SSL/TLS & IPsec

Dr. Anwar Shah
Assistant Professor,
Department of Computer Science,
FAST National University of Computer and Emerging Sciences

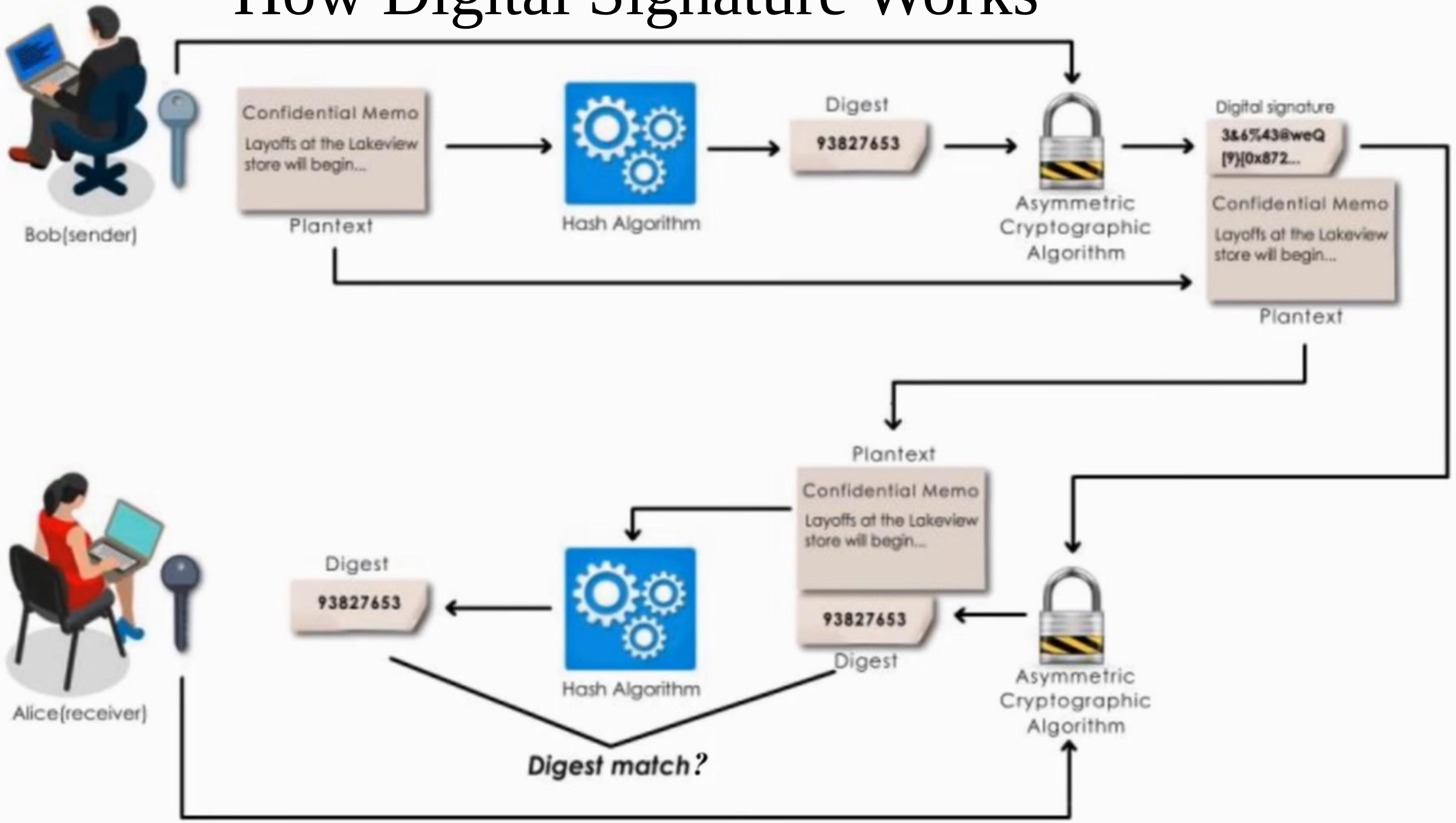
Digital Signature

- A digital signature is equivalent to a handwritten signature in paper. It is an electronic verification of the sender.
- A digital signature serves three basic purposes.

  - Authentication
  - Non-repudiation
  - Integrity

Keep in mind, digital signature is not about encrypting document, just like paper-based signature.

How Digital Signature Works



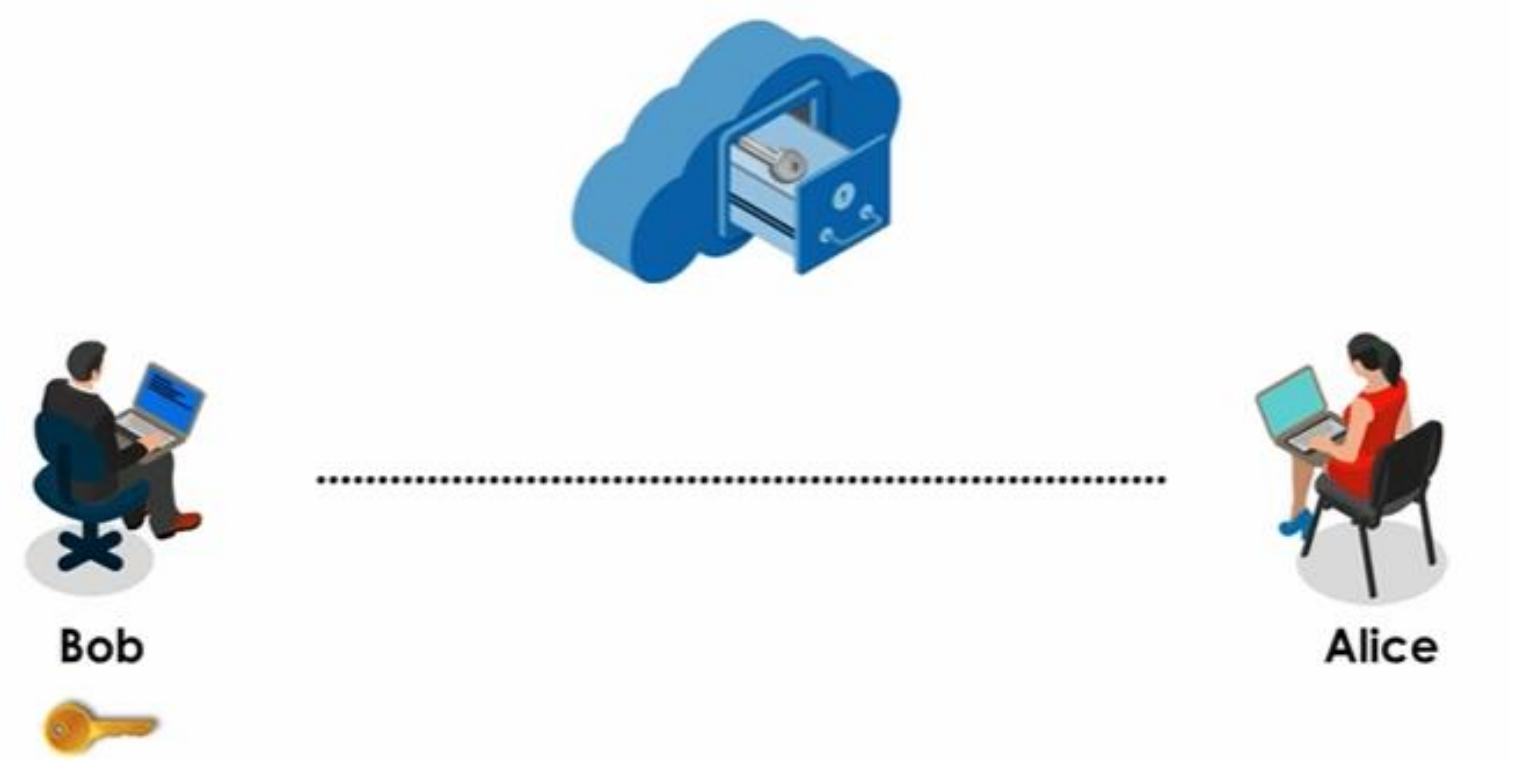
Weakness of Digital Signature

- Digital signature lacks authentication! (Anyone can pretend he is Bob.)

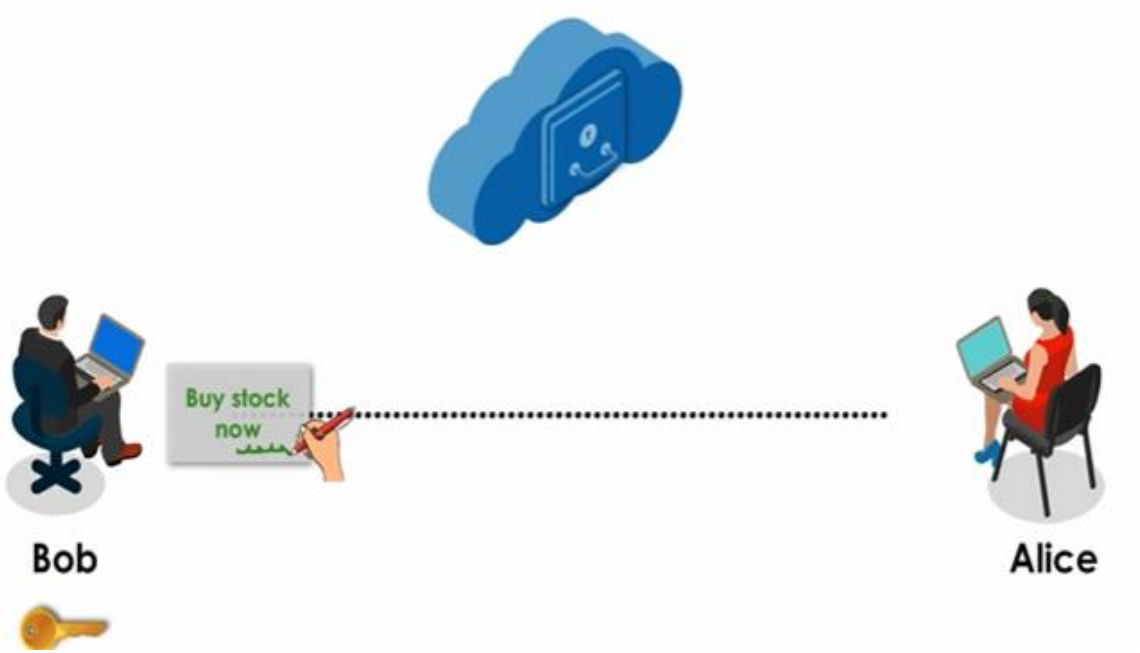
- **Man-in-the-middle attack**

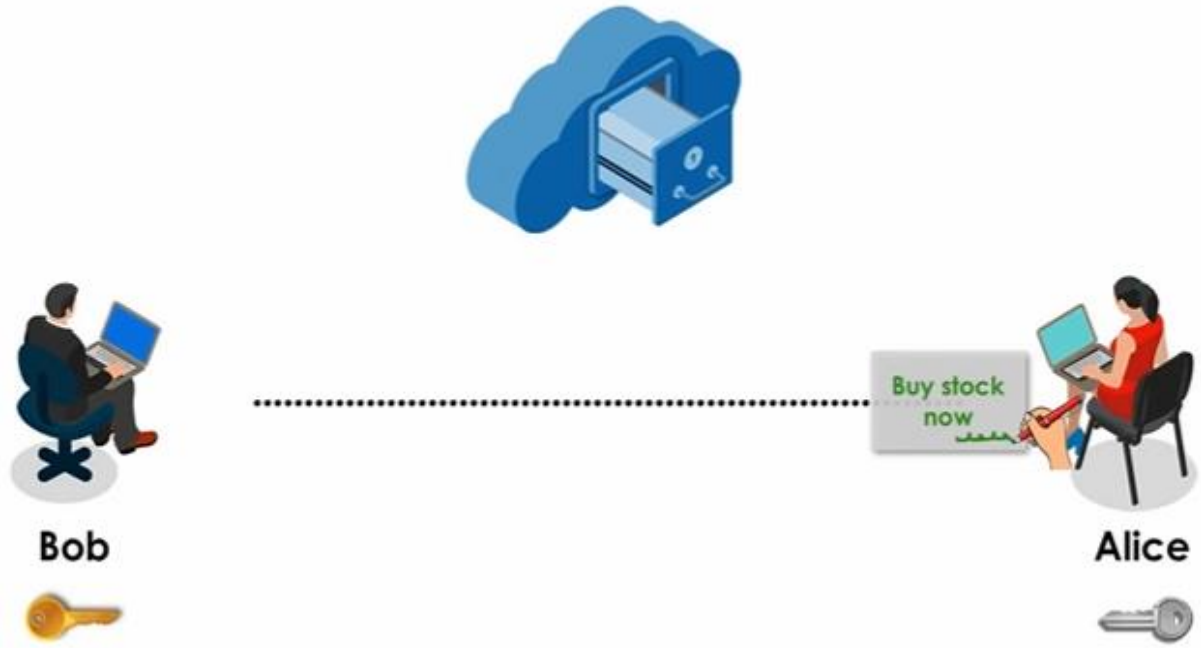
Man-in-the-middle attack

How to verify that the document with the signature is really sent by the desired sender

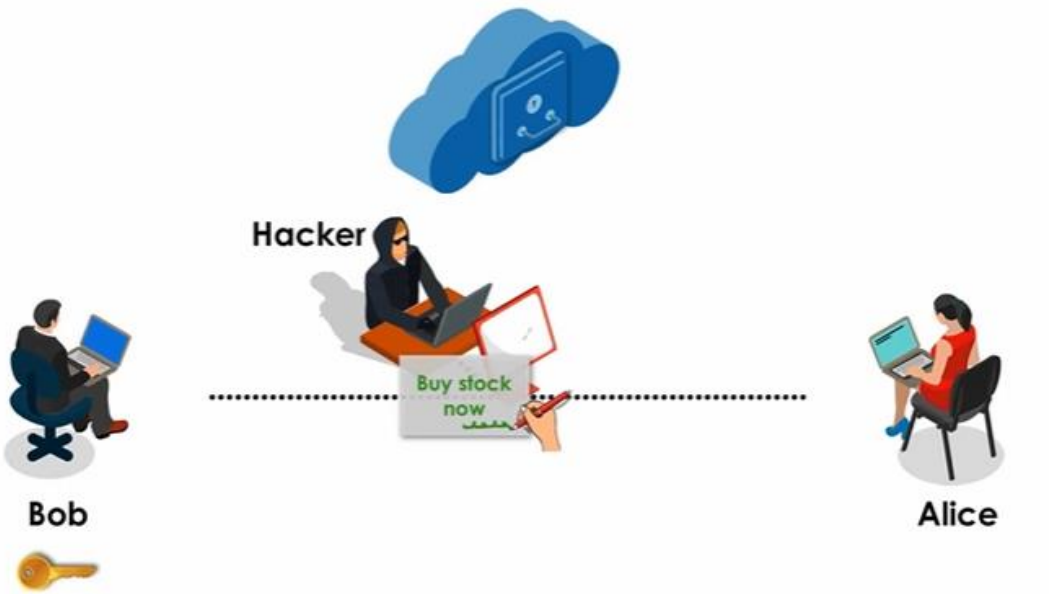


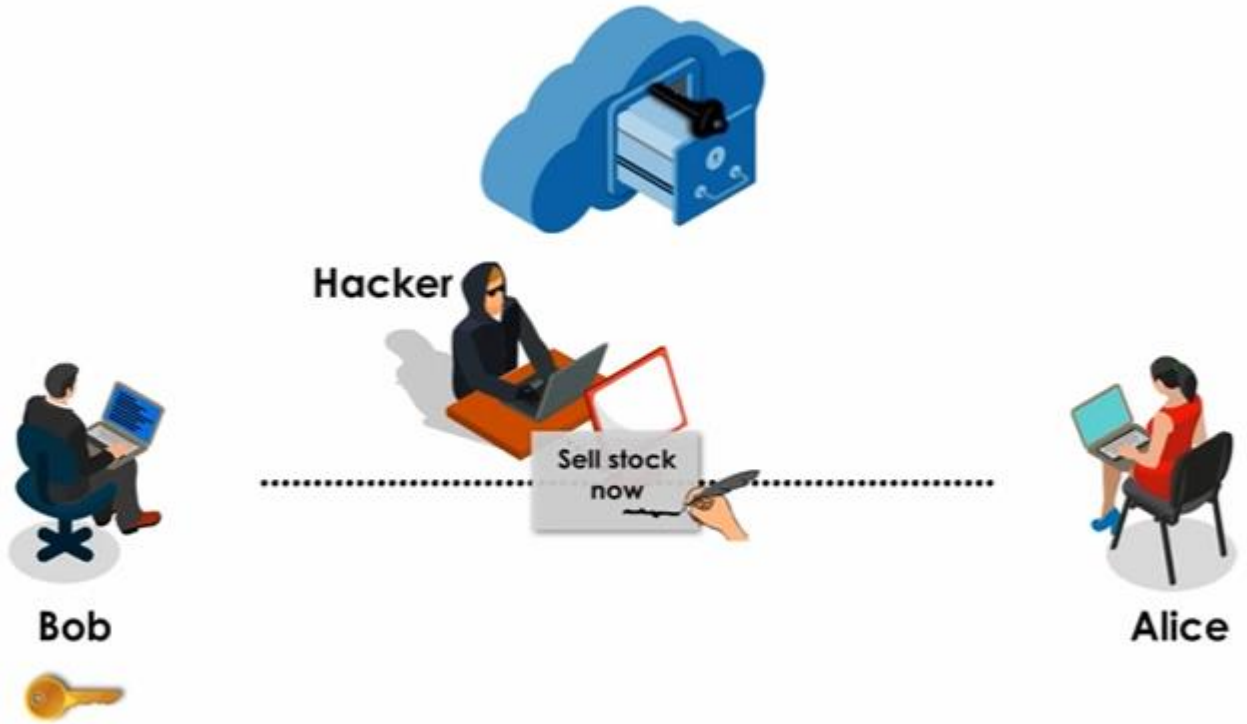
Man-in-the-middle attack



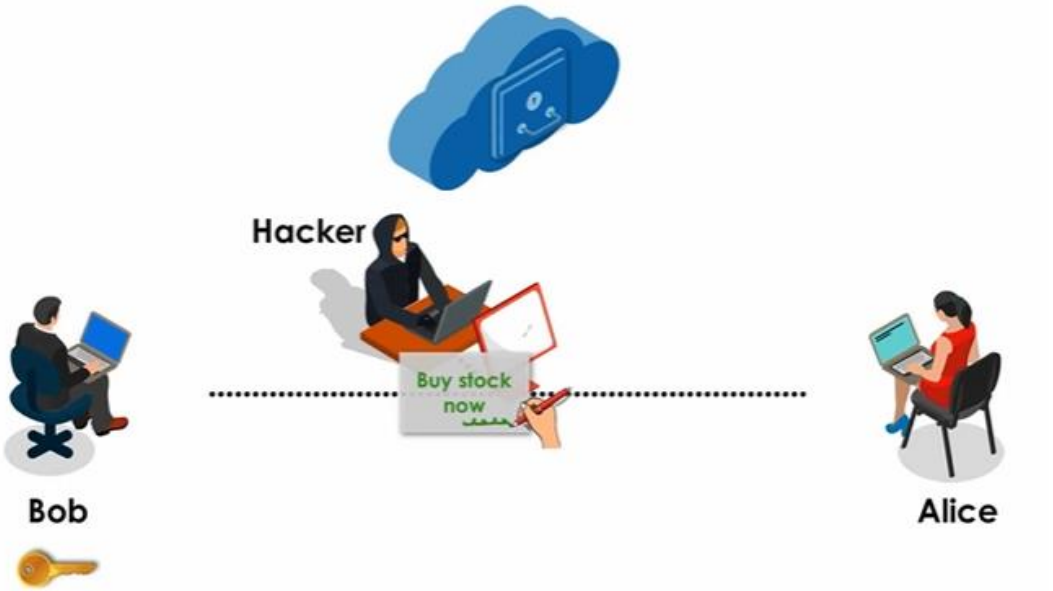


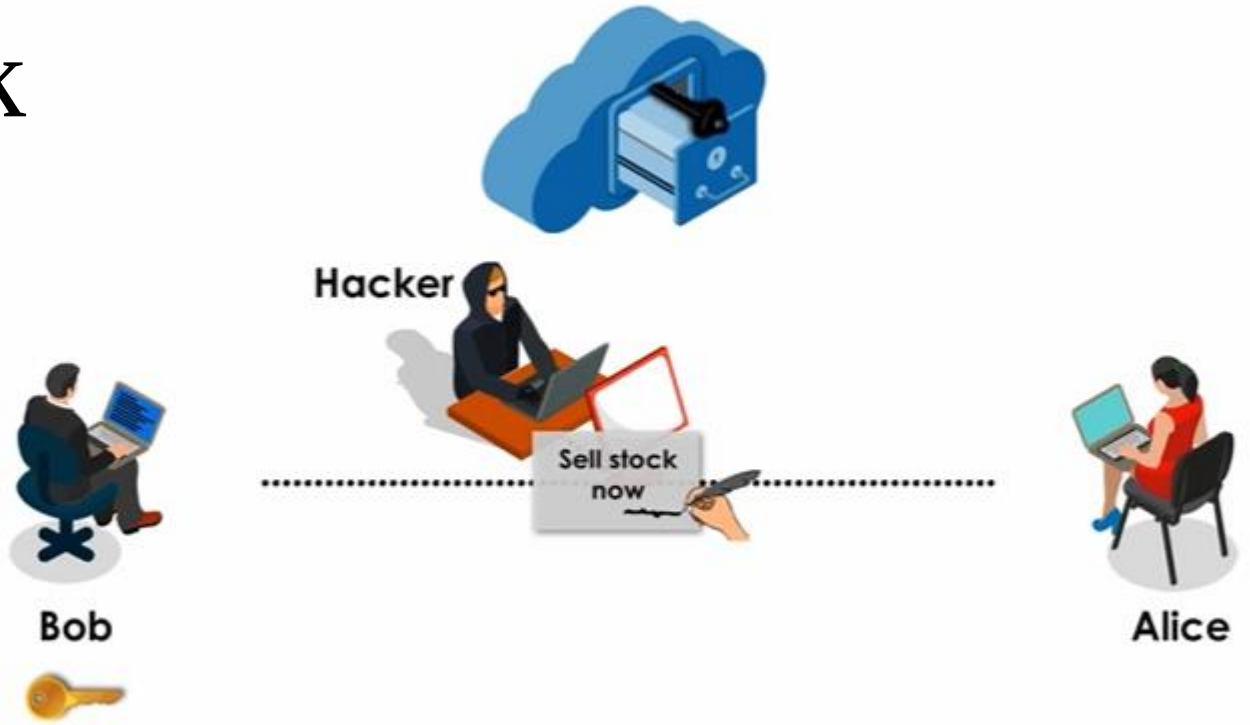
Man-in-the-middle attack



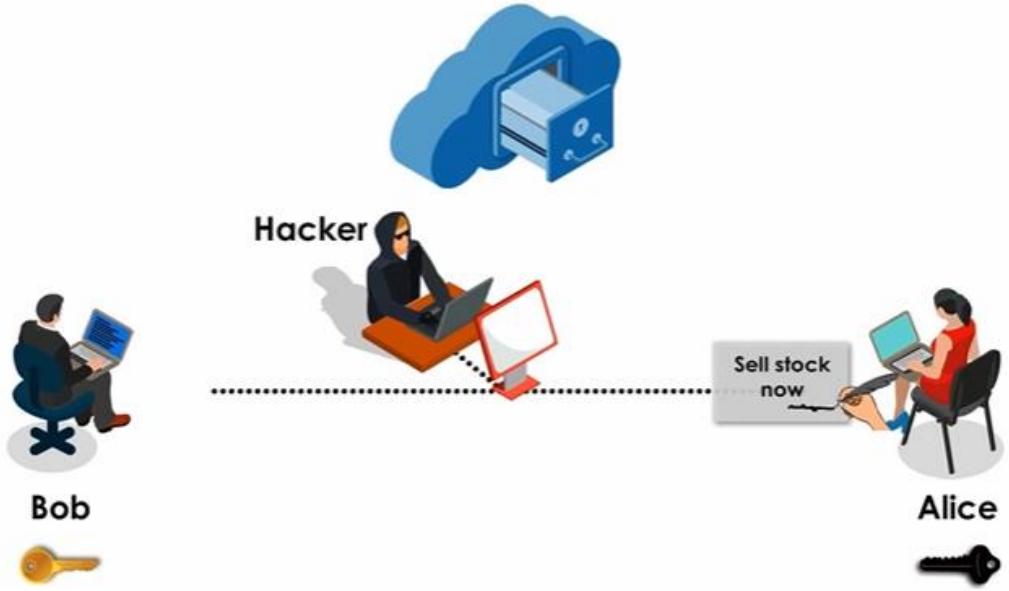


Man-in-the-middle attack





So we need digital certificate!



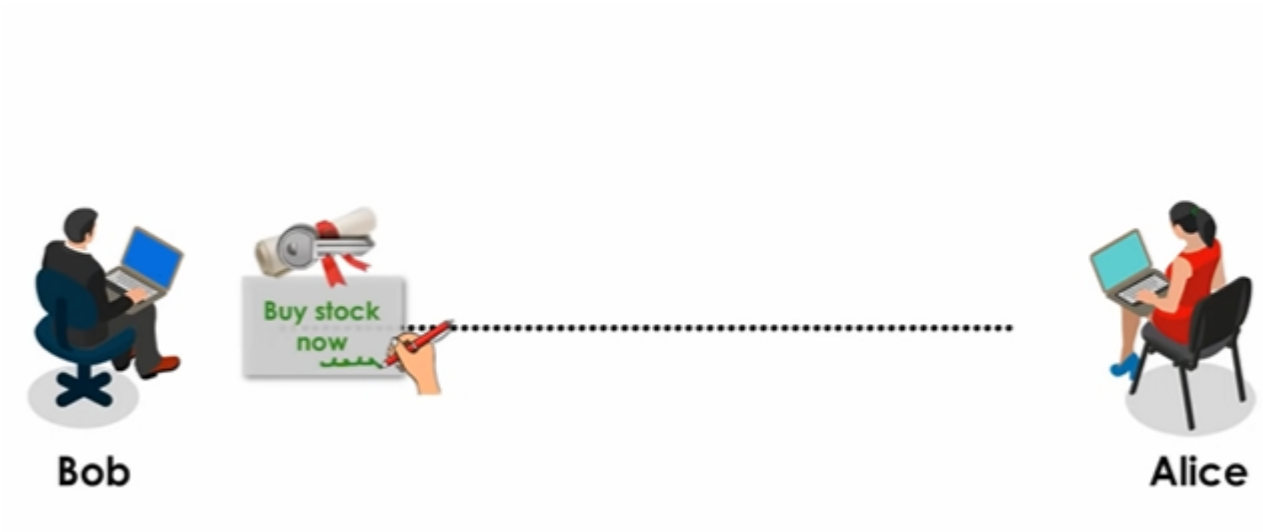
Digital Certificate

- Digital certificates are electronic credentials issued by a trusted third party.
- A digital certificate is a cryptographic key pair (public and private key) that is issued by a Certificate Authority (CA) to verify the identity of an entity on the internet. It includes information about the key, the identity of its owner, and the digital signature of the CA.

✓ **Why Do We Need Digital Certificate**

- Because digital certificate verifies not only the identity of the owner, but also that the owner owns the public key.
- Digital certificate verifies the digital signature is truly signed by the claimed signer.
- **Authentication:** Certifies the identity of the entity, ensuring it is who it claims to be.
- **Integrity:** Ensures that the information in the certificate has not been altered.
- **Confidentiality:** Facilitates secure communication by enabling encryption.

Digital Certificate



Digital Certificate

- A certificate has information
- Certificate owner name
- Owner public key and its expiration
- Certificate issuer name i.e., certificate authority
- The certificate issuer digital signature
- .

Digital Certificate

Digital Certificate Issuing Authorities

- DigiCert
- Symantec (now part of DigiCert)
- Comodo (now Sectigo)
- GlobalSign
- Trustwave.
- Let's Encrypt
- GoDaddy
- Entrust Datacard
- Thawte (now part of DigiCert)
- GeoTrust (now part of DigiCert)

HTTP, HTTPS, SSL, TLS

- HTTP, or Hypertext Transfer Protocol, is widely used for viewing web pages but transfers information in clear text, making it vulnerable to hackers.

- HTTPS, or Secure Hypertext Transfer Protocol, was developed to encrypt data transferred over the internet, securing sensitive information from potential interception by hackers.

- Secure HTTP uses encryption algorithms, making the data impossible to read, and is indicated by an 'S' added to the HTTP in the web address, along with a padlock symbol in the address bar.

HTTP, HTTPS, SSL, TLS

- Secure HTTP, including HTTPS, protects data using protocols like SSL (Secure Sockets Layer) or TLS (Transport Layer Security), ensuring authentication, encryption, and secure data exchange between computers and servers.

- SSL authenticates a website using a digital certificate, establishing trust between the computer and the server, and encrypts data during exchange.

- TLS, the successor to SSL, is an industry-standard cryptographic protocol that authenticates and encrypts data, providing security for online communication.

HTTP, HTTPS, SSL, TLS

- Many websites use secure HTTP by default, driven in part by Google's practice of flagging non-SSL protected sites as "not secure" and impacting their search rankings.

- SSL certificates, available from providers like GoDaddy, are essential for website security and to comply with the growing trend of using secure HTTP.

HTTP, HTTPS, SSL, TLS

- **SSL:**
  - **Introduction:** SSL was developed by Netscape in the mid-1990s.
  - **Purpose:** Initially designed to provide secure communication over the insecure internet, ensuring confidentiality and integrity.
  - **Versioning:** SSL 1.0, SSL 2.0, SSL 3.0 (with SSL 3.0 being the first widely adopted version).
- **TLS:**
  - **Introduction:** TLS was introduced as an improvement to SSL by the Internet Engineering Task Force (IETF).
  - **Purpose:** Addressed vulnerabilities in SSL, enhancing security and flexibility.
  - **Versioning:** TLS 1.0, TLS 1.1, TLS 1.2, TLS 1.3 (TLS 1.2 being widely used, and TLS 1.3 being the latest, emphasizing security improvements).

Notes

- HTTP, secure HTTP, and SSL.
- HTTP stands for Hypertext Transfer Protocol.
- It is probably the most widely used protocol in the world today.
- HTTP is the protocol that is used for viewing web pages on the internet. So, when you type in a web address, like google.com, you'll notice that HTTP is automatically added at the beginning of the web address. And this indicates that you are now using HTTP to retrieve this web page.
- In standard HTTP, all the information is sent in clear text. So, all the information that is exchanged between your computer and that web server, which includes any text that you type on that website, that information is transferred over the public internet. And because it's transferred in clear text, it's vulnerable to anybody who wants it, such as hackers.
- Now normally this would not be a big deal if you were just browsing regular websites and no sensitive data such as passwords or credit card information are being used. But if you were to type in personal sensitive data, like your name, address, phone number, passwords, or credit card information, that sensitive data goes from your computer and then it has to travel across the public internet to get to that web server and this makes your data vulnerable because a hacker that somewhere on the internet can listen in as that data is being transferred and steal your information. So, the hacker is stealing personal information as it's traveling over the internet. Hence this is a problem as far as security.
- This is why HTTPS was developed. HTTPS stands for Secure Hypertext Transfer Protocol. And this is HTTP with a security feature. Secure HTTP encrypts the data that being retrieved by HTTP. SECURE HYPERTEXT TRANSFER PROTOCOL (HTTPS) ensures that all the data that's being transferred over the internet between computers and servers, is secure by making the data impossible to read.
- It does this by using encryption algorithms to scramble the data that's being transferred. So, for example if you were to go to a website that requires you to enter personal information, such as passwords or credit card numbers, you will notice that an 'S' will be added to the HTTP in the web address. And this 'S' indicates that you are now using secure HTTP and have entered a secure website where sensitive data is going to be passed and that data is going to be protected.

Notes

- • In addition to the 'S' being added, a lot of web browsers will also show a padlock symbol in the address bar to indicate that secure HTTP is being used. So, by using secure HTTP, all the data which includes anything that you type, is no longer sent in clear text. It's scrambled in an unreadable form as it travels across the internet. So if a hacker were to try and steal your information he would get a bunch of meaningless data because the data is encrypted and the hacker would not be able to crack the encryption to unscramble the data.
- • Now secure HTTP protects the data by using one of two protocols. And one of these protocols is SSL. SSL or Secure Sockets Layer, is a protocol that’s used to ensure security on the internet. It uses public key encryption to secure data. So basically, this is how SSL works.
- • When a computer connects to a website that's using SSL, the computer's web browser will ask the website to SSL SECURE SOCKETS LAYER identify itself. Then the web server will send the computer a copy of its SSL certificate. An SSL certificate is a small digital certificate that is used to authenticate the identity of a website. Basically, it's used to let your computer know that the website you're visiting is trustworthy.
- • So, then the computer’s browser will check to make sure that it trusts the certificate. And if it does, it will send a message to the web server. Then after the web server will respond back with an acknowledgment and then an SSL session can proceed. Then after all these steps are complete, encrypted data can now be exchanged between your computer and the web server.
- • The other protocol that secure HTTP can use is called TLS. TLS or transport layer security is the latest industry standard cryptographic protocol. It is the successor to SSL and based on the same specifications. And like SSL, it also authenticates the server, client, and encrypts the data.
- • It's also important to point out, that a lot of websites are now using secure HTTP by default on their websites regardless if sensitive data is going to be exchanged or not. And a lot of this has to do with Google. Because Google is now flagging websites as not secure if they are not protected with SSL. And if a website is not SSL protected, Google will penalize that website in their search rankings. So that's why now if you go to any major website, you'll notice that secure HTTP is being used rather than standard HTTP. You can buy SSL certificate for your website from Godaddy etc.

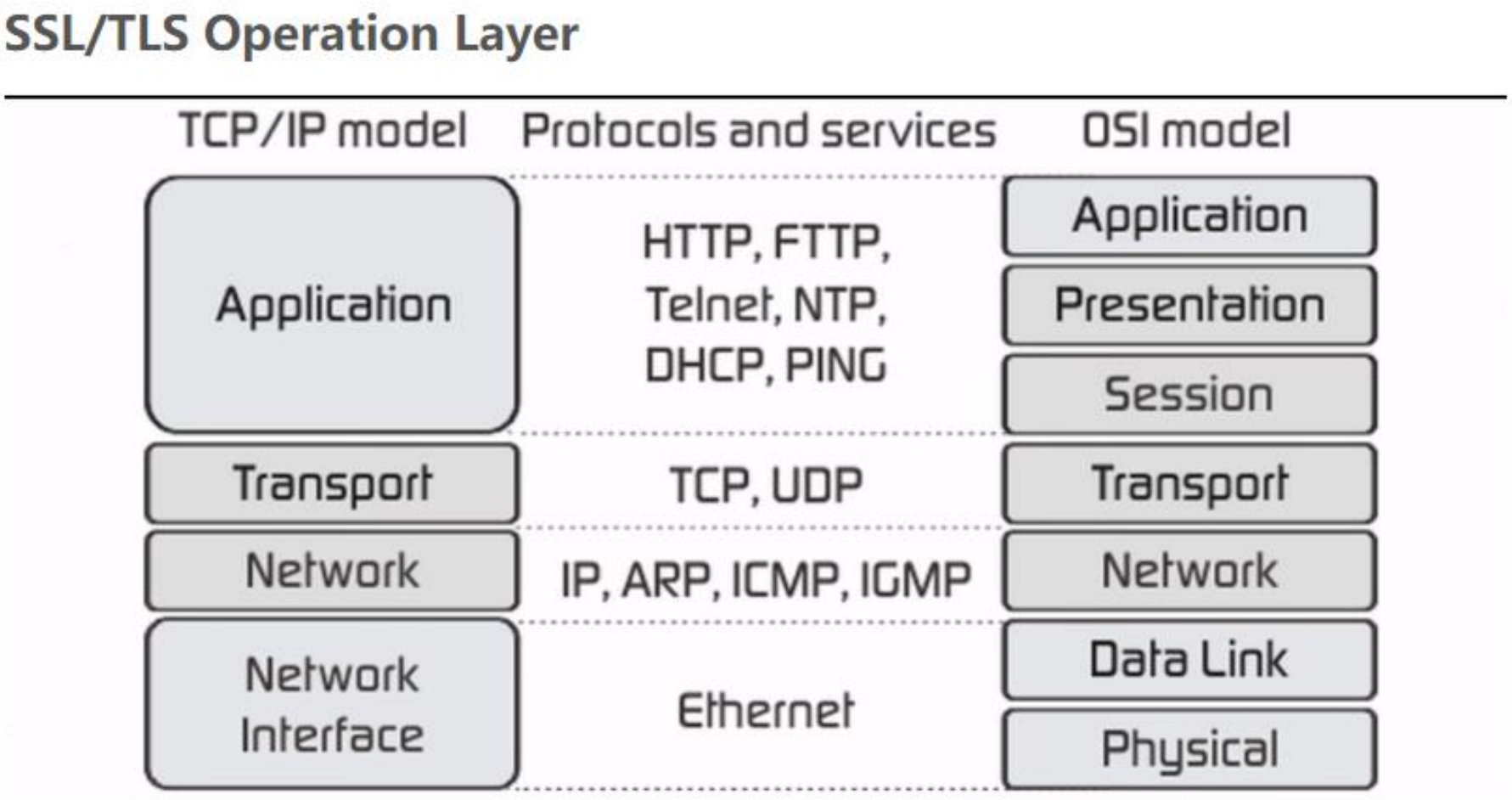
How SSL Work?

- Secure Sockets Layer SSL and is newer version transport layer security TLS are cryptographic protocols that provide security on the internet.
- It is basically used between a web client and web server to establish trust and they negotiate what a secret key should be used to encrypt and decrypt. The conversation with these protocol in place an eavesdropper can only see the connection endpoints but it cannot read or modify any of the actual data, thus it can protect the user's personal data and ensure a safe transaction.

On which layer of OSI model does the protocol operate?

- It is on the application layer because is just HTTP over Secure Sockets Layer.
- It is on presentation layer because encryption and decryption operate on this layer.
- It is on session layer because the protocol provides point-to-point session security
- It is indeed a gray area and each argument is valid. The OSI model is not a science and is only a guideline.
- The handshake protocol is involved with the top three layers OSI model and if we use tcp/ip model we can simply say the protocol belongs to the application layer.

How SSL Work?


SSL/TLS Operation Layer
TCP/IP model
Protocols and services
OSI model
Application
HTTP, FTTP,
Telnet, NTP,
DHCP, PING
Application
Presentation
Session
Transport
TCP, UDP
Transport
Network
IP, ARP, ICMP, IGMP
Network
Network
Interface
Ethernet
Data Link
Physical

How SSL Work?

- HTTPS is the secure version of HTTP. A protocol used between the browser and web server. The s at the end of HTTP stands for secure. Technically it refers to HTTP over secure socket layer or SSL.

How SSL Work? (SSL/TLS handshake Protocol)

- Process of how a client and a server uses a handshake protocol to negotiate how to securely exchange data.
- Step 1 Client browser requests secure pages HTTPS from Yahoo web server.
- The client sends hello message that contain information such as SSL/TLS version and the cryptographic algorithms and a data compression methods supported by the client.

Step 1:

How SSL Work?

- Step 2 The Yahoo server sends its a public key with his SSL certificate which is digitally signed by a third party or we call certificate authority or simply CA.

How SSL Work?

- Server responds with server hello message that contains cryptographic algorithms choosing by the server from the list provided by the client, the session ID and the server's digital certificate and it is a public key.

How SSL Work?

- Step 3 once the client browser gets the certificate it will check the issuers digital signature to make sure the certificate is valid as we know a digital signature is created by CA'S private key and the client browser either Chrome or Firefox is previously installed with many major CA's public keys. Thus, digital signature can be verified. Once the certificate is signature is verified, the detailed certificate can be trusted and a green padlock icon appears in the address bar. The green catalog simply indicates that web servers public key really belongs to the web server not someone else.
- The verification is done.
- This step is basically establishing trust on the web server once the client trusts the web server it will take step 4 client key exchange.

How SSL Work?

- Step 3

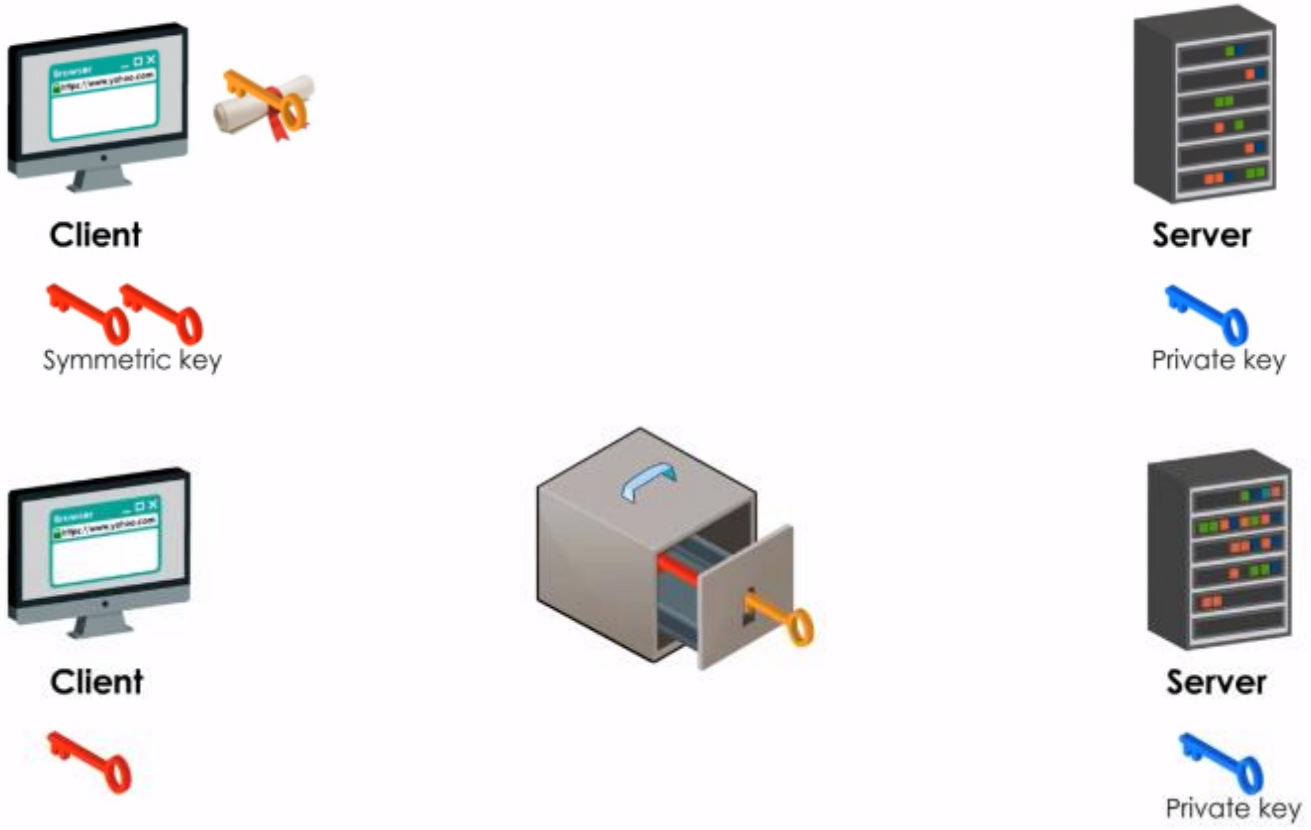
Issued To	Issued By	Expiratio...	Friendly Name
AAA Certificate Ser...	AAA Certificate Services	12/31/2028	COMODO CA
AddTrust External ...	AddTrust External CA...	5/30/2020	The USERTrust ...
AffirmTrust Comme...	AffirmTrust Commercial	12/31/2030	Trend Micro
Baltimore CyberTru...	Baltimore CyberTrust ...	5/12/2025	DigiCert Baltimor...
Certum CA	Certum CA	6/11/2027	Certum
Certum Trusted Ne...	Certum Trusted Netw...	12/31/2029	Certum Trusted ...
Class 3 Public Prima...	Class 3 Public Primary ...	8/1/2028	VeriSign Class 3 ...
COMODO RSA Cert...	COMODO RSA Certific...	1/18/2038	COMODO SECU...
Copyright (c) 1997 ...	Copyright (c) 1997 Mi...	12/30/1999	Microsoft Timest...

Certificates in browser

Padlock Appeared

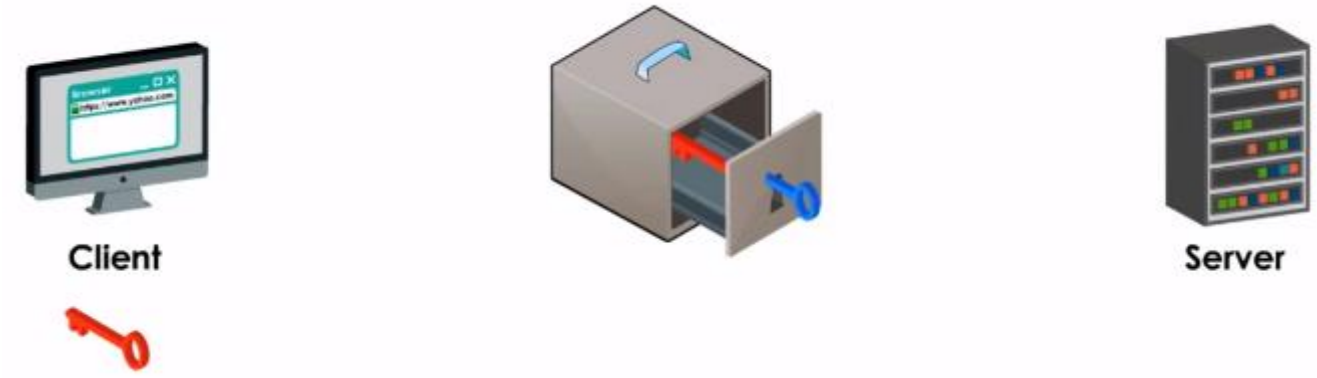
How SSL Work?

- Step 4 client browser creates one symmetric key or shared secret. It keeps one and give a copy to web server, however, client browser does not want to send the secret in plaintext. Therefore it uses the web servers public key to encrypt the secret and then sends it to the web server.



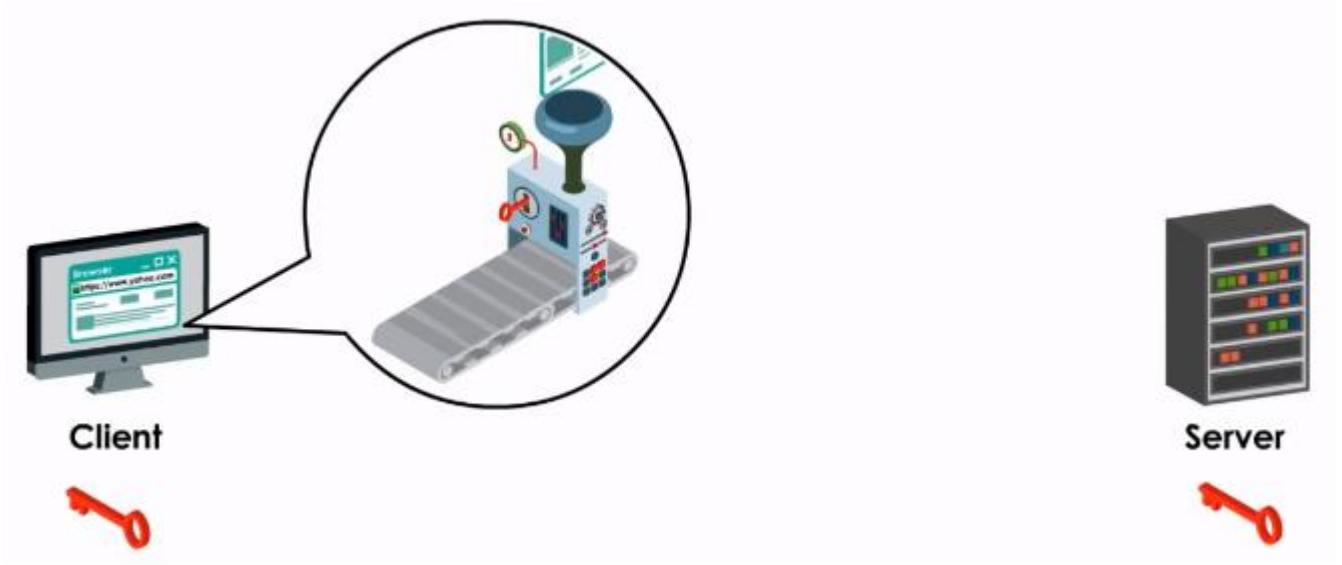
How SSL Work?

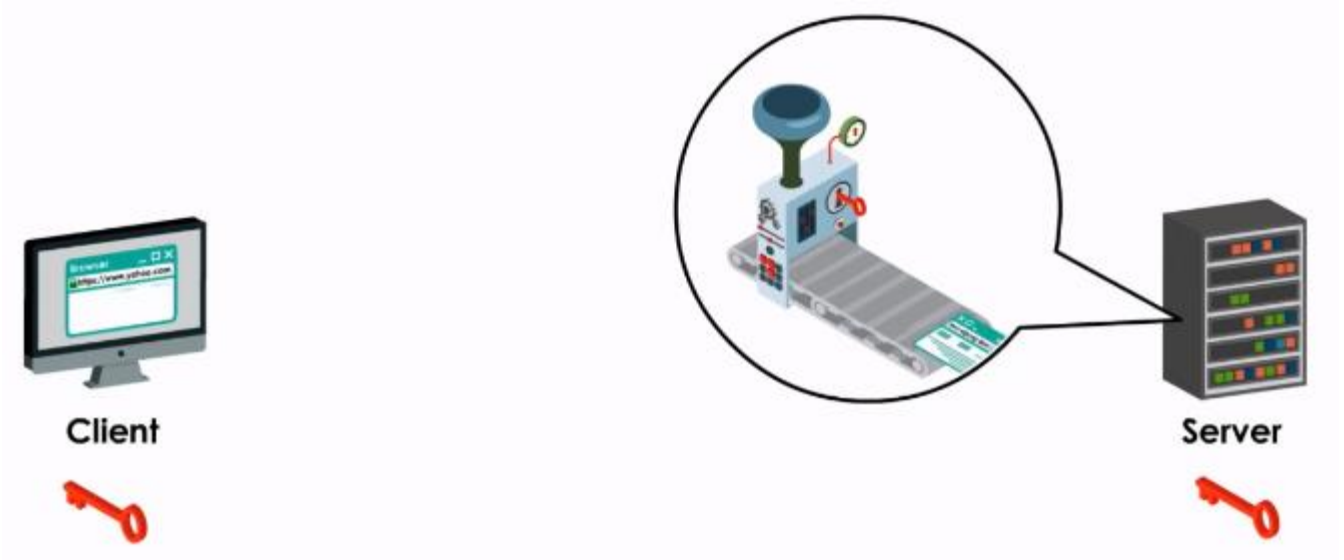
- Step 5 when the web server gets the encrypted symmetric key it uses private key to decrypt it. Now the web server gets the browser's shared key. From now on, all traffic between the client and a web server will be encrypted and decrypted with the same key, the symmetric key.



How SSL Work?

- Step 5





How SSL Works?

- The above process demonstrates how asymmetric key algorithm and symmetric key algorithm work together.
- Asymmetric key algorithm (public key & private key) is used to verify the identity of the owner and its public key so that trust is built.
- Once the connection is established, Symmetric key algorithm (shared key) is used to encrypt and decrypt all traffic between them.
- The green padlock: indicates that the web server's public key really belongs to the web server, not someone else.
- The Https and the green padlock only indicates the communications between client and server are encrypted. It does not says the website is "safe and good".

Notes

What does HTTPS mean? and what's the significant about that a small grain padlock

To answer these two questions, we need to understand SSL certificate and how it works?

HTTPS is the secure version of HTTP. A protocol used between the browser and web server. The s at the end of HTTP stands for secure. Technically it refers to HTTP over secure socket layer or SSL. HTTPS means all communications between your browser and a web server are encrypted. Behind HTTPS, SSL certificate plays an important role in building trust between a browser and a web server. By definition SSL certificate is web server’s digital certificate issued by a third party and verifies the identity of the web server and its public key.

let us consider one example to demonstrate how SSL certificate works.

In the scenario I want to connect with the Yahoo web server and I want all communications with the Yahoo web server are encrypted. I type in HTTP:\\yahoo.com, here is what happens when I hit enter. step 1 my browser requests secure pages HTTPS from Yahoo web server

step 2 the Yahoo server sends its a public key with his SSL certificate which is digitally signed by a third party or we call certificate authority or simply CA

step 3 once my browser gets the certificate it will check the issuers digital signature to make sure the certificate is valid as we know a digital signature is created by CA’S private key and my browser either Chrome or Firefox is previously installed with many major CA’s public keys. Thus, digital signature can be verified. once the certificate is signature is verified, the detailed certificate can be trusted and a green padlock icon appears in the address bar. The green catalog simply indicates that web servers public key really belongs to the web server not someone else. The verification is done.

Now it's time to exchange of a secret that's the step 4.

Step 4 my browser creates one symmetric key or shared secret. It keeps one and give a copy to web server, however, my browser does not want to send the share the secret in plaintext. Therefore it uses the web servers public key to encrypt the secret and then sends it to the web server

Step 5 when the web server gets the encrypted symmetric key it uses private key to decrypt it. Now the web server gets the browser's shared key. From now on, all traffic between the client and a web server will be encrypted and decrypted with the same key, the symmetric key.

In this example we actually demonstrate how asymmetric key algorithm and symmetric key algorithm worked together. Asymmetric key algorithm is used to verify the identity of the owner and is public key so that Trust is built, once the connection is established, symmetric key algorithm is used to encrypt and decrypt all traffic between them. Keep in mind, HTTPS and a green padlock only indicated the communications between client and web server are encrypted, it does not mean the website itself is a safe and a good. Remember anyone including hackers can't launch online business and anyone including hackers can get SSL certificate for their sites. When you are online shopping, you still want to use websites with a good repetition.

It is SSL or TLS Certificate?

- you can colloquially refer to a digital certificate as an SSL certificate, but it's worth noting that the term "SSL certificate" is a bit outdated. SSL (Secure Sockets Layer) was a cryptographic protocol that provided secure communication over a computer network. However, it has been deprecated in favor of its successor, Transport Layer Security (TLS). TLS is the protocol that secures communication on the internet today.
- So, while people commonly use the term "SSL certificate" to refer to digital certificates used in the context of securing websites and online communications, it’s common and accepted practice but it's more accurate to refer to them as TLS certificates. These certificates are essential for establishing a secure connection between a user's web browser and a website's server, ensuring that data transmitted between them is encrypted and secure.

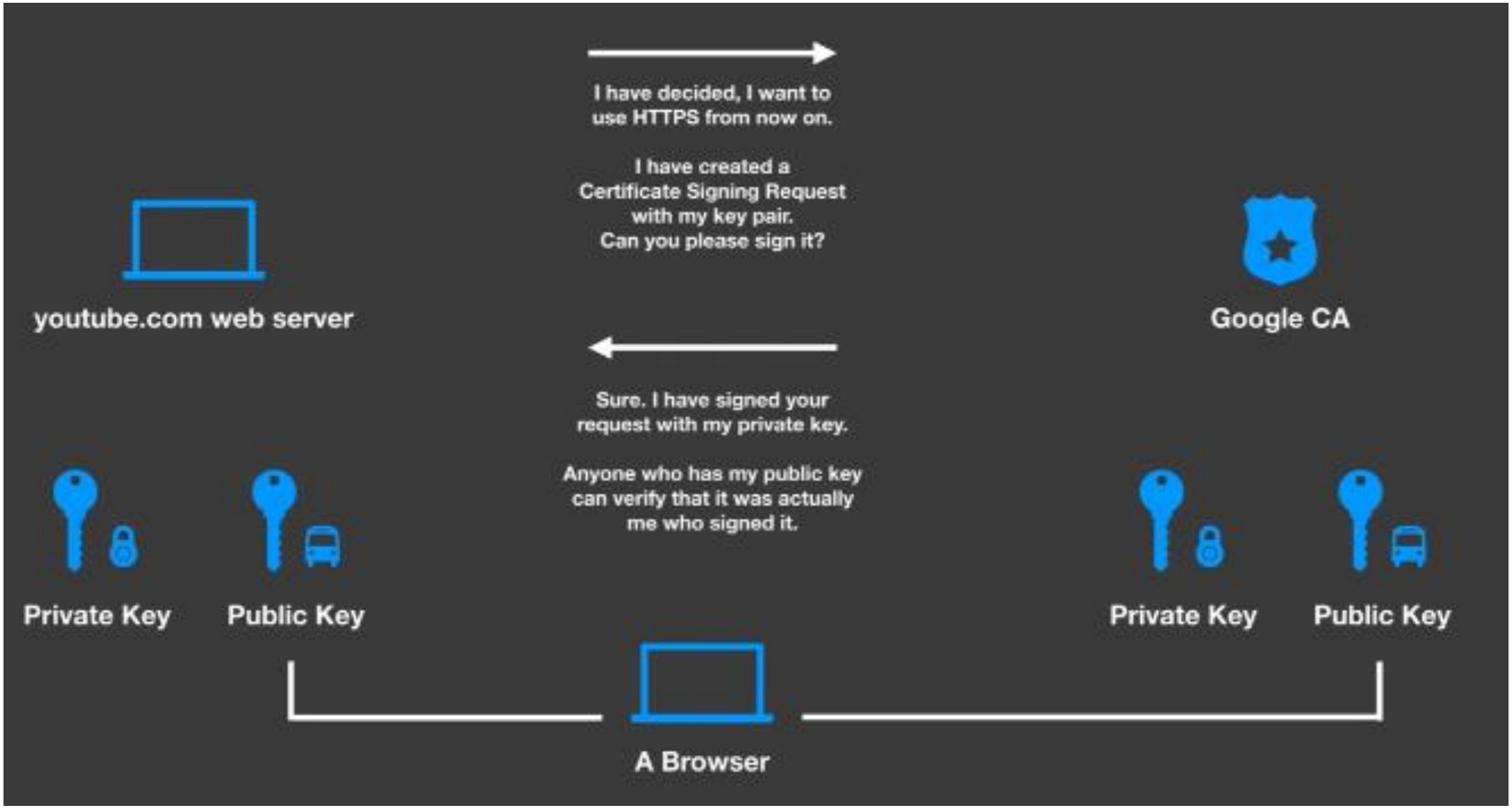
What is a Certificate Authority and How is our Certificate Signed?

- How you make sure that the party you're talking to is actually who they claim they are and this is where the certificate authority comes.
- We had a certificate that was signed by the google certificate authority but how do you get such a certificate how is it signed
- Let the youtube.com web service says hey i've decided i want to use https from now on. I've created a certificate signing request with my key pair, google certificate authority can you please sign it, then the google certificate authority says sure, i've signed your request with my private key. Anyone who has my public key can verify that it was actually me who signed this.

What is a Certificate Authority and How is our Certificate Signed?

- Most browsers when they are delivered already have a list of trusted certificates and these certificates are issued by known certificate authorities. Google is one of them and youtube uses the google one because youtube belongs to google.
- There are also standalone certificate authorities on the internet that you can ask to sign your certificate signing request, the important thing is that you choose one that most browsers trust by default.
- Because when the browser comes into contact with this newly created youtube.com web server even though youtube.com might not have existed when the browser was created. it can now verify that this public key which claims to be signed by google was actually signed by google because it knows google's public key from their certificate and can now reliably determine that the information in this certificate that youtube.com provided is true as long as we trust the google certificate authority.

What is a Certificate Authority and How is our Certificate Signed?



Self Signed Certificates?

- let's imagine you have your own app and it's currently deployed to your staging environment where you want to do some testing so if you say i now also want to encrypt all the communication with this app. You have to create a private key and a public key.
- So, the next step would now be to create a signing request but who will sign this if there is no certificate authority.
- So, in this case, you can create a second pair of keys and basically by having this set of keys, you've just created your very own certificate authority.
- You can now do the same stuff as before you create your certificate signing request and send it to the certificate authority. The authority will make sure that you actually have access to this url that you're claiming and say hey here's your certificate it was signed by me and again anyone with my public key can verify that it was signed by me.

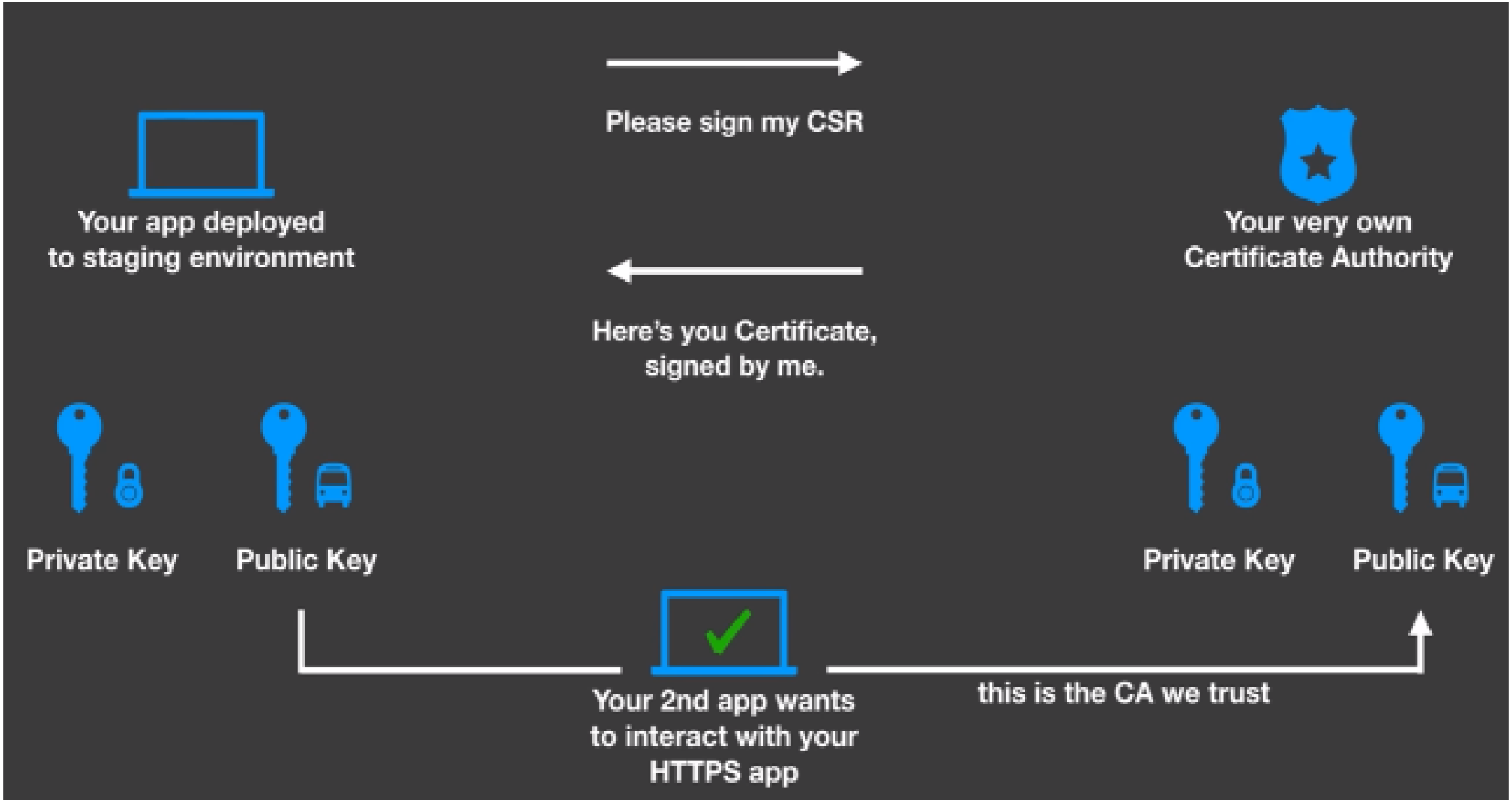
Self Signed Certificates?

- So instead of a browser, this time your second app comes along and says hey i want to interact with the first app that we just deployed to the staging environment, it realizes that your app wants to use https. So it looks at the certificate and finds a public key in there.
- This certificate now says, i was signed by whatever we called our own certificate authority and then of course your second app says this is not a trusted authority. I will not trust this certificate, however, because all of this is in your controlled environment and you do trust the certificate authority that you've just created.
- You can now tell your second app hey if you trust the certificate that i point to this certificate authority that contains this public key then you can use that to validate the public key from my first app that we deploy to staging and then your second app will say oh yeah that matches as long as we trust this certificate authority we can use https here.

Self Signed Certificates?

- The process is really the same as with a certificate that was signed by a known authority but that is effort and there might be limits or there might be cost involved so if you don't intend your app to be used by end users that have browsers that only trust specific certificate authorities then you can use a self-signed certificate at the cost of having to tell the user yes you need to trust this and this is very common in testing environments

Self Signed Certificates?



SSL Limitations and TLS Improvements

Aspect	SSL Limitation	TLS Improvement
Security Protocol	*SSL is considered less secure compared to later versions of TLS.*	*TLS is an improved and more secure successor to SSL.*
Protocol Versions	*SSL has multiple versions (SSL 2.0, SSL 3.0), some of which have known vulnerabilities.*	*TLS offers improved versions, and it's recommended to use the latest TLS version for security.*
Cryptographic Algorithms	*SSL has limited support for cryptographic algorithms, and some are considered weak by modern standards.*	*TLS supports stronger and more secure cryptographic algorithms. It provides better flexibility for choosing encryption algorithms.*
Vulnerabilities	*SSL has been susceptible to various vulnerabilities and attacks, such as POODLE, BEAST, and DROWN.*	*TLS addresses known vulnerabilities present in SSL and introduces mechanisms to mitigate new threats.*
Key Exchange Methods	*SSL supports limited key exchange methods, and some are vulnerable to attacks.*	*TLS introduces improved key exchange methods, including more robust options like Diffie-Hellman key exchange.*
Forward Secrecy	*SSL lacks strong support for forward secrecy, which protects past sessions if long-term secret keys are compromised.*	*TLS emphasizes the use of forward secrecy, enhancing the security of encrypted communications.*
Renegotiation	*SSL renegotiation can be vulnerable to certain attacks.*	*TLS includes improvements in renegotiation procedures to address security concerns.*
Algorithm Flexibility	*SSL has limited flexibility in negotiating cryptographic algorithms during the handshake process.*	*TLS provides greater flexibility in negotiating algorithms, allowing for more robust and adaptable configurations.*
Compatibility	*SSL may face compatibility issues with modern web browsers and applications.*	*TLS is designed to be more compatible with a broader range of applications and platforms.*

TLS Step by Step

- Transport Layer Security (TLS) introduces improved key exchange methods to enhance the security of encrypted communications. The key exchange process is a crucial aspect of establishing a secure communication channel between a client (such as a web browser) and a server.
- How TLS introduces improved key exchange methods:

TLS do three tasks

1. **Authentication:** ensures that the parties exchanging information are who they claim to be.
2. **Encryption:** hides the data being transferred from third parties.
3. **Integrity:** verifies that the data has not been forged or tampered with.

TLS Handshake

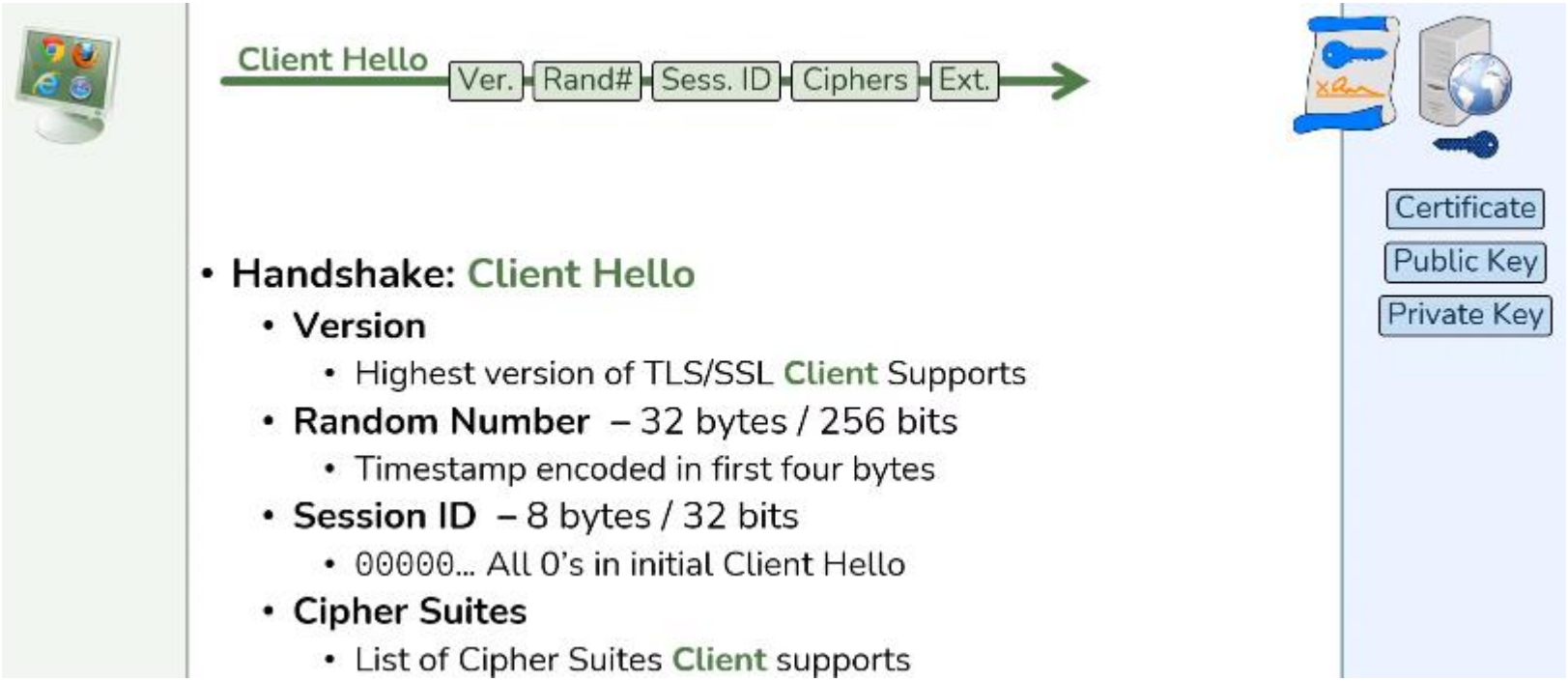
TLS Handshake

- TLS Handshake
  - Create the “protected tunnel” between Client and Server
    - Protects Bulk Data transfer with C I A
  - Handshake will be illustrated on a Record by Record basis
    - Records ≠ Packets

TLS Handshake

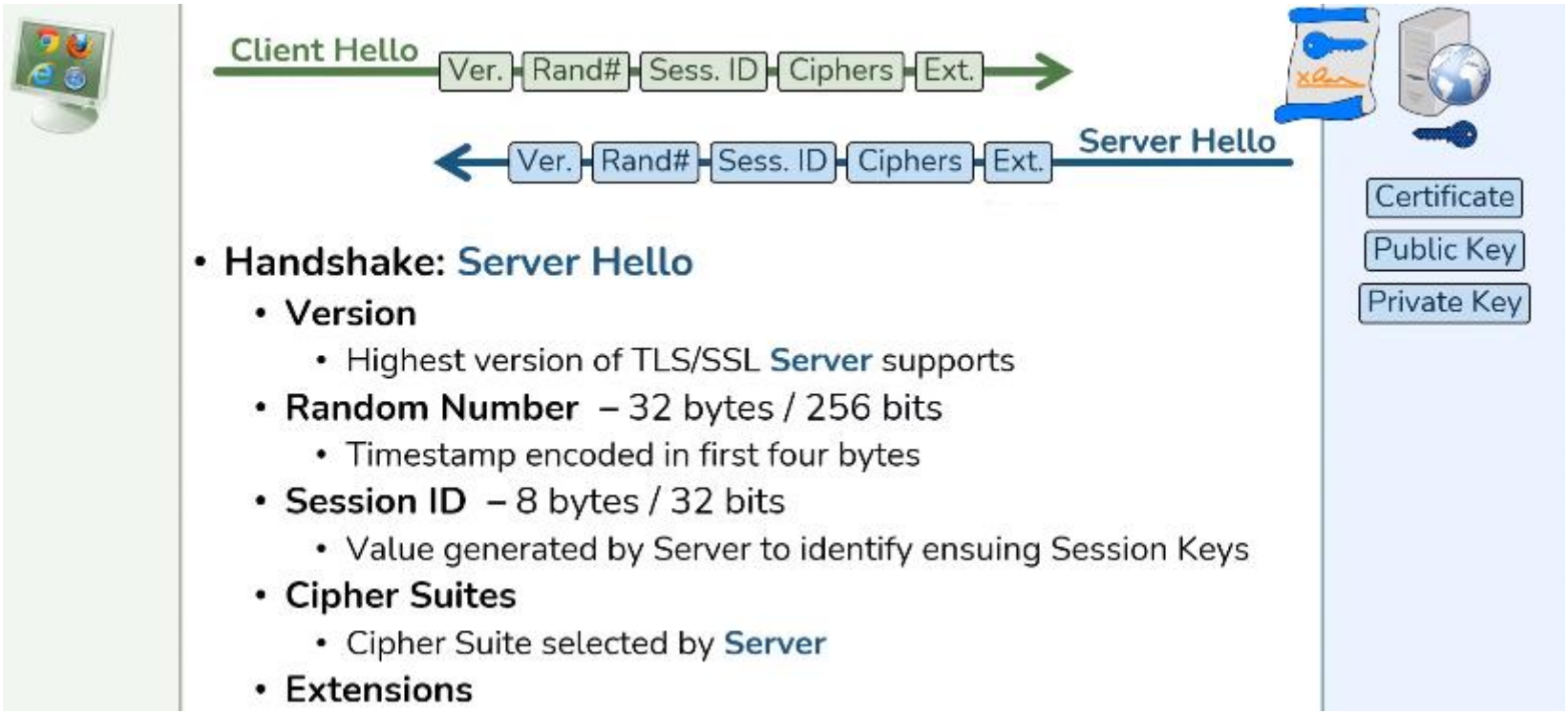
Starting Point

TLS Handshake



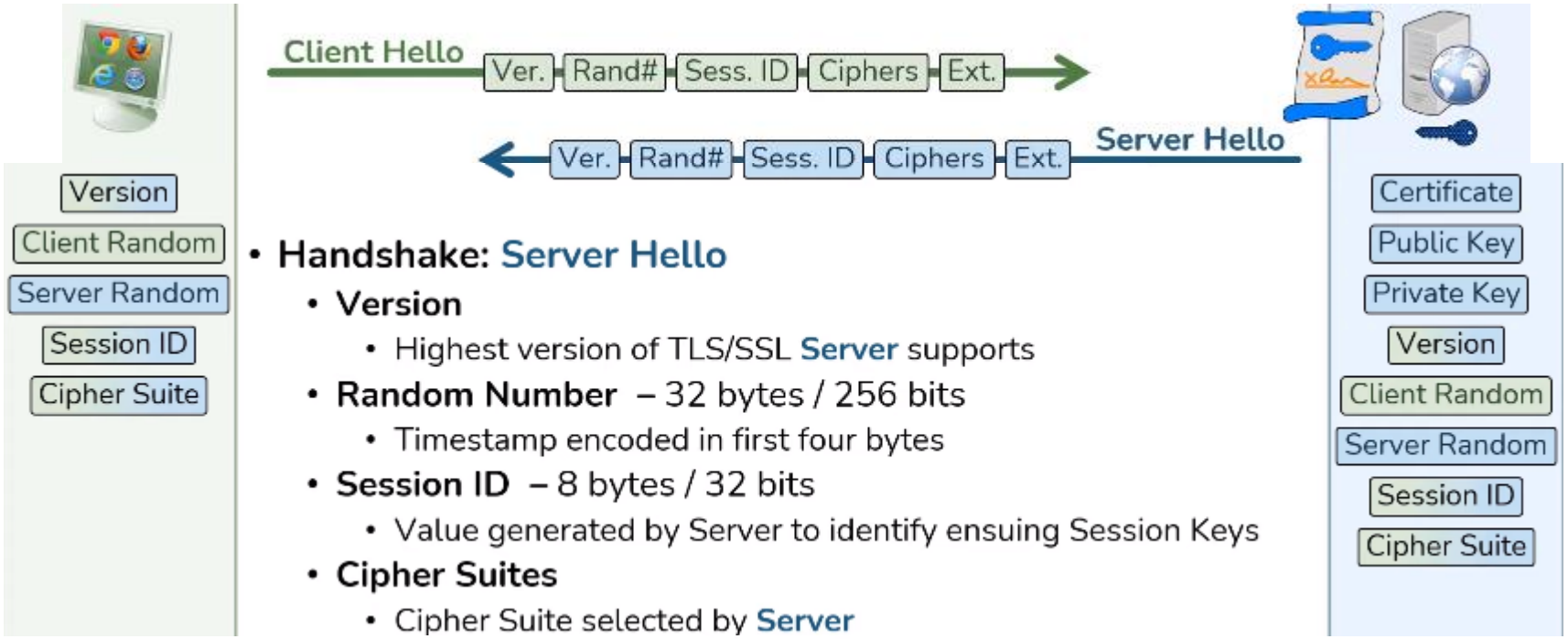
- **Handshake: Client Hello**
  - **Version**
    - Highest version of TLS/SSL **Client** Supports
  - **Random Number** – 32 bytes / 256 bits
    - Timestamp encoded in first four bytes
  - **Session ID** – 8 bytes / 32 bits
    - `00000`... All 0's in initial Client Hello
  - **Cipher Suites**
    - List of Cipher Suites **Client** supports
  - **Extensions**
    - Optional additional features added to TLS/SSL

TLS Handshake



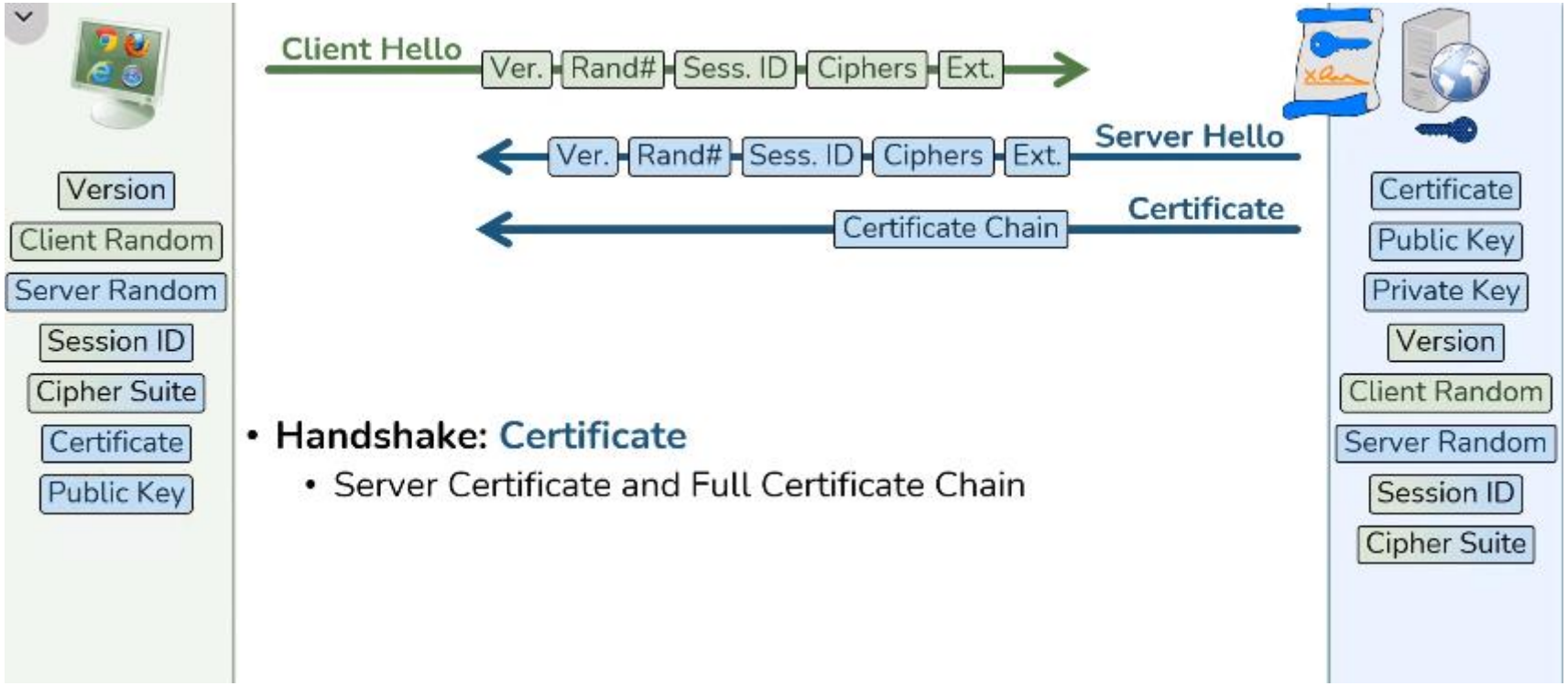
- **Handshake: Server Hello**
  - **Version**
    - Highest version of TLS/SSL **Server** supports
  - **Random Number** – 32 bytes / 256 bits
    - Timestamp encoded in first four bytes
  - **Session ID** – 8 bytes / 32 bits
    - Value generated by Server to identify ensuing Session Keys
  - **Cipher Suites**
    - Cipher Suite selected by **Server**
  - **Extensions**
    - Optional additional features added to TLS/SSL

TLS Handshake

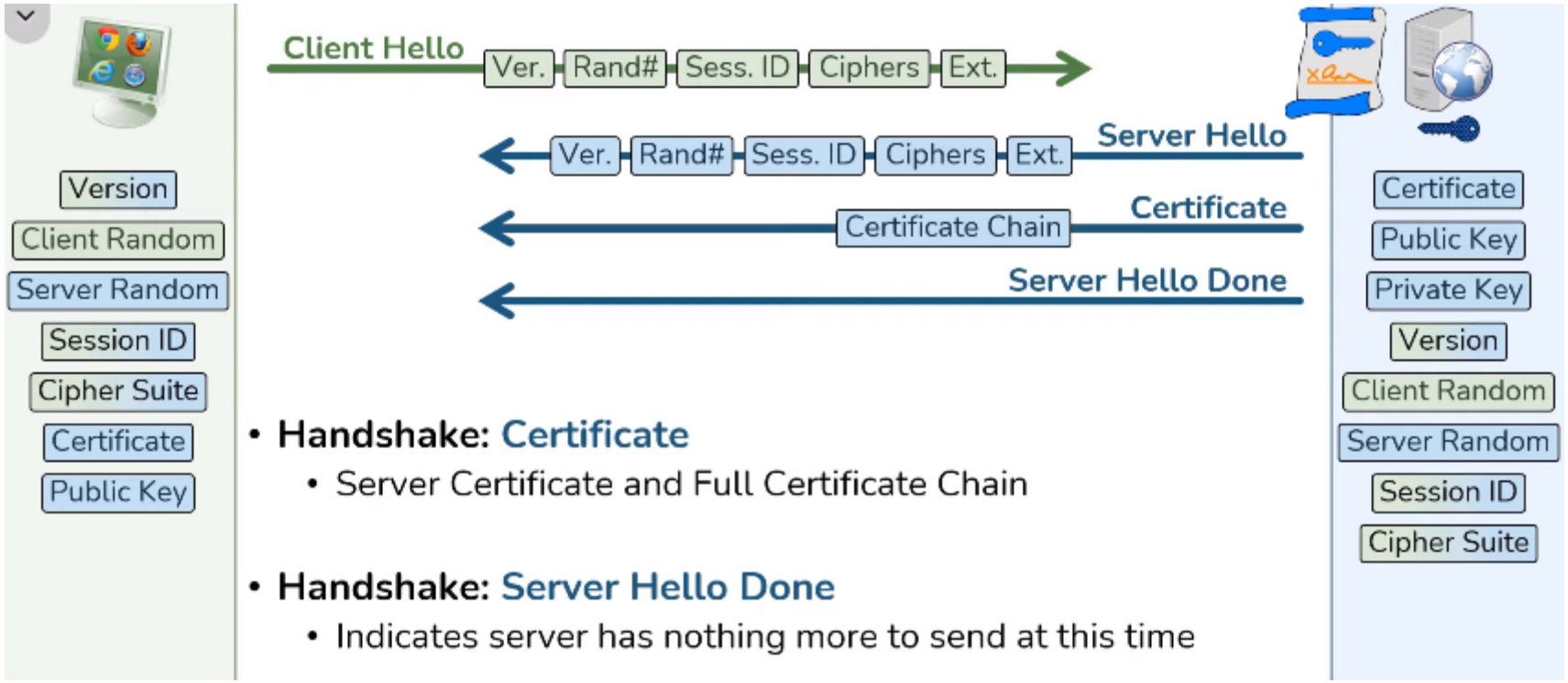


- **Handshake: Server Hello**
  - **Version**
    - Highest version of TLS/SSL **Server** supports
  - **Random Number** – 32 bytes / 256 bits
    - Timestamp encoded in first four bytes
  - **Session ID** – 8 bytes / 32 bits
    - Value generated by Server to identify ensuing Session Keys
  - **Cipher Suites**
    - Cipher Suite selected by **Server**
  - **Extensions**
    - Optional additional features added to TLS/SSL

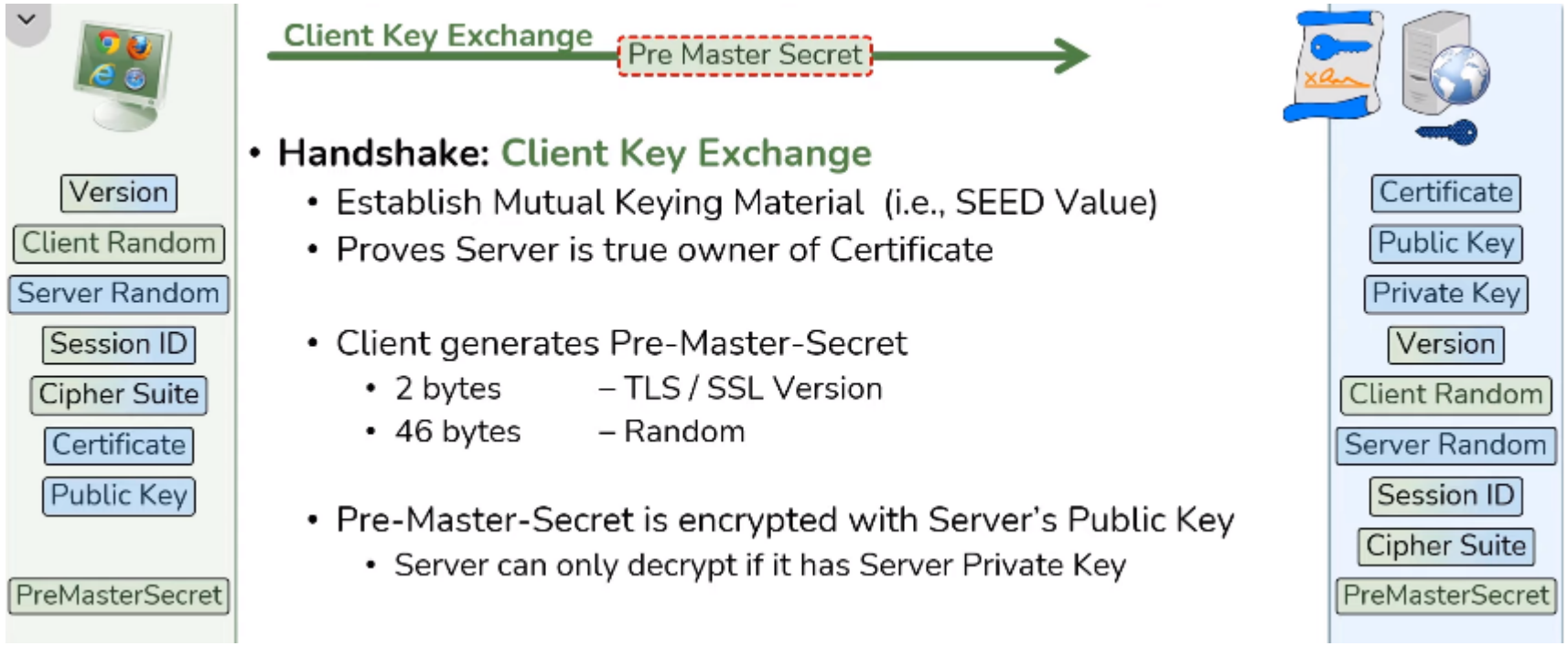
TLS Handshake



TLS Handshake



TLS Handshake



- **Handshake: Client Key Exchange**
  - Establish Mutual Keying Material (i.e., SEED Value)
  - Proves Server is true owner of Certificate
  - Client generates Pre-Master-Secret
    - 2 bytes – TLS / SSL Version
    - 46 bytes – Random
  - Pre-Master-Secret is encrypted with Server's Public Key
    - Server can only decrypt if it has Server Private Key

This is RSA Key Exchange

Other Key Exchange protocols create SEED value differently

TLS Handshake

Version

Client Random

Server Random

Session ID

Cipher Suite

Certificate

Public Key

PreMasterSecret

Master Secret

Session Keys

Certificate

Public Key

Private Key

Version

Client Random

Server Random

Session ID

Cipher Suite

PreMasterSecret

Master Secret

Session Keys

- Both parties have matching SEED Values
  - SEED value used to generate Session Keys
  - Pre Master Secret used to derive Master Secret
  - Master Secret used to generate Session Keys

PreMasterSecret
"master secret"
Client Random
Server Random

PRF

Master Secret
"key expansion"
Client Random
Server Random

PRF

Client Encryption Key
Client HMAC Key

Server Encryption Key
Server HMAC Key

IV Client I.V.

IV Server I.V.

- Calculations involve a PRF – **P**seudo **R**andom **F**unction
  - Hashing algorithm that generates digests at any desired length

TLS Handshake

Version

Client Random

Server Random

Session ID

Cipher Suite

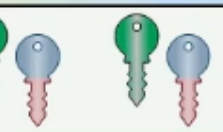
Certificate

Public Key

PreMasterSecret

Master Secret

Session Keys



Client Hello — Ver. | Rand# | Sess. ID | Ciphers | Ext. →

← Server Hello — Ver. | Rand# | Sess. ID | Ciphers | Ext.

← Certificate — Certificate Chain

← Server Hello Done

Client Key Exchange → Pre Master Secret

- At this point, both parties have identical Session Keys
- *BUT* C/S don't know whether the ***other*** has the same Keys
  - Rest of the handshake will prove to both parties that the other party has the correct session keys

Certificate

Public Key

Private Key

Version

Client Random

Server Random

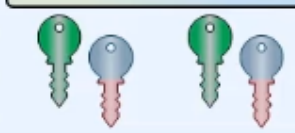
Session ID

Cipher Suite

PreMasterSecret

Master Secret

Session Keys



TLS Handshake

Version

Client Random

Server Random

Session ID

Cipher Suite

Certificate

Public Key

PreMasterSecret

Master Secret

Session Keys

Client Hello — Ver. | Rand# | Sess. ID | Ciphers | Ext. →

Server Hello ← Ver. | Rand# | Sess. ID | Ciphers | Ext.

Certificate ← Certificate Chain

Server Hello Done ←

Client Key Exchange → Pre Master Secret

Change Cipher Spec →

Finished → Encrypted Verification

Certificate

Public Key

Private Key

Version

Client Random

Server Random

Session ID

Cipher Suite

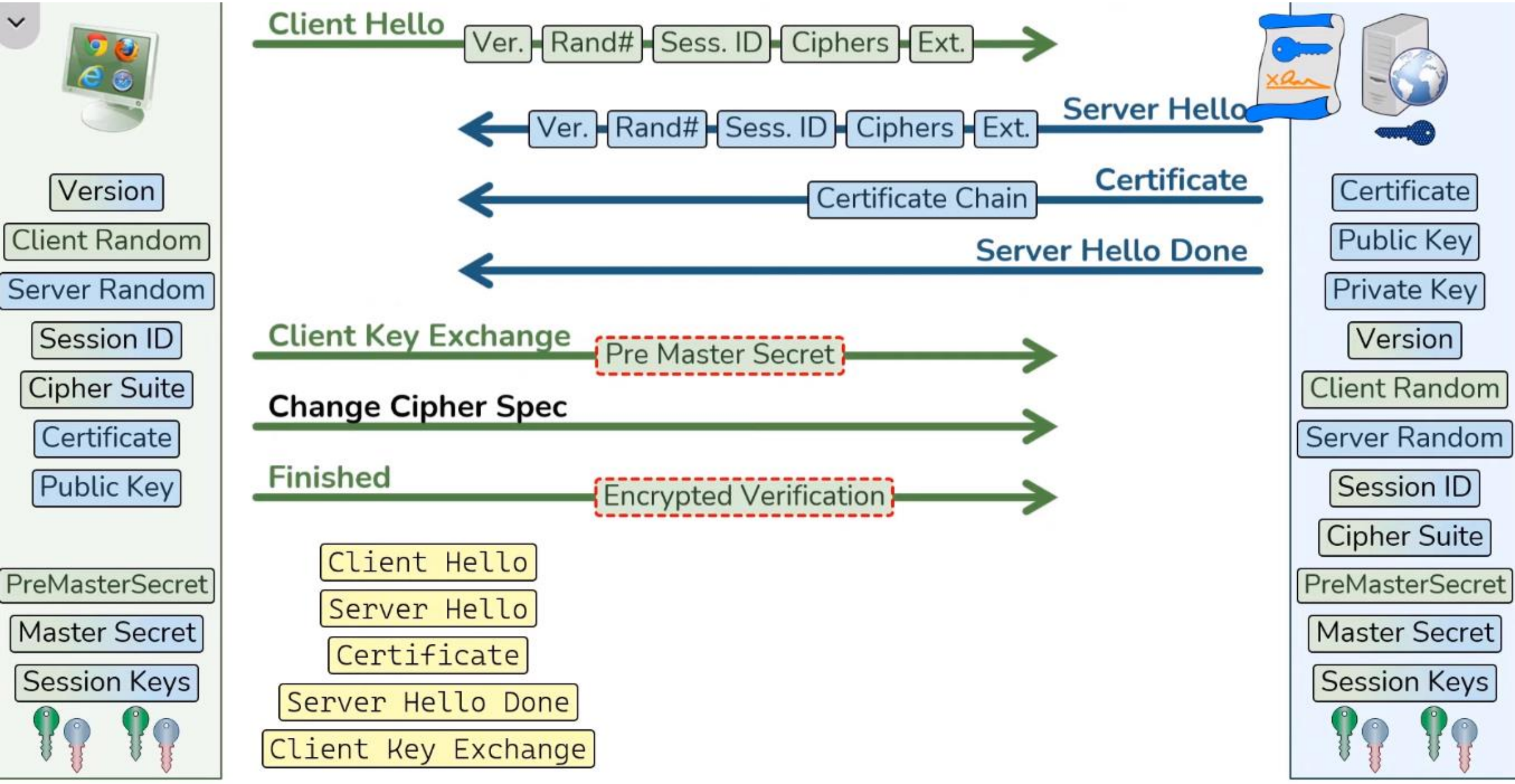
PreMasterSecret

Master Secret

Session Keys

- **Change Cipher Spec**
  - Indicates **Client** has everything necessary to speak securely
- **Handshake: Finished**
  - Proves to the **Server** that **Client** has correct Session Keys

TLS Handshake



TLS Handshake

Version

Client Random

Server Random

Session ID

Cipher Suite

Certificate

Public Key

PreMasterSecret

Master Secret

Session Keys

Change Cipher Spec

Finished

Encrypted Verification

- **Handshake: Finished**
  - Proves to the **Server** that **Client** has correct Session Keys
  - Client calculates Hash of all Handshake Records seen so far
  - Combined with other values to create Verification Data
  - Verification Data encrypted with **Client** Session Keys
    - **Server** verifies with **Server's** copy of **Client** Session Keys
      - Validates that **C/S** "saw" the same Handshake Records

Client Hello

Server Hello

Certificate

Server Hello Done

Client Key Exchange

Hash

Master Secret

"client finished"

Handshake Hash

PRF

Verification Data

Client Session Keys

Encrypted Verification

Certificate

Public Key

Private Key

Version

Client Random

Server Random

Session ID

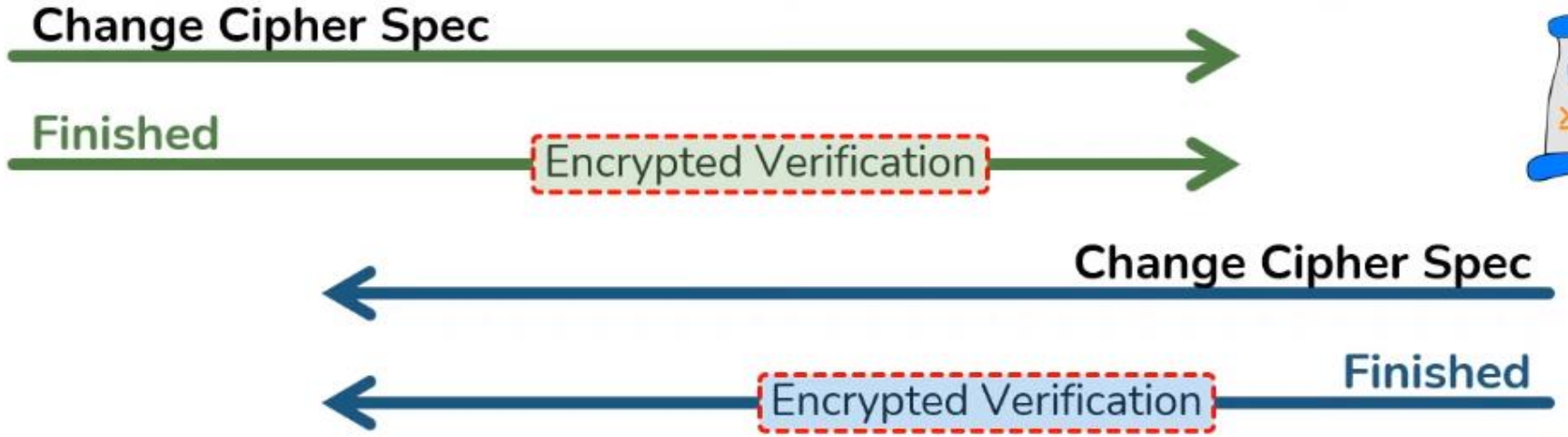
Cipher Suite

PreMasterSecret

Master Secret

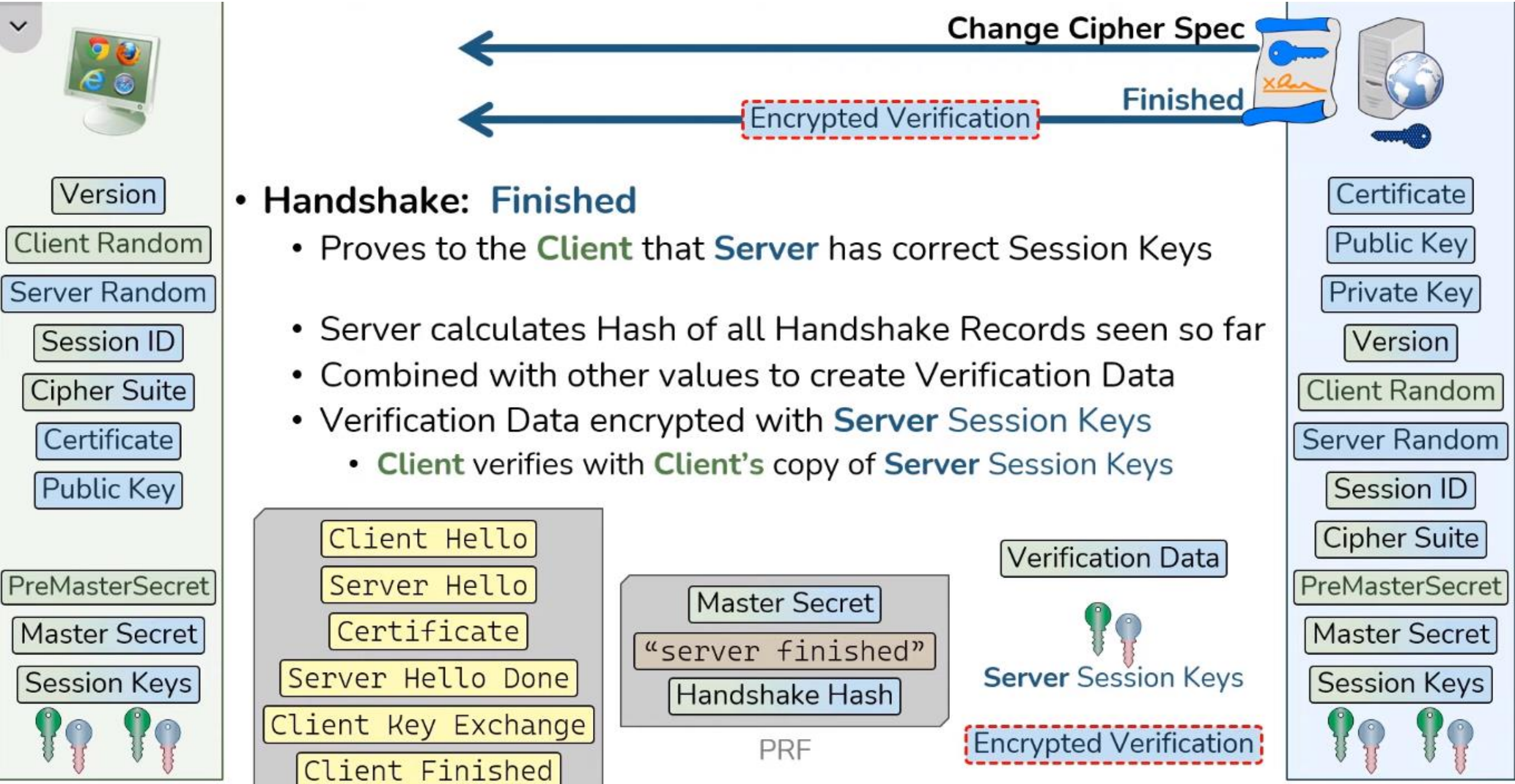
Session Keys

TLS Handshake



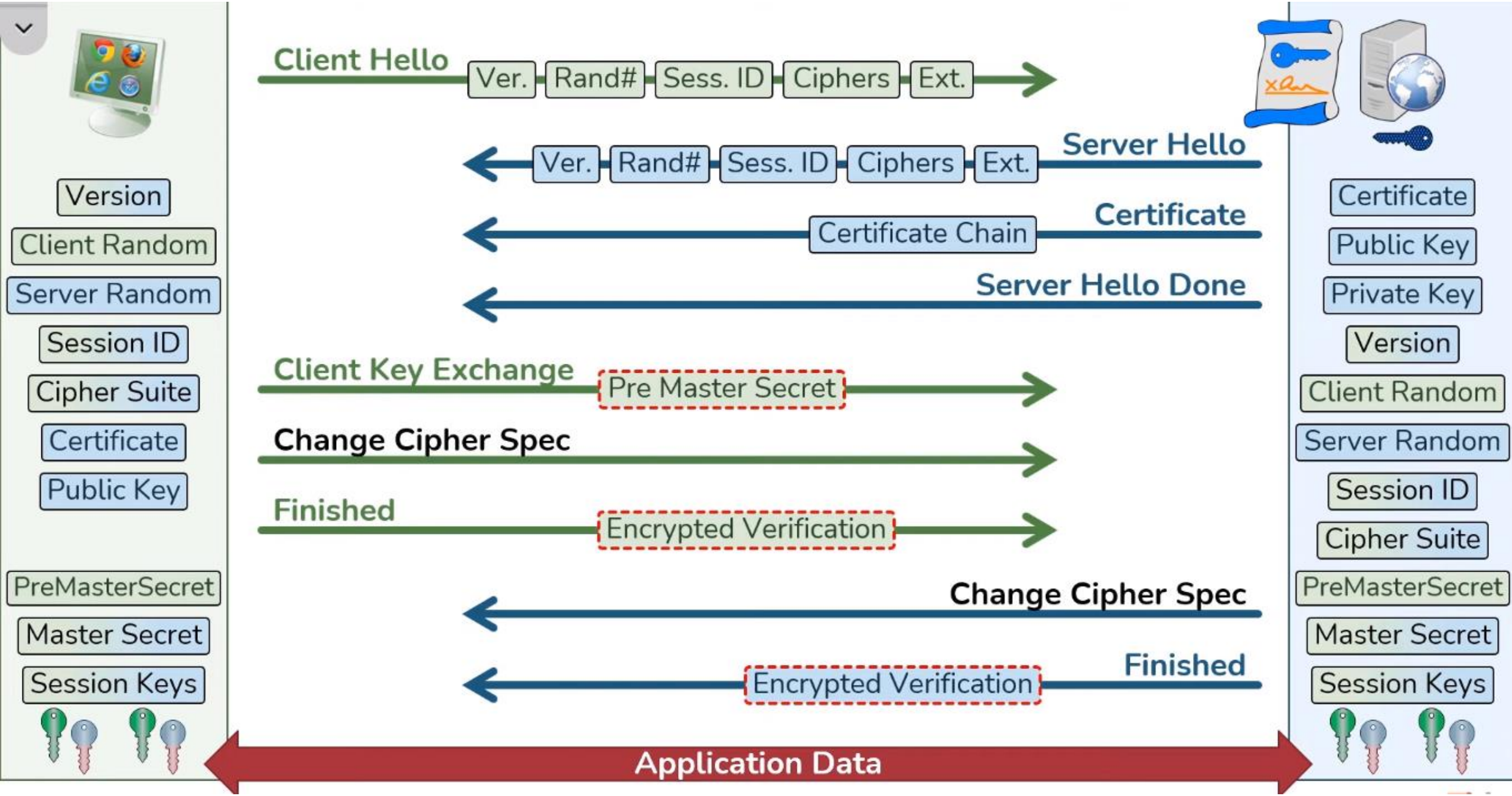
- **Handshake: Finished**
  - Proves to the **Server** that **Client** has correct Session Keys
- **Change Cipher Spec**
  - Indicates **Server** has everything necessary to speak securely
- **Handshake: Finished**
  - Proves to the **Client** that **Server** has correct Session Keys

TLS Handshake



- **Handshake: Finished**
  - Proves to the **Client** that **Server** has correct Session Keys
  - Server calculates Hash of all Handshake Records seen so far
  - Combined with other values to create Verification Data
  - Verification Data encrypted with **Server** Session Keys
    - **Client** verifies with **Client's** copy of **Server** Session Keys

TLS Handshake



TLS Handshake Complete Sequence

TLS Handshake Complete Sequence

1. **Certificate Authority** is the cornerstone of the TLS/SSL Process
   - **CA** has a **Public Key** and a **Private Key**
   - **CA** has a **Self-Signed Certificate**
2. **Server** wants to acquire a **Certificate**
3. **Server** generates a **Public Key** and a **Private Key**

The CA verifies the Server's Identity based upon the type of Certificate the Server purchased: **DV**, **OV**, or **EV**.
(more details in the full course: pracnet.net/tls)

4. **Server** generates a **Certificate Signing Request** (**CSR**)
   - **CSR** contains Server's **Public Key**
   - **CSR** is signed by Server's **Private Key**
5. **Server** gives signed **CSR** to **Certificate Authority** (**CA**)
6. **CA** inspects and validates information in **CSR**

TLS Handshake Complete Sequence

7. CA creates Certificate using information from CSR
8. CA signs Certificate using CA's Private Key
9. CA gives Certificate to Server
10. Server can then provide Certificate to prove its identity

11. Client wants to connect to the Server securely
    - Web Browsers already have CA Certificates installed
12. Client requests Server's Certificate
    - Client validates Certificate is legitimate
    - Client validates that Server truly owns Certificate

TLS Handshake Complete Sequence

13. Client validates Server's Certificate in the SSL / TLS Handshake
14. SSL / TLS Handshake produces **Session Keys**:
    - Symmetric Encryption
    - Message Authentication Code (MAC)
15. **Session Keys** form a Secure Tunnel to protect communication

TLS Handshake Notes

Understanding ssl and tls involves understanding the interaction between three key players, the client the server and the certificate authority.

As a high level overview of the entire ssl and tls ecosystem Certificate Authority it all starts with the certificate authority. The certificate authority is the cornerstone of the entire process the ca has its own asymmetric key pair that is a public key and a private key. In addition, the certificate authority also has a self-signed certificate. This certificate affirms the identity of this particular certificate authority. Notice that the certificate includes the ca's public key and the orange certificate which identifies the orange certificate authority is also signed in orange that's what's meant by a self-signed certificate.

With that as the starting point we can now introduce the server. This could be a website this could be an ssl vpn client this could also be an sslvpn termination point, the server is anything that wants to prove its identity and to do so it's going to need a certificate. To get a certificate, the server is going to start by generating its own public and private key then the server is going to generate a csr or a certificate signing request this is simply a file that is used to request a actual certificate. Inside this csr is the server's public key, moreover this csr is signed with the server's private key. This proves that the server has the correlating private key to the public key inside the csr. This csr will then be sent to the certificate authority. The certificate authority is then going to verify the information inside the csr. It's going to try and validate the identity of the server and make sure the server is who they say they are. Now there's a few different ways that a ca is going to do this. once the certificate authority is content with the identity of the server. It'll then generate an actual certificate using information that was inside the csr namely the public key. The ca has the server's public key because it was included in the csr, that the server sent furthermore this certificate is signed with the cas private key what this signed certificate does is it ties a particular set of asymmetric keys to a particular identity and this identity is guaranteed by the certificate authority. This certificate is then handed back to the server and now the server can use this to prove its identity for its clients.

client is simply an entity that wants to connect securely to the server but before it ever connects to the server there's something that the client already has the cas certificate pre-installed. This comes bundled with the various web browsers or sometimes with the operating system itself either way at this point we actually no longer need the certificate authority. Everything else that happens is simply between the client and the server. Now the client can make a request to the server and ask for the server's certificate upon receiving the certificate there are two things that the client must verify first the client has to validate that the certificate itself is legitimate certificates are just text inside of a file. how do we know that text hasn't been changed how do we know that text hasn't simply just been typed into notepad the client is going to verify this by checking the signature using the ca's public key. Remember the signature was created using the ca's private key which means it can be verified using the ca's public key which was included in the ca certificate, the client already has installed. The second thing that the client needs to verify is that the server is the true owner of that certificate. Remember certificates are public knowledge in fact every time you connect to a new website you are downloading that website certificate what's stopping you from presenting somebody else's certificate in order to spoof their identity that's the purpose of the second item right here and the way that is validated is the client is going to make sure that the server has the matching private key to the public key that was presented in the server certificate. In theory only the server should have the matching private key and if the server can prove that it has the matching private key, then we know that the server is indeed the true owner of this certificate. SSL handshake both of these items are validated in what's known as an ssl handshake within this handshake the client and the server exchange pieces of information which allow the client to verify the two items we just discussed. The handshake also allows the client and the server to establish cryptographic symmetric keys the outcome of a successful handshake is the agreement upon various session keys to protect bulk data transferred between the client and the server one set of symmetric encryption keys to provide confidentiality and one set of symmetric mac keys to provide integrity and authentication. These keys effectively create a secure tunnel within which data can be transferred securely between the client and the server. These 15 or so steps are the 10 000 foot view of the full tls and ssl.

What happens during a TLS handshake?

During the course of a TLS handshake, the client and server together will do the following:

- Specify which version of TLS (TLS 1.0, 1.2, 1.3, etc.) they will use
- Decide on which cipher suites (see below) they will use
- Authenticate the identity of the server via the server’s public key and the SSL certificate authority’s digital signature
- Generate session keys in order to use symmetric encryption after the handshake is complete

Cipher Suites
SSL

Cipher Suites
SSL

- For Client and Server to securely exchange data they need:
  - Authentication – verify Server's identity
  - Symmetric Encryption – confidentiality for bulk data transfer
  - Hashing Algorithm – Used within MAC for data integrity
  - Key Exchange Protocol – to generate necessary keys
- Client and Server must agree on specific protocols
- **Cipher Suite:**
  - Defines the actual protocols used to attain secure communication

Key Exchange | Authentication | Encryption | Hashing

TLS_DHE_RSA_WITH_AES_256_CBC_SHA

TLS_DH_DSS_WITH_3DES_EDE_CBC_SHA

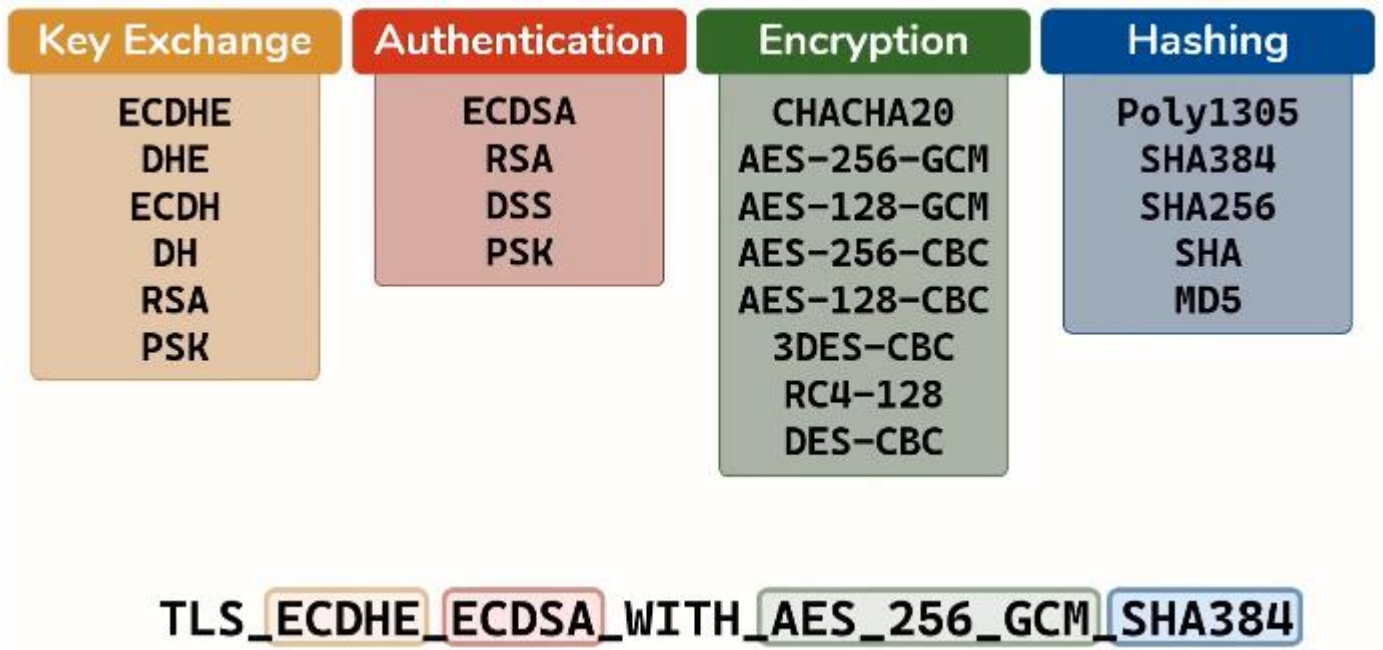
TLS_RSA_WITH_RC4_128_MD5

TLS_NULL_WITH_NULL_NULL

TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384

Cipher Suites SSL

- Each Cipher Suite specifies all four items
  - The user isn't picking and choosing individual protocols
- Cipher Suites are defined by IANA:
  - `https://www.iana.org/assignments/tls-parameters/tls-parameters.txt`



Changes to Cipher Suites TLS

Changes to Cipher Suites TLS

Old Protocols No longer supported

- TLS 1.2 released in 2008, included support for 3DES *(1978~)*
- TLS 1.1 released in 2006, included support for DES *(1972~)*

Key Exchange	Authentication	Encryption	Hashing
ECDHE	ECDSA	CHACHA20	Poly1305
DHE	RSA	AES-256-GCM	SHA384
~~ECDH~~	~~DSS~~	AES-128-GCM	SHA256
~~DH~~	~~PSK~~	~~AES-256-CBC~~	~~SHA~~
~~RSA~~		~~AES-128-CBC~~	~~MD5~~
~~PSK~~		~~3DES-CBC~~	
		~~RC4-128~~	
		~~DES-CBC~~	

- **TLS 1.3 removed insecure / suspect protocols**

Changes to Cipher Suites TLS

Simpler Cipher Suites

- Previously, a single Cipher Suite specifies 4 protocols:

TLS 1.2-Cipher Suite: `TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384`

Key Exchange | Authentication | Encryption | Hashing

- Over 300+ C.S.'s in TLS 1.2 and prior (270~ considered not secure)

- **TLS 1.3 divides these functions into 3 "*orthogonal*" choices:**

Key Exchange | Authentication | Encryption | Hashing

Key Exchange & Authentication chosen independently

`TLS_AES_256_GCM_SHA384`

TLS 1.3 Cipher Suite

- Three core functions of secure communication:
  - Who are you? — **Authentication**
  - How are we going to secure data? — **Encryption + Hashing**
  - How are we going to establish mutual keys? — **Key Exchange**

Changes to Cipher Suites TLS

Fewer Cipher Suite Choices

- **TLS 1.3 only includes 5 Cipher Suites:**
  - TLS_AES_128_GCM_SHA256 — MUST implement
  - TLS_AES_256_GCM_SHA384 — SHOULD implement
  - TLS_CHACHA20_POLY1305_SHA256 — SHOULD implement
  - TLS_AES_128_CCM_SHA256 — CAN implement
  - TLS_AES_128_CCM_8_SHA256 — CAN implement

All TLS 1.3 Ciphers are AEAD Ciphers

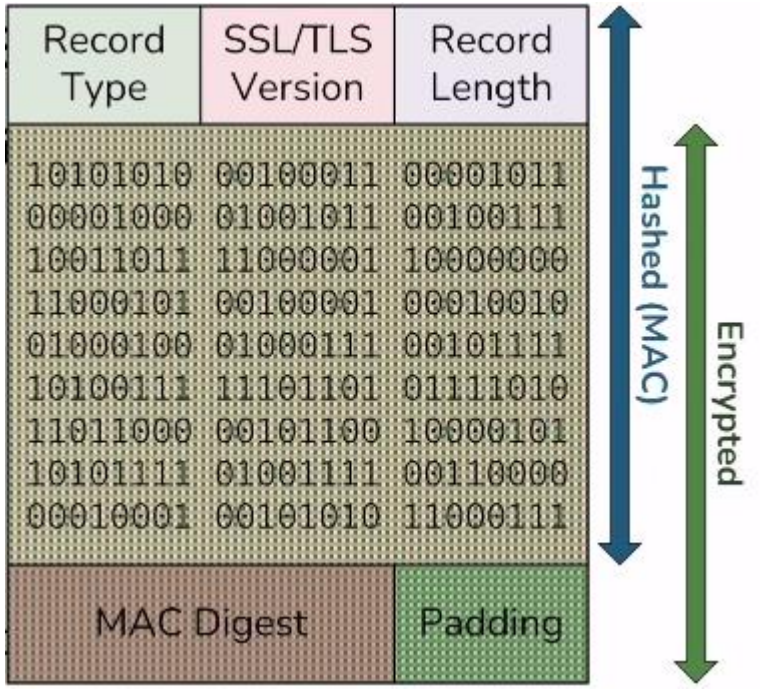
- **A**uthenticated **E**ncryption with **A**uthenticated **D**ata
  - Encryption & Integrity at the same time
    - No more consideration from "*mac-then-encrypt*" vs "*encrypt-then-mac*"
  - Introduced in TLS 1.2, required in TLS 1.3

- **TLS 1.3 only includes 5 Cipher Suites:**
  - TLS_AES_128_GCM_SHA256 — MUST implement
  - TLS_AES_256_GCM_SHA384 — SHOULD implement
  - TLS_CHACHA20_POLY1305_SHA256 — SHOULD implement
  - TLS_AES_128_CCM_SHA256 — CAN implement
  - TLS_AES_128_CCM_8_SHA256 — CAN implement

mac-then-encrypt TLS 1.2

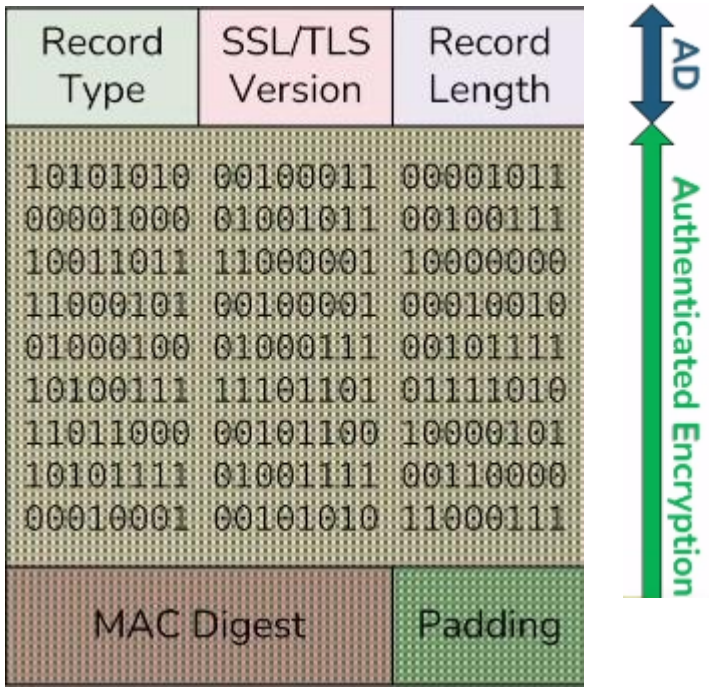
encrypt-then-mac IPSec

All TLS 1.3 Ciphers are AEAD Ciphers



Two Steps

Changes to Cipher Suites TLS



Same single Step

TLS 1.3

TLS 1.3 mandates Forward Secrecy

- Forward Secrecy – “once encrypted, always encrypted”
  - Keying Material generated from purely *ephemeral* values
    - Nothing saved, nothing hardcoded, new values every session
  - Future compromise of Certificate / Private Key does not lead to compromise of Encrypted Application Data
  - Provided in TLS 1.2- with KX that end with “E”
  - **TLS 1.3 only supports Ephemeral DH Key Exchanges**

Changes to Cipher Suites TLS

Changes to Cipher Suites TLS

Removes custom DH Groups

- Diffie-Hellman starts with agreeing upon some values
  - Starting Values known as "***DH Groups***"
    - Traditional DH – Prime / Generator
    - Elliptic Curve DH – Curve / Starting Coordinate

  - Approved **DH Groups** are designated via various Standards:
    - Traditional DH Groups: RFC 2409, RFC 3526 *("finite field DH" – real numbers)*
    - Elliptic Curve Groups: RFC 5639, SEC2, FIPS 186-4, Curve25519

- TLS 1.2- supported proposing custom Groups
  - Seems more secure... but lead to many insecure groups being used

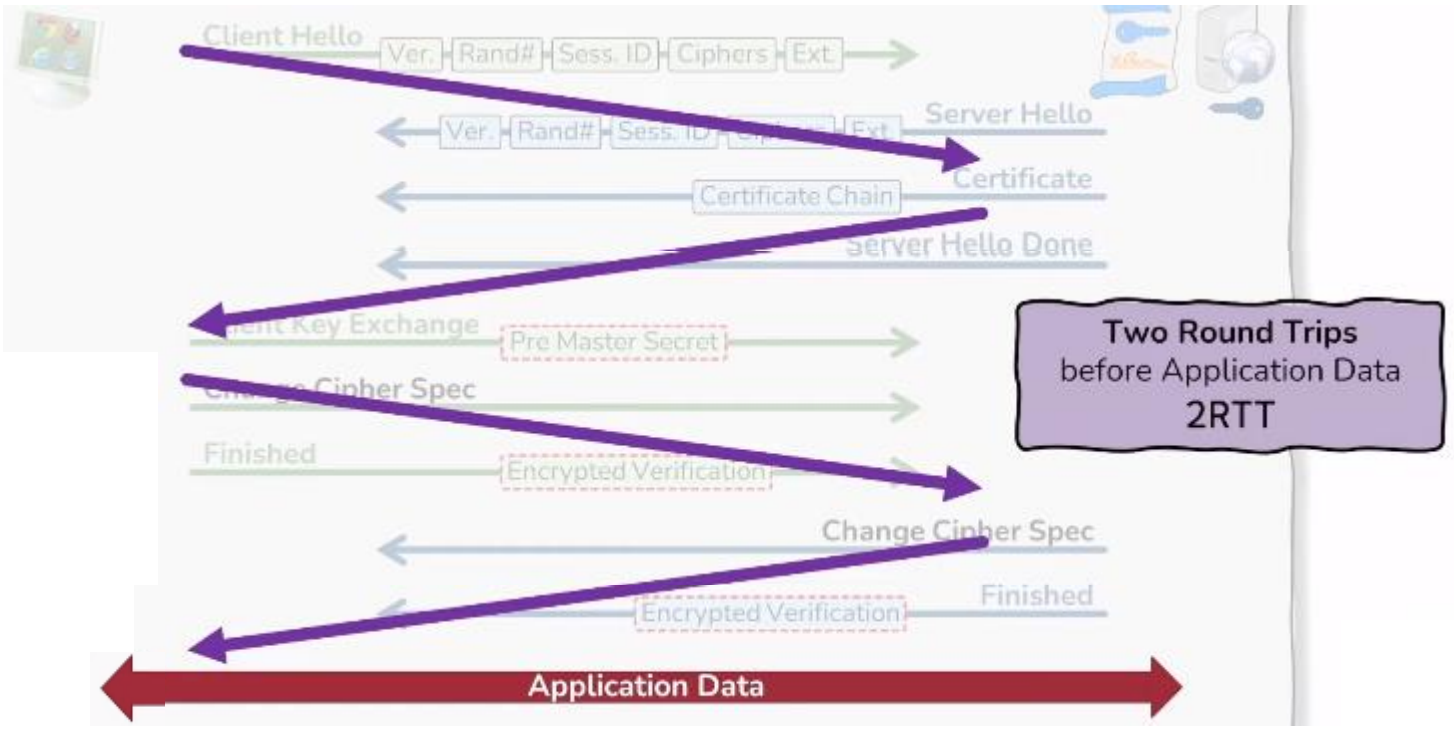
- **TLS 1.3 mandates approved, standards based groups only**

Changes to Cipher Suites TLS

- **TLS 1.3 changes related to Ciphers and Cipher Suites:**
  - Old protocols no longer supported
  - Simpler Cipher Suites
  - Fewer Cipher Suites
  - All TLS 1.3 Ciphers are AEAD Ciphers
  - Forward Secrecy
  - Removed Custom DH Groups / Curves

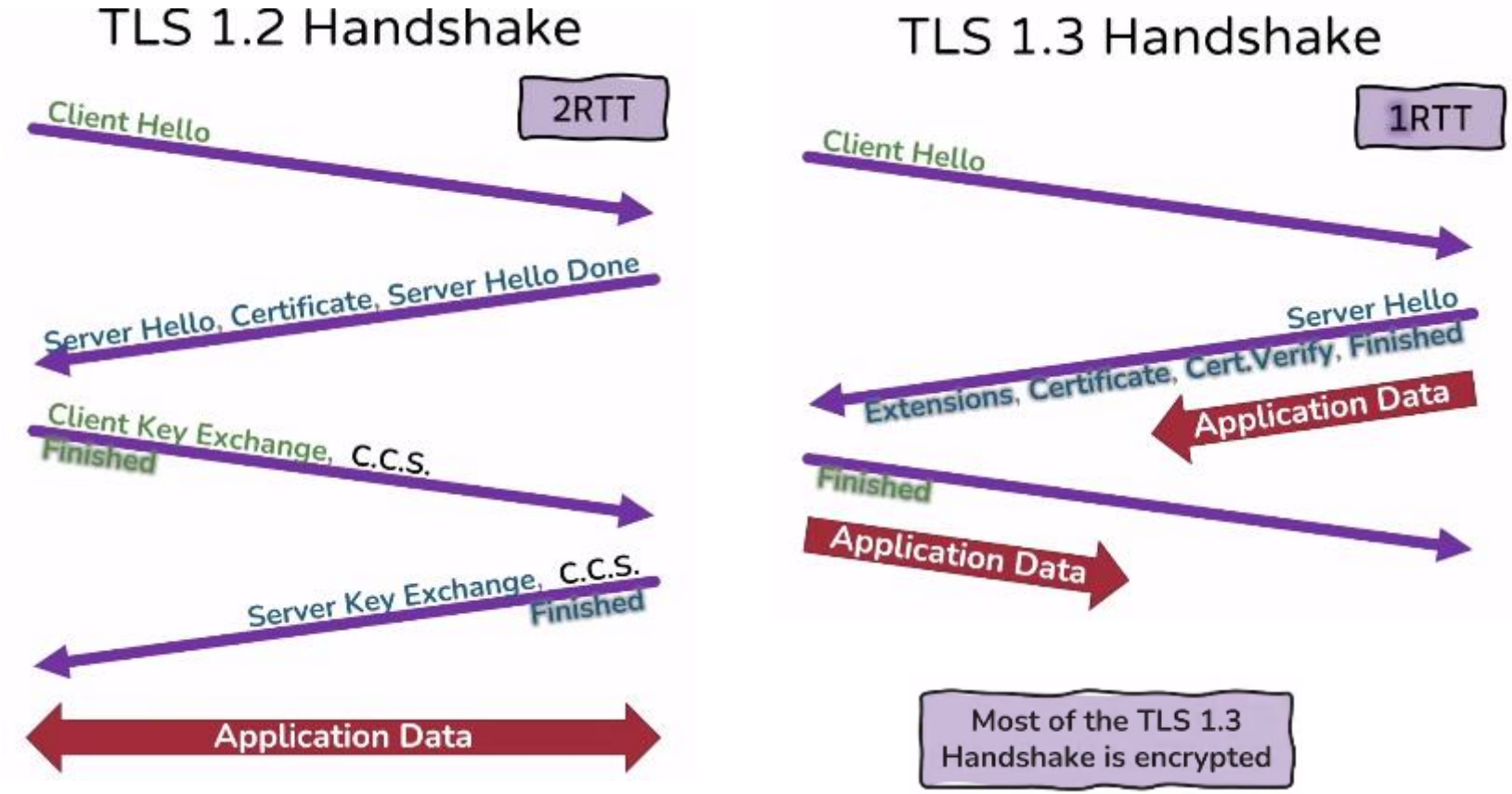
Changes to Handshake TLS

TLS 1.2 Handshake

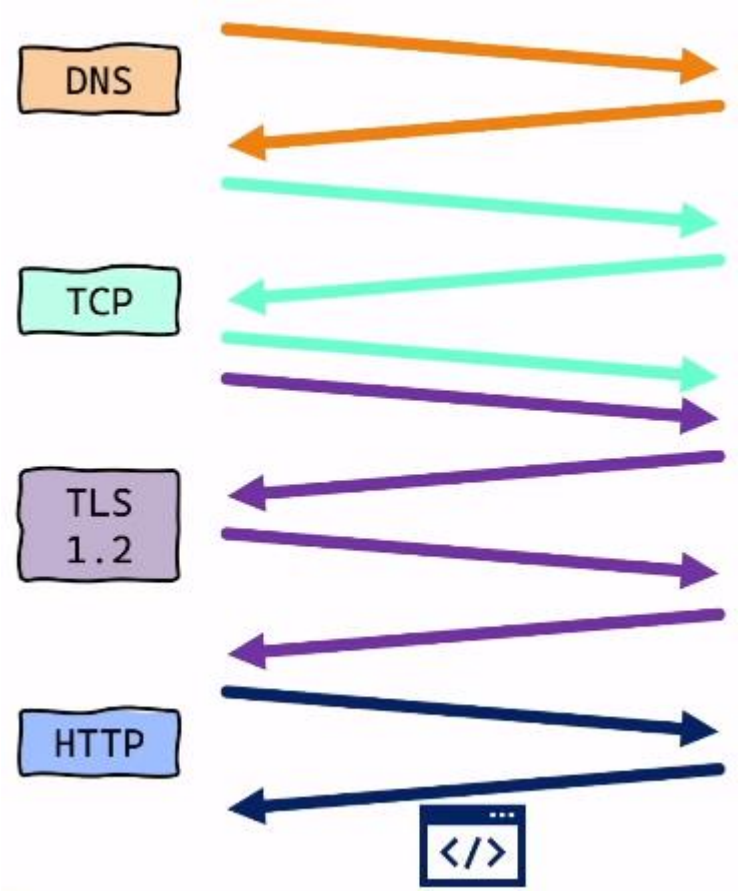


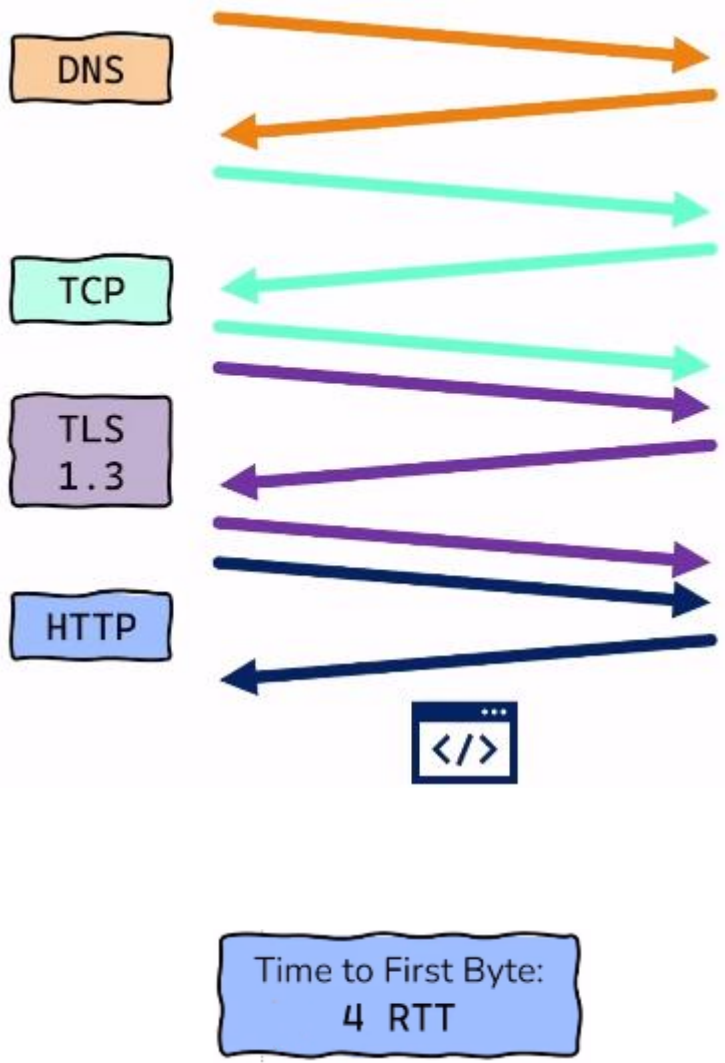
Changes to Handshake TLS

Changes to Handshake TLS

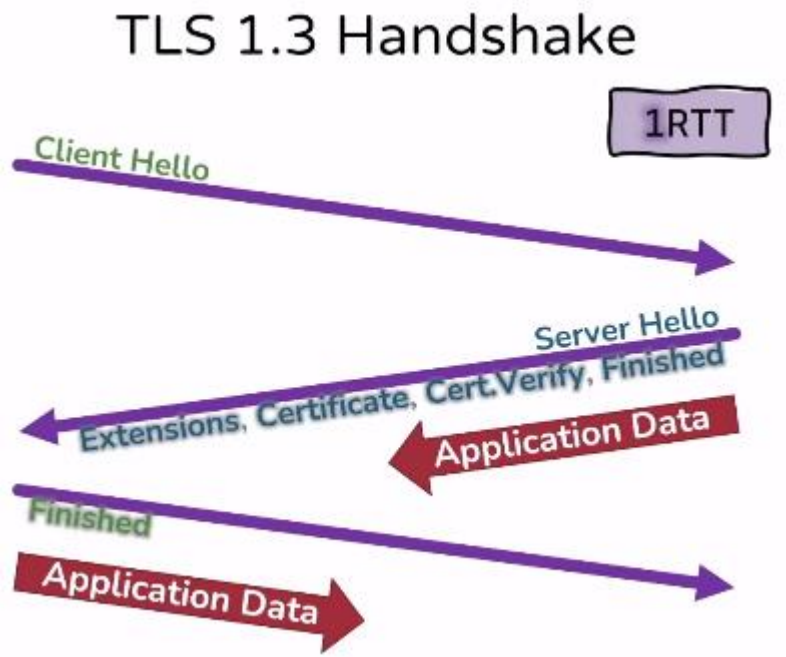


Changes to Handshake TLS





Changes to Handshake TLS



Comparison in the Handshake Process between TLS 1.3 and SSL

Feature/Aspect	SSL	TLS 1.3
Round Trips	More round trips are generally required.	Significantly reduces the number of round trips. Introduces 0-RTT for session resumption, allowing data to be sent in the initial "ClientHello."
Cipher Suites	Supports a variety of cipher suites, including some that are now considered insecure.	Simplifies and streamlines the list of supported cipher suites. Removes insecure algorithms.
Key Exchange	Options include RSA, Diffie-Hellman, and others.	Emphasizes and mandates the use of secure key exchange mechanisms like ECDHE (Elliptic Curve Diffie-Hellman Ephemeral).
Forward Secrecy	SSL may or may not provide forward secrecy, depending on the key exchange mechanism.	Mandates forward secrecy through the use of ephemeral key exchange mechanisms (e.g., ECDHE).
Session Resumption	Session resumption is supported through session IDs and session tickets.	Introduces a more efficient and secure session resumption mechanism, including "0-RTT" for quicker resumption.
Certificate Authentication	Certificate authentication is present but may not be mandatory.	Strengthens the importance of certificate authentication. The server must provide a digital certificate.
Renegotiation	SSL supports renegotiation, which can introduce vulnerabilities (e.g., BEAST attack).	Eliminates renegotiation for enhanced security.
Obsolete Cryptographic Algorithms	Supports obsolete and insecure cryptographic algorithms.	Removes support for obsolete and insecure algorithms, promoting modern cryptographic practices.
Resistant to Known Attacks	May be vulnerable to known attacks such as POODLE, BEAST, and others.	Designed with security improvements to resist known attacks.
Padded RSA	Vulnerable to certain padding oracle attacks.	Removes support for certain padding schemes, reducing the

TLS Step by Step Student Notes

- Transport Layer Security (TLS) introduces improved key exchange methods to enhance the security of encrypted communications. The key exchange process is a crucial aspect of establishing a secure communication channel between a client (such as a web browser) and a server.
- How TLS introduces improved key exchange methods:

1. Client Hello:

1. The TLS handshake begins with the client sending a "ClientHello" message to the server.
2. This message includes information about the TLS version supported by the client, a random value, and a list of cryptographic algorithms and options it supports.

2. Server Hello:

1. The server responds with a "ServerHello" message, selecting the highest TLS version that both the client and server support.
2. The server also sends its digital certificate containing its public key, which is used to establish the authenticity of the server.

Student Notes TLS Step by Step

3. Key Exchange:

1. In traditional SSL, the key exchange method was often based on the server's private key, which posed some vulnerabilities.
2. TLS introduces various key exchange methods, including:
    1. **RSA Key Exchange:** The server's public key is used for key exchange, and the actual key is sent encrypted with the server's public key.
    2. **Diffie-Hellman Key Exchange (DHE):** This method allows the client and server to jointly agree on a shared secret over an insecure communication channel without directly exchanging the secret.

4. Perfect Forward Secrecy (PFS):

1. TLS emphasizes the use of Perfect Forward Secrecy (PFS) to enhance security.
2. With PFS, even if a long-term secret key is compromised, past sessions remain secure because the session keys are ephemeral and not derived from the long-term secret.

Student Notes TLS Step by Step

5. Ephemeral Key Exchange:

1. Ephemeral key exchange methods, such as DHE or Ephemeral Diffie-Hellman (EDH), generate temporary, session-specific keys for key exchange.
2. This enhances security by ensuring that even if an attacker intercepts the key exchange, they cannot derive the session keys used for encryption.

6. Key Derivation:

1. Once the key exchange is complete, both the client and server derive session keys from the agreed-upon shared secret.
2. These session keys are then used for encrypting and decrypting the data exchanged during the secure communication session.

7. Finish:

1. The TLS handshake concludes with a "Finished" message from both the client and server, confirming that they have successfully established a secure connection.

Student Notes TLS Step by Step

By introducing improved key exchange methods like Diffie-Hellman and emphasizing Perfect Forward Secrecy, TLS enhances the security of the key exchange process compared to earlier versions of SSL. These enhancements contribute to the overall security of encrypted communication on the internet.

Student Notes Perfect Forward Secrecy (PFS)

- Perfect Forward Secrecy (PFS) is a property of certain key exchange algorithms used in cryptographic protocols like Transport Layer Security (TLS). PFS ensures that, even if a long-term secret key is compromised, past communication sessions remain secure because the session keys are ephemeral and not derived from the long-term secret. Following is a step-by-step explanation of how Perfect Forward Secrecy works:

1. Key Exchange Initialization:

1. The cryptographic protocol (e.g., TLS) initiates the key exchange process between the client and the server during the handshake phase.

2. Ephemeral Key Generation:

1. PFS relies on the use of ephemeral keys, which are temporary and unique to each session.
2. During the key exchange, both the client and the server generate ephemeral key pairs specifically for that session.

Student Notes Perfect Forward Secrecy (PFS)

3. Key Exchange Process:

1. The client and server engage in a key exchange process using the ephemeral keys.
2. The specifics of the key exchange method may vary (e.g., Diffie-Hellman, Elliptic Curve Diffie-Hellman), but the essence is that both parties contribute to the creation of a shared secret without directly transmitting it.

4. Shared Secret Establishment:

1. Through the key exchange process, the client and server establish a shared secret known only to them.

5. Session Key Derivation:

1. From the shared secret, session keys are derived. These session keys are used for encrypting and decrypting the actual data transmitted during the session. KDF

6. Temporary Nature of Keys:

1. The ephemeral keys used for the key exchange are discarded after the session is established. They are not stored or used beyond the specific communication session.

Student Notes Perfect Forward Secrecy (PFS)

7. Data Encryption:

1. The session keys derived from the ephemeral keys are now used for encrypting and decrypting the data exchanged between the client and server during the session.

8. Post-Session Security:

1. Even if an attacker gains access to the long-term secret key of either the client or the server in the future, they cannot decrypt past communications because those sessions used ephemeral keys that are no longer available.

9. Reduced Impact of Key Compromise:

1. PFS significantly reduces the impact of a key compromise. In the absence of PFS, if a long-term secret key is compromised, all past and future communications encrypted with that key are at risk.

10. Renewal for Each Session:

- For subsequent sessions, new ephemeral keys are generated, and the process repeats. This ensures that each session has its own unique set of keys.

Student Notes Perfect Forward Secrecy (PFS)

Perfect Forward Secrecy ensures that the compromise of long-term secret keys doesn't jeopardize the security of past communication sessions. It achieves this by using temporary, session-specific keys for key exchange, and these keys are discarded after the session is established. This property significantly enhances the overall security of encrypted communications.

Note: TLS uses DH and Elliptic curve Diffie Hellman

PFS Student Notes

We all have lots of private stuff we leave online. Our chat logs can reveal a lot about our personal lives and even expose some sensitive data. So, we certainly don’t want anyone accessing them, especially cybercriminals.

What is perfect forward secrecy

When we communicate online, cryptographic protocols usually encrypt all our messages. However, in the case of certain man-in-the-middle attacks, hackers’ can intercept users’

traffic. And then they can get hold of the data. This is why developers implement the perfect forward secrecy feature into their apps and websites. It generates unique encryption and decryption keys for each session. The compromise of such a key would affect only the data of that particular session as the other sessions use different keys. A snooper would only be able to penetrate a single chat message, video call, or website visit rather than the whole history of your activities. They may even hack your long-term secrecy keys, but they still won’t be able to access the data encrypted with the forward secrecy system. It is because it uses its own set of ever-changing keys. The flaws of forward secrecy Basically, you get an extra layer of encryption, and your data gets much safer. Unfortunately, the concept is not without its flaws and, in rare cases, hackers can exploit it too. If they learn how a device generates the keys, forward secrecy won’t work. In this case, they can crack a session key generator and access your data too.

Conclusion

However, forward secrecy still offers reliable protection and is an essential companion to end-to-end encryption. While most secure instant message platforms use it, it’s not always the default option in the case of HTTPS websites. So, do some research on whether websites you visit use it.

SSL/TLS Student Notes

n the process of secure communication using SSL/TLS, the main difference lies in how the initial secure connection is established and how subsequent data is encrypted:

SSL/TLS Handshake:

SSL (Secure Sockets Layer): In SSL, the handshake begins with the client sending a "ClientHello" message containing the SSL version and supported cipher suites. The server responds with a "ServerHello" message, selecting the SSL version and cipher suite to use. The server then sends its certificate to the client, which contains its public key. The client verifies the certificate and generates a pre-master secret, encrypts it with the server's public key, and sends it back to the server. Both the client and server use the pre-master secret to generate the session keys used for symmetric encryption of data.

TLS (Transport Layer Security): The TLS handshake is similar to SSL but includes additional security features. It also supports more secure cipher suites and algorithms, such as AES encryption, which are not available in SSL. The TLS handshake establishes a secure connection by exchanging certificates, generating session keys, and verifying the integrity of the communication.

Data Encryption:

SSL: In SSL, the initial handshake establishes a secure connection using asymmetric encryption (public-private key pairs) to exchange the symmetric session key. Once the session key is established, SSL switches to symmetric encryption for data transmission using this session key. This means that SSL uses both asymmetric and symmetric encryption in the process.

TLS: Similarly, TLS uses asymmetric encryption during the handshake to establish a secure connection and exchange the session key. After the handshake, TLS also switches to symmetric encryption using the session key for data transmission. Like SSL, TLS uses both asymmetric and symmetric encryption in the process.

Note: both SSL and TLS use both asymmetric (for handshake) and symmetric (for data encryption) encryption in the process of establishing a secure connection and encrypting data. The main differences lie in the versions of the protocols, the supported cipher suites and algorithms, and the security features.

Step-by-Step Example of Diffie-Hellman Key Exchange in TLS

Public Parameters Setup:

Choose a large prime number, p = 23, and a primitive root modulo p, g = 5. These are public parameters shared by both the client and server.

Key Pair Generation:

Both the client and server generate their key pairs.

Client:

- Private Key (a): 6
- Public Key (A): (g^a) mod p = (5^6) mod 23 = 8

Server:

- Private Key (b): 15
- Public Key (B): (g^b) mod p = (5^15) mod 23 = 19

Step-by-Step Example of Diffie-Hellman Key Exchange in TLS

Key Exchange:

The client sends its public key (A = 8) to the server. The server sends its public key (B = 19) to the client.

Shared Secret Calculation:

Both the client and server independently calculate the shared secret using each other's public key and their own private key.

Client:

- Shared Secret (s_c): (B^a) mod p = (19^6) mod 23 = 2

Server:

- Shared Secret (s_s): (A^b) mod p = (8^15) mod 23 = 2

.

Step-by-Step Example of Diffie-Hellman Key Exchange in TLS

Encryption Key Derivation:

Both the client and server use the shared secret to derive a symmetric encryption key.

Client:

- Encryption Key (K_c): Hash(s_c) = Hash(2) = Some value

Server:

- Encryption Key (K_s): Hash(s_s) = Hash(2) = Same value

Now, the client and server share the same secret key (K) that can be used for symmetric encryption and decryption during the TLS session. The key exchange is secure because, even if an eavesdropper intercepts the public keys (A and B), they cannot easily compute the shared secret without knowledge of the private keys (a and b). This ensures the confidentiality of the communication. This is a simplified example, and in real-world scenarios, more advanced algorithms and additional security measures are employed in TLS

Must visit Links

https://www.cloudflare.com/en-gb/learning/ssl/what-is-https/

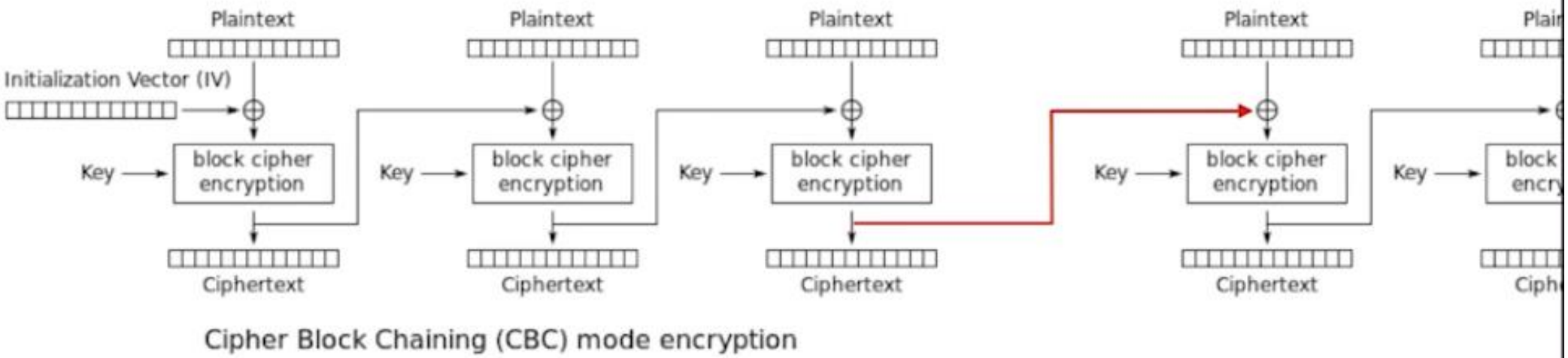
https://www.cloudflare.com/en-gb/learning/ssl/what-is-an-ssl-certificate/

https://www.cloudflare.com/en-gb/learning/cdn/what-is-a-cdn/

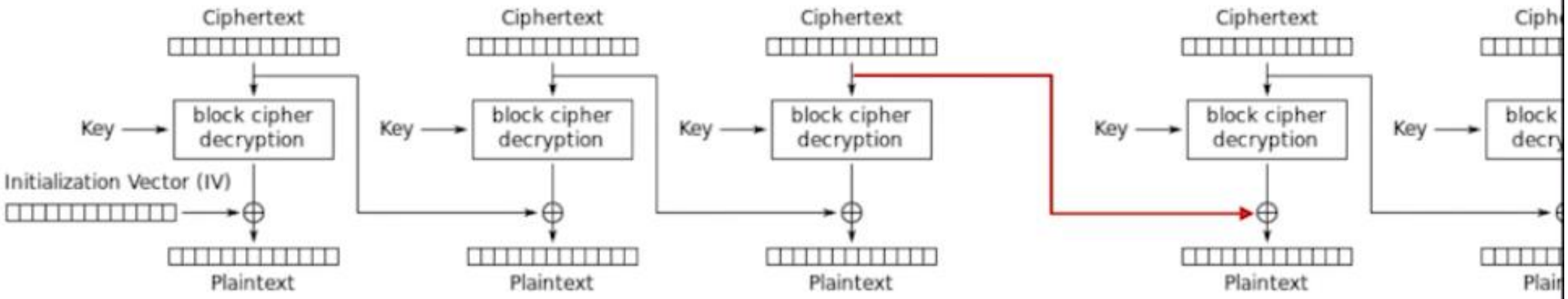
https://www.cloudflare.com/en-gb/learning/ssl/keyless-ssl/

https://www.cloudflare.com/en-gb/learning/ssl/what-happens-in-a-tls-handshake/

BEAST Attack



Cipher Block Chaining (CBC) mode encryption



Cipher Block Chaining (CBC) mode decryption

IPsec

IPsec, or Internet Protocol Security, is a suite of protocols and standards that provides secure communication over the Internet. It's commonly used to establish Virtual Private Networks (VPNs) and to secure communication between devices over an IP network. IPsec operates at the network layer of the OSI model and is designed to protect the integrity, confidentiality, and authenticity of data as it traverses the network.

How IPsec Works

- IPsec (Internet Protocol Security) operates in two main phases: IKE (Internet Key Exchange) Phase 1 and IKE Phase 2. These phases are part of the process used to establish a secure communication channel between two devices. Following is a step-by-step explanation of both IKE Phase 1 and IKE Phase 2:
- **IKE Phase 1:** IKE Phase 1 is responsible for establishing a secure, authenticated communication channel and negotiating the parameters for subsequent communication. It typically involves the negotiation of keying material and the establishment of a secure channel for the exchange of IKE Phase 2 information.

How IPsec Works

The steps are as follows:

1. Initiation:

1. The process begins with one party initiating the IKE Phase 1 negotiation. This could be either the client or the server.

2. Security Association (SA) Proposal:

1. The initiator proposes a set of security parameters, including encryption algorithms, integrity algorithms, and Diffie-Hellman (DH) group for key exchange.

3. Response:

1. The responder evaluates the proposal, and if acceptable, responds with its own set of security parameters.

4. DH Key Exchange:

1. The parties perform a Diffie-Hellman key exchange to establish a shared secret. This shared secret is used to derive keys for subsequent encryption and integrity protection.

How IPsec Works

5. Authentication:

1. Mutual authentication occurs, ensuring that both parties are who they claim to be. This is typically achieved using digital certificates or pre-shared keys.

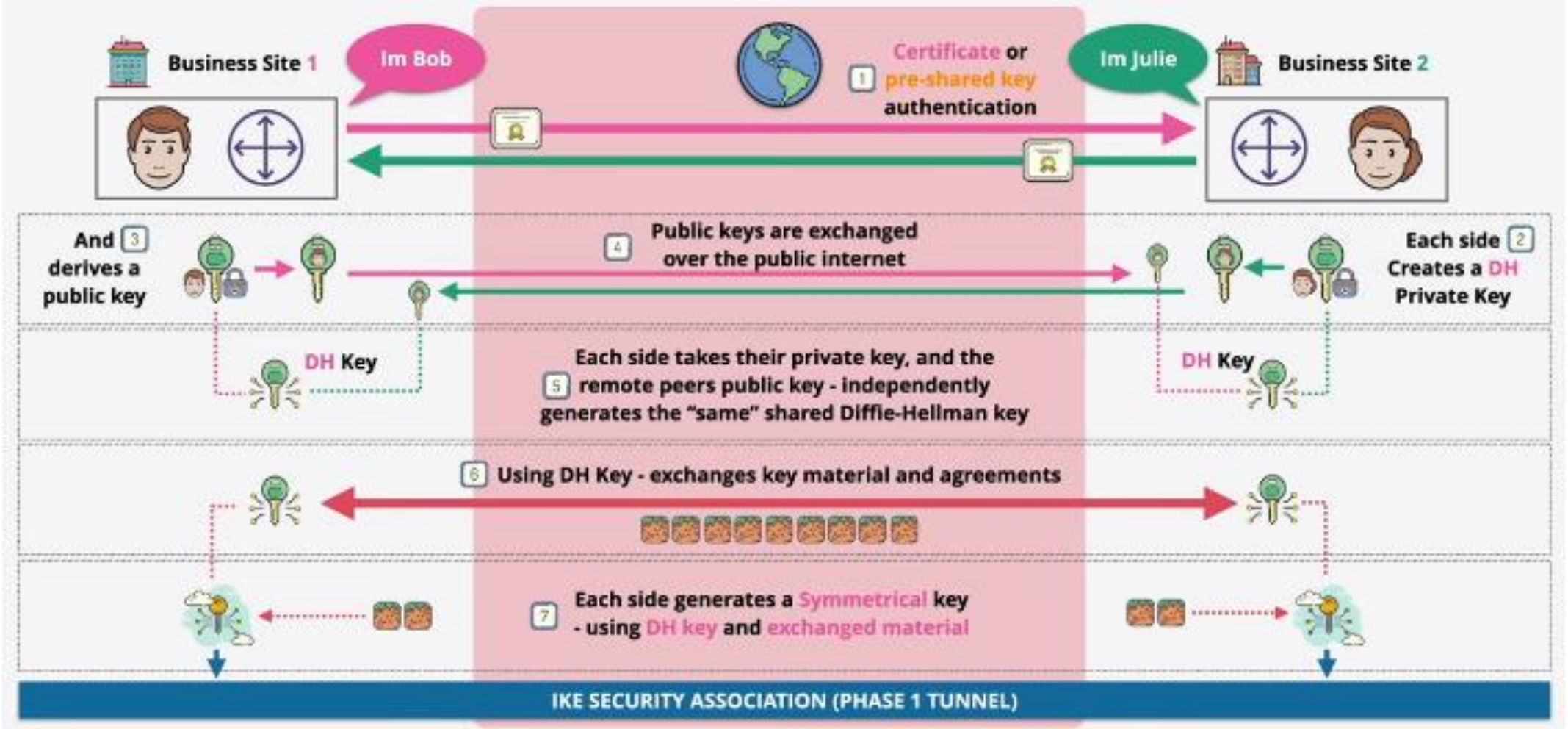
6. Generation of Keying Material:

1. Based on the shared secret and additional parameters, keying material is generated for use in subsequent phases.

7. IKE Phase 1 SA Establishment:

1. Once the negotiation is successful, the IKE Phase 1 Security Association (SA) is established. This SA contains the keying material and parameters agreed upon during the negotiation.

How IPsec Works



How IPsec Works

IKE Phase 2:

IKE Phase 2 builds upon the established Phase 1 SA and focuses on negotiating parameters for IPsec, such as the type of IPsec protocol (AH or ESP), encryption algorithms, and the lifetimes of the security associations. The steps are as follows:

1. Initiation:

1. The initiator (which may be the same device as in Phase 1) initiates the IKE Phase 2 negotiation.

2. Security Association (SA) Proposal:

1. The initiator proposes a set of parameters for the IPsec SA, including the IPsec protocol (AH or ESP), encryption and integrity algorithms, and lifetime values.

3. Response:

1. The responder evaluates the proposal and responds with its own set of parameters.

How IPsec Works

4. Traffic Selectors:

1. The parties negotiate traffic selectors, defining the specific traffic that will be protected by the IPsec SA.

5. DH Key Exchange (Optional):

1. An optional Diffie-Hellman key exchange may occur for keying material refresh.

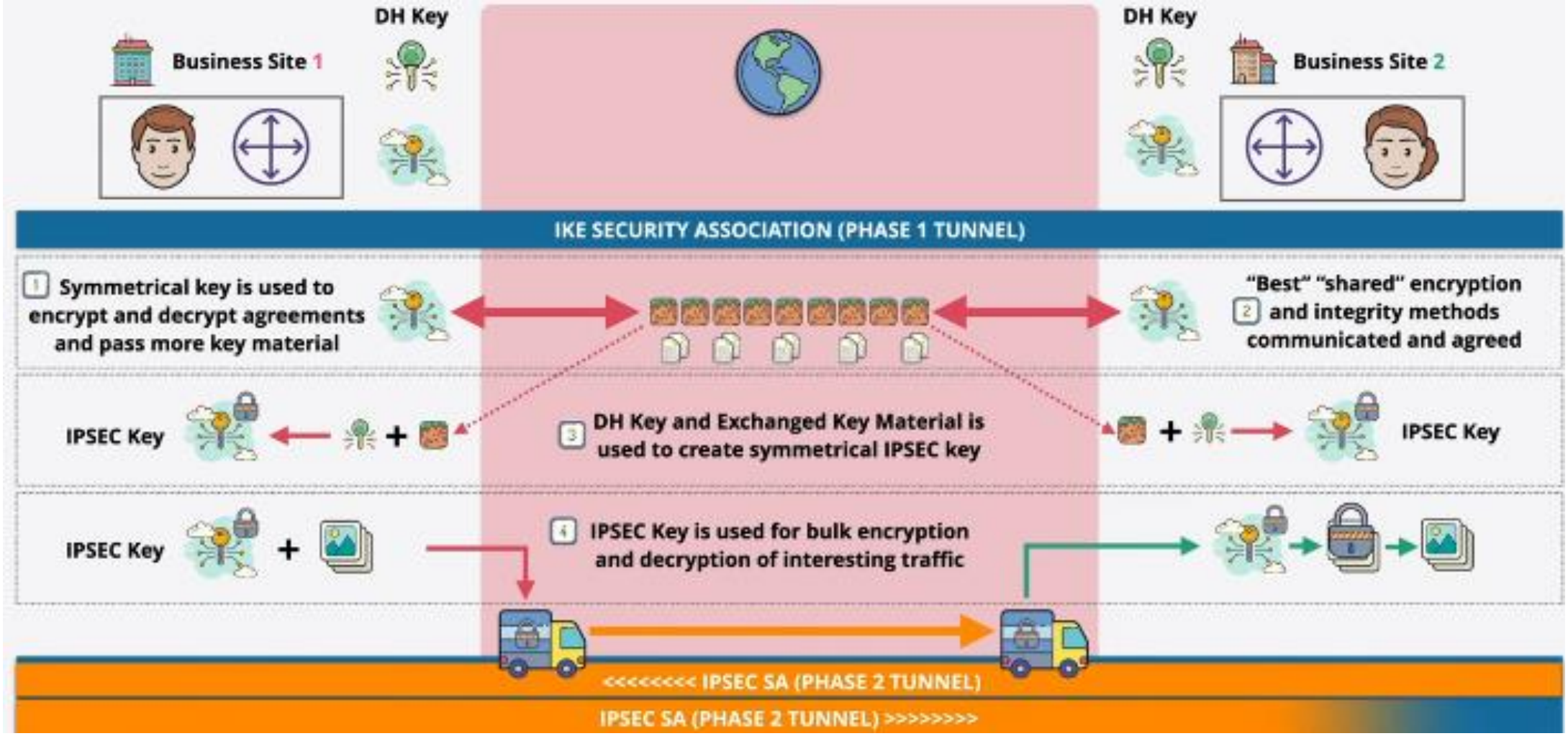
6. Generation of Keying Material:

1. Keying material for IPsec is generated based on the shared secret established during Phase 1.

7. IKE Phase 2 SA Establishment:

1. Once the negotiation is successful, the IKE Phase 2 Security Association (SA) is established. This SA contains the parameters and keying material agreed upon during the negotiation.

How IPsec Works



How IPsec Works

After IKE Phase 1 and Phase 2, secure communication can take place using the negotiated security associations, ensuring confidentiality, integrity, and authenticity of the transmitted data. The periodic refresh of keying material helps maintain the security of the communication over time.

How IPsec Works

After IKE Phase 1 and Phase 2, secure communication can take place using the negotiated security associations, ensuring confidentiality, integrity, and authenticity of the transmitted data. The periodic refresh of keying material helps maintain the security of the communication over time.

Key components and features of IPsec

1. Authentication Header (AH):

1. Provides authentication and integrity for the entire packet.
2. Verifies that the data has not been tampered with during transmission.
3. Typically used when data integrity is a primary concern.

2. Encapsulating Security Payload (ESP):

1. Provides encryption and optional authentication.
2. Encrypts the payload of the packet to ensure confidentiality.
3. May also include authentication to verify the source and integrity of the data.

3. Security Associations (SA):

1. Security associations are established between devices to define the parameters of security services.
2. This includes information such as encryption algorithms, authentication methods, and key management.

Key components and features of IPsec

4. Key Management:

1. IPsec requires a robust key management system to secure communication.
2. Establishing, maintaining, and exchanging keys securely is essential for the effectiveness of IPsec.

5. Modes of Operation:

1. IPsec operates in two main modes:
    1. **Transport Mode:** Only the payload (data) of the packet is encrypted and/or authenticated.
    2. **Tunnel Mode:** The entire packet, including the original IP header, is encrypted and/or authenticated. This is commonly used for VPNs.

6. Diffie-Hellman Key Exchange:

1. IPsec often employs Diffie-Hellman key exchange for the negotiation of shared secret keys between devices.

7. Use Cases:

1. IPsec is used for various purposes, including securing communication between branch offices in a corporate network, providing remote access VPNs for users, and ensuring the security of data transmitted over the Internet.

Key components and features of IPsec

8. Protocols and Standards:

IPsec is based on a set of open standards, and its specifications are defined in documents such as RFC 4301 and RFC 4303.

IPsec is a versatile and widely adopted technology for securing network communications. It adds a layer of security to the Internet and enables secure communication between devices even over potentially insecure networks. It's commonly used in conjunction with other network security measures to create a comprehensive security infrastructure.

Approaches to Implement VPNs

Policy-based VPNs and route-based VPNs are two different approaches to implementing Virtual Private Networks (VPNs) with distinct characteristics.

- **Policy-Based VPN:**

1. Policy Enforcement:

1. **Key Feature:** Policies are defined based on specific criteria such as source/destination IP addresses, protocols, or application types.
2. **Pros:** Granular control over traffic, allowing specific policies for different types of data.
3. **Cons:** Can become complex to manage and scale as the number of policies increases.

2. Tunnel Creation:

1. **Key Feature:** VPN tunnels are typically created based on policies that dictate which traffic is sent through the tunnel.
2. **Pros:** Flexible in specifying which traffic should be protected and sent through the VPN tunnel.
3. **Cons:** As the network grows, managing numerous policies can become challenging.

Approaches to Implement VPNs

3. Routing:

1. **Key Feature:** Routing decisions are often made based on the policies defined for specific traffic.
2. **Pros:** Allows for fine-grained control over how different types of traffic are routed.
3. **Cons:** Complexity increases as more policies are added.

4. Use Cases:

1. **Common Use Cases:** Suitable for scenarios where traffic segregation based on policies is essential, such as enforcing different security policies for specific applications.

Approaches to Implement VPNs

- **Route-Based VPN:**

1. Routing-Based Enforcement:

1. **Key Feature:** Routing tables determine which traffic is sent through the VPN tunnel.
2. **Pros:** Simplifies policy management as routing rules dictate traffic flow.
3. **Cons:** May not provide the same granularity of control as policy-based VPNs.

2. Tunnel Creation:

1. **Key Feature:** The VPN tunnel is created based on routes, without the need for extensive policy definitions.
2. **Pros:** Simplifies configuration and management, especially in large networks.
3. **Cons:** May lack the ability to easily define specific policies for different types of traffic.

Approaches to Implement VPNs

3. Routing:

1. **Key Feature:** Traffic is routed based on routing table entries rather than specific policies.
2. **Pros:** Easier to scale and manage as the network grows.
3. **Cons:** May not offer as much specificity in terms of traffic control as policy-based VPNs.

4. Use Cases:

1. **Common Use Cases:** Suitable for scenarios where simplified management and scalability are important, such as large enterprise networks.

Approaches to Implement VPNs

Comparison:

- **Complexity:**
  - **Policy-Based VPNs:** Can become complex with many policies.
  - **Route-Based VPNs:** Generally simpler and easier to manage, especially at scale.
- **Granularity:**
  - **Policy-Based VPNs:** Offer more granularity in terms of traffic control.
  - **Route-Based VPNs:** May provide less granularity but are often more straightforward.
- **Scalability:**
  - **Policy-Based VPNs:** May become challenging to scale in large networks.
  - **Route-Based VPNs:** Tend to be more scalable and easier to manage in larger deployments.

Proxy

A proxy server acts as an intermediary between a user's device and the internet. When a user connects to the internet through a proxy, their requests are first sent to the proxy server, which then forwards the requests to the internet. The response from the internet is then sent back to the proxy, which forwards it to the user. Proxies can serve various purposes, including:

- **Anonymity:** Proxies can hide the user's IP address from the websites they visit, providing a level of anonymity.
- **Content Filtering:** Proxies can be used to filter or block specific content, websites, or types of traffic.
- **Caching:** Proxies can cache frequently requested content, improving performance by delivering it quickly to users.

Proxy

When accessing a banned website using a proxy, the data indeed passes through the proxy server, but it's crucial to understand how the process works and why it might allow users to access blocked content:

Bypassing Geo-restrictions: Proxies can be used to bypass geographical restrictions imposed on websites. When you access a website through a proxy, the website sees the IP address of the proxy server, not your actual IP address. If the proxy server is located in a region where the website is not blocked, the website may allow access.

DNS Resolution: The Domain Name System (DNS) is responsible for translating human-readable domain names (like www.example.com) into IP addresses. Even if a website is blocked based on its domain name, the DNS resolution can be performed by the proxy server, which may be located in a region where the website is accessible.

Encrypted Connections: Some proxies, especially those that support HTTPS, can encrypt the connection between the user and the proxy server. This encryption can make it challenging for network administrators or ISPs to inspect the content of the traffic and determine the specific websites being accessed.

Proxy

Proxy Server Location: If the proxy server is located in a region where the website is not blocked or restricted, the user can access the website through the proxy server.

Dynamic IP Addresses: Some proxies might use dynamic IP addresses, meaning that the IP address seen by the blocked website changes over time. This dynamic nature can make it more difficult for websites to enforce IP-based bans.

It's important to note that while proxies can provide a degree of anonymity and bypass certain restrictions, they are not foolproof. Some websites employ more advanced techniques to detect and block proxy traffic, and network administrators may implement additional measures to control access.

Additionally, the use of proxies to access blocked content might violate terms of service or local laws, so users should be aware of the legal and ethical implications of circumventing restrictions.

How Proxy Works

When using a proxy to access a blocked website, the process involves sending requests to the proxy server first. The proxy server then forwards these requests to the destination server (the blocked website), retrieves the content, and sends it back to the user's device. Following is the step by step process of: how accessing a blocked website through a proxy might work:

User Accesses Blocked Website via Proxy: The user enters the URL of the blocked website in their browser while connected to the proxy.

Request to Proxy Server: The user's request is sent to the proxy server instead of directly to the blocked website.

Proxy Server Forwards Request: The proxy server forwards the request to the destination server (blocked website).

Destination Server Responds: The destination server sends the requested content back to the proxy server.

Proxy Server Sends Content to User: The proxy server then sends the content of the blocked website back to the user's device.

How Proxy Works

How a blocked website can be accessed through a proxy:

Domain Name Resolution: When the user types the URL of the blocked website, the domain name resolution (DNS) is typically performed by the user's device or the proxy server.

If the website is blocked at the DNS level, users might use a proxy that uses different DNS servers, allowing them to resolve the domain name and access the blocked site.

IP Address and Port Bypass: Some blocking measures are based on IP addresses and ports. By using a proxy, users can circumvent these restrictions because the request is made to the proxy server, not directly to the blocked website.

The proxy server acts as an intermediary, and its IP address is used for communication with the destination server.

How Proxy Works

Note: The effectiveness of accessing blocked content through a proxy depends on the specific blocking mechanisms in place. Some proxies are designed to bypass certain types of censorship, while others may be blocked themselves. Additionally, some countries or organizations may employ more sophisticated methods, such as deep packet inspection (DPI), to detect and block proxy usage.

While proxies can provide a level of anonymity and help users access blocked content, they are not foolproof, and the availability of blocked content can vary based on the specific methods used for blocking and the countermeasures employed by users.

Proxy Vs VPN

1. Encryption:

1. **Proxy:** Typically, proxies do not encrypt the user's internet traffic. Any encryption is limited to the specific protocols used (e.g., HTTPS).
2. **VPN:** VPNs encrypt all internet traffic, providing a higher level of security.

2. Privacy:

1. **Proxy:** Proxies may hide the user's IP address but do not provide the same level of privacy as VPNs.
2. **VPN:** VPNs offer enhanced privacy by encrypting all data, preventing ISPs and other entities from monitoring internet activities.

3. Scope:

1. **Proxy:** Proxies are often used for specific purposes, such as accessing geo-restricted content or filtering web content.
2. **VPN:** VPNs provide a comprehensive solution for securing all internet traffic and are commonly used for general online privacy and security.

What Next?

How to create self signed SSL certificate using OpenSSL
https://www.youtube.com/watch?v=H2LlHOw_ANg

HTTP, HTTPS, SSL, TLS

Definitions to Remember

- HTTP: Hypertext Transfer Protocol
  - Clear text
  - Vulnerable to hackers
- HTTPS: Secure Hypertext Transfer Protocol
  - Encrypted text
  - HTTP over Secure Socket Layer(SSL)
-

HTTP, HTTPS, SSL, TLS

- SSL: Secure Sockets Layer
  - Protocol that’s used to ensure security on the internet.
  - Use public key encryption to secure data.
- TLS: Transport Layer Security
  - The latest industry standard cryptographic protocol.
  - The successor to SSL.
  - Authenticates the server, client and encrypts the data.

Questions