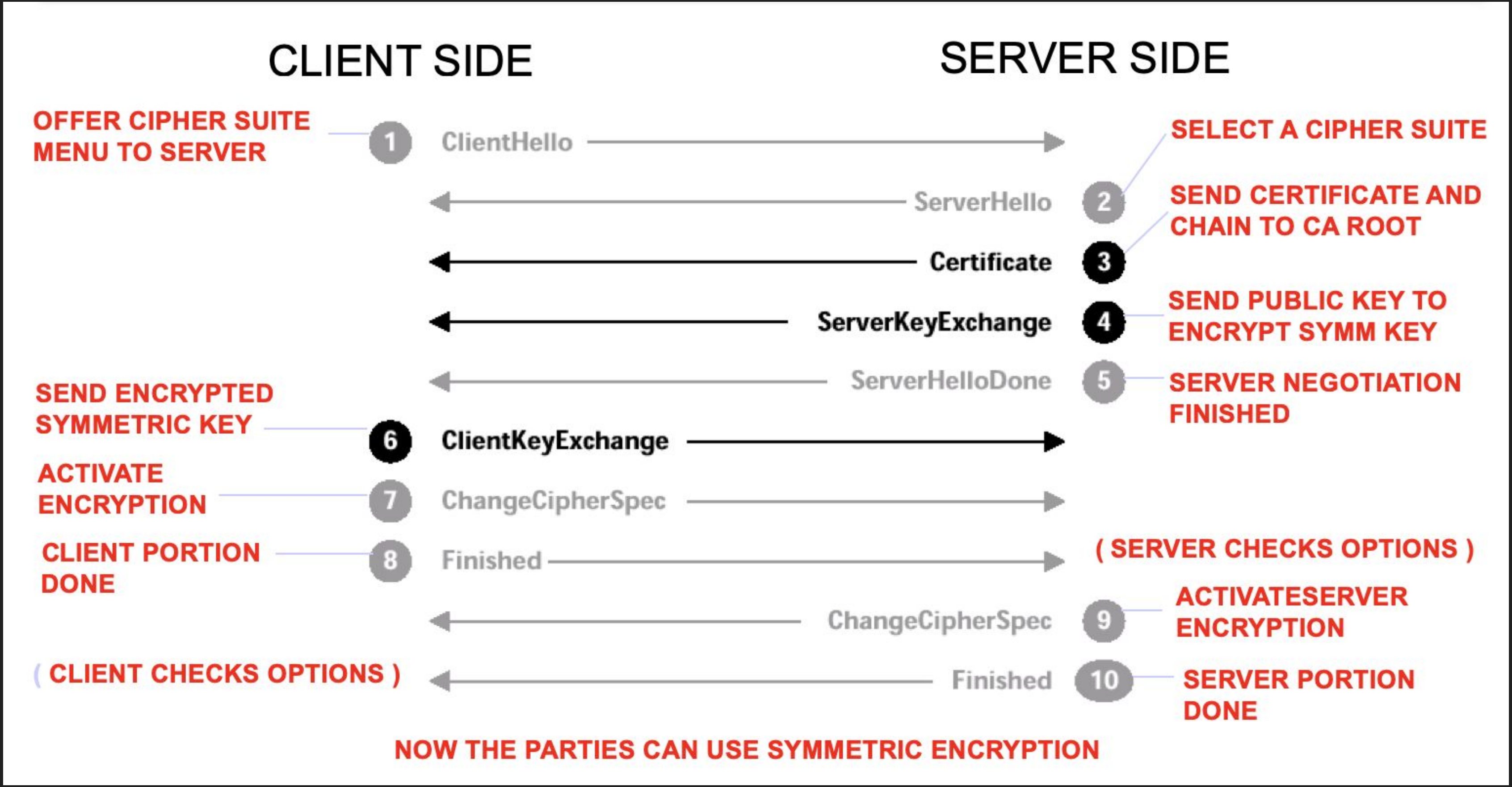
- But how does our system know which Root Certificate Authorities it should trust?
    - Root CAs need to be pre-loaded into browsers, operating systems, etc and be continually updated via secure means as they expire

- So how do we get issued a certificate from a Certificate Authority?

How to get issued a Certificate

- For this class, we’ll be covering three methods for getting issued a SSL Cert
- Lets Encrypt
  - LE is a free, automated and open Certificate Authority - started in 2016
  - Only provides Domain Validation (DV) certificates
  - Certs have short expiry periods (usually 90 days)
- Purchasing a Certificate
  - Certs can be purchased from Verisign, DigiCert, GeoTrust, etc - $$$
  - Can provide Owner Validation (OV) and Extended Validation (EV) certificates - higher level of verification and therefore trust - used by large organizations and often required for banking/etc purposes
  - Usually have an expiry of 1-3 years
- Self Signing
  - You can make your own CA and sign your own certs!
  - Useful for internal systems within an organization, you’ll need to handle distributing the CA and ensuring systems are configured to trust it

SSL/TLS Handshake


CLIENT SIDE
SERVER SIDE
OFFER CIPHER SUITE MENU TO SERVER
1 ClientHello
SELECT A CIPHER SUITE
ServerHello 2
SEND CERTIFICATE AND CHAIN TO CA ROOT
Certificate 3
ServerKeyExchange 4
SEND PUBLIC KEY TO ENCRYPT SYMM KEY
ServerHelloDone 5
SERVER NEGOTIATION FINISHED
SEND ENCRYPTED SYMMETRIC KEY
6 ClientKeyExchange
ACTIVATE ENCRYPTION
7 ChangeCipherSpec
CLIENT PORTION DONE
8 Finished
(SERVER CHECKS OPTIONS)
ChangeCipherSpec 9
ACTIVATESERVER ENCRYPTION
(CLIENT CHECKS OPTIONS)
Finished 10
SERVER PORTION DONE
NOW THE PARTIES CAN USE SYMMETRIC ENCRYPTION

SSL/TLS Handshake

- As shown in the previous diagram, the asymmetric keys and certificate infrastructure are in place to verify the server you are talking to is who they claim to be, and to securely exchange a symmetric session key to use for the rest of the communication

- Why do you think this is the case instead of continuing to use the Public and Private Keys for encryption?
  - Speed/Performance!

- What can happen when a certificate expires or is revoked?
  - How can we prevent this from occurring?

Questions?