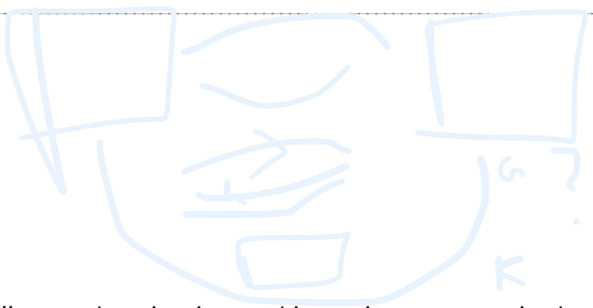
Lecture 7

Class: TLS

Date: 07.58 AM , 24 Nov

Author: Lasal Hettiarachchi

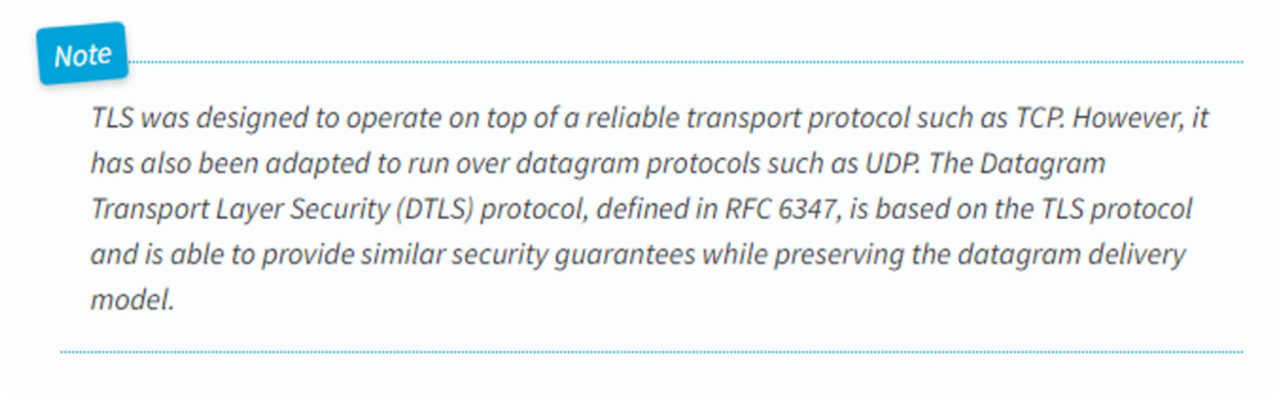


Key learnings:

- TLS handshake
- Foward secrecy

TLS

- SSL developed to encrypt customer data as well as authentication and integrity garuntee in the web
- Implemented at the application layer on top of TCP enabling protocols above it



Note

TLS was designed to operate on top of a reliable transport protocol such as TCP. However, it has also been adapted to run over datagram protocols such as UDP. The Datagram Transport Layer Security (DTLS) protocol, defined in RFC 6347, is based on the TLS protocol and is able to provide similar security guarantees while preserving the datagram delivery model.

- SSL only allows the third parties to only infer the connection endpoints,types of encryption ,frequency of data but not the data inself.
- SSL was later renamed as TSL
- TSL is designed to provide 3 things to the applications running above it: Encryption ,Authentication, Integrity
- We are not required to all 3
- Firstly the server and the client must agree upon a cryptographically secure data channel
- The connection peers must agree upon a **cipher suit** and the keys used to encrypt the data
- This is done through the TLS handshake (The TLS algorithm uses public key encryption or asymetric key cryptography) which allows peers to negotiate a secret key over an unsecured channel. This also allows
- When used in the browser, this authentication mechanism allows the client to verify that the server is who it claims to be (e.g., your bank) and not someone simply pretending to be the destination by spoofing its name or IP address. This verification is based on the established chain of trust

- In addition, the server can also optionally verify the identity of the client — e.g., a company proxy server can authenticate all employees, each of whom could have their own unique certificate signed by the company.
- Finally, with encryption and authentication in place, the TLS protocol also provides its own message framing mechanism and signs each message with a message authentication code (MAC).
- The MAC algorithm is a one way has function

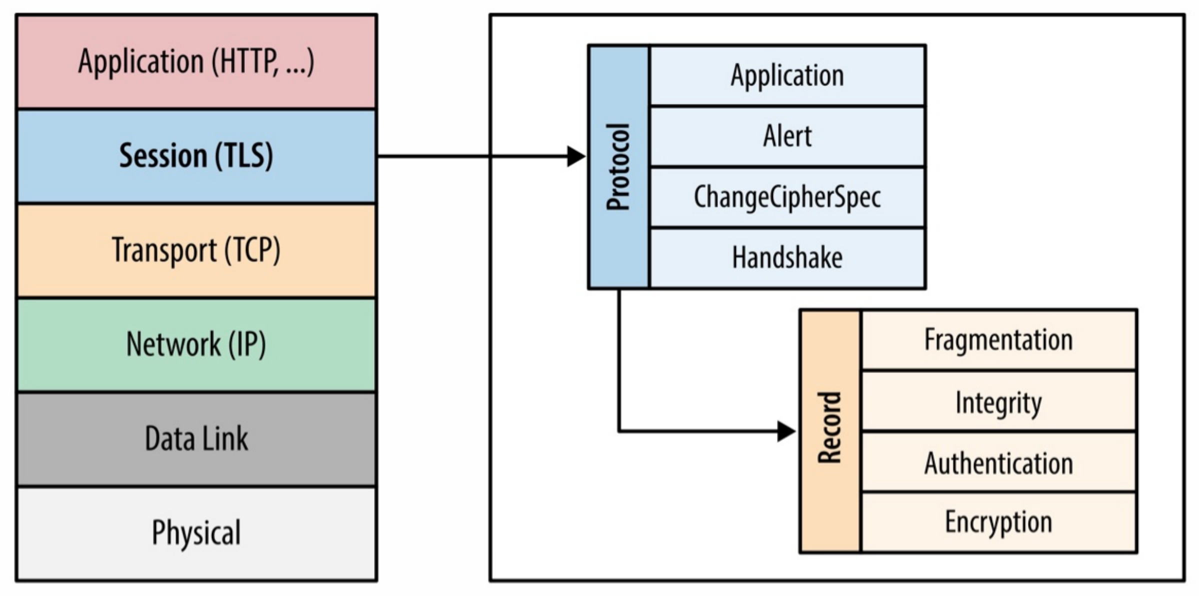


Figure 4-1. Transport Layer Security (TLS)

How TLS provides

Authentication :

•TLS authentication allows the client to verify the server who they clain to be

•TLS verification is based on the chian of trust

•The server can verify the client aswell (Company proxy servers can authenticate all users who have a certificate signed by the company).

Integrity and non-repudiation

•Provides its own message framing mechanism

•Signs each message with message authentication code (MAC)

•Use one way hash functions

Why HTTPS ?

Unencrypted communication (HTTP and other protocols) creates a large number of privacy, security, and integrity vulnerabilities. Interception, manipulation, and impersonation are among them

How HTTP protects integrity

- Encryption prvents intrucers from tampering with the messages and intoducing unwanter malicious content

How HTTP protects user privacy

- Prevents listning on data that are being exchanged. This specially protects user data

How HTTP enables powerful features on web

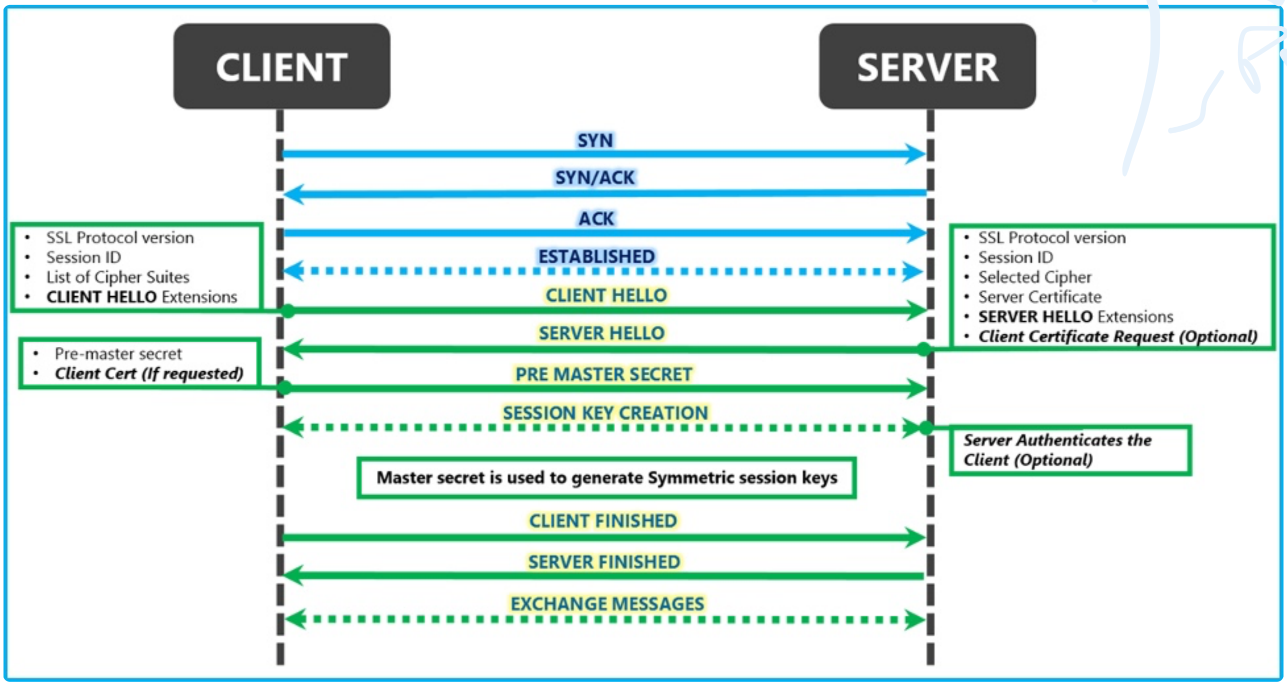
- Features such as users geo loaction, pictures ,videos, enebling offline app experiences require explicit user opt-in that, in turn, requires HTTPS.

TLS Handshake

Before exchanging application data over TLS, the encrypted tunnel must be negotiated. The TLS handshake is only takes place after the TCP handshake

The client and the server must,

- Agree on the version of the TLS protocol
- Choose the ciphersuite(ciphersuite: Algorithms for use in establishing a secure communication connection)
- Verify certificates if necessary.

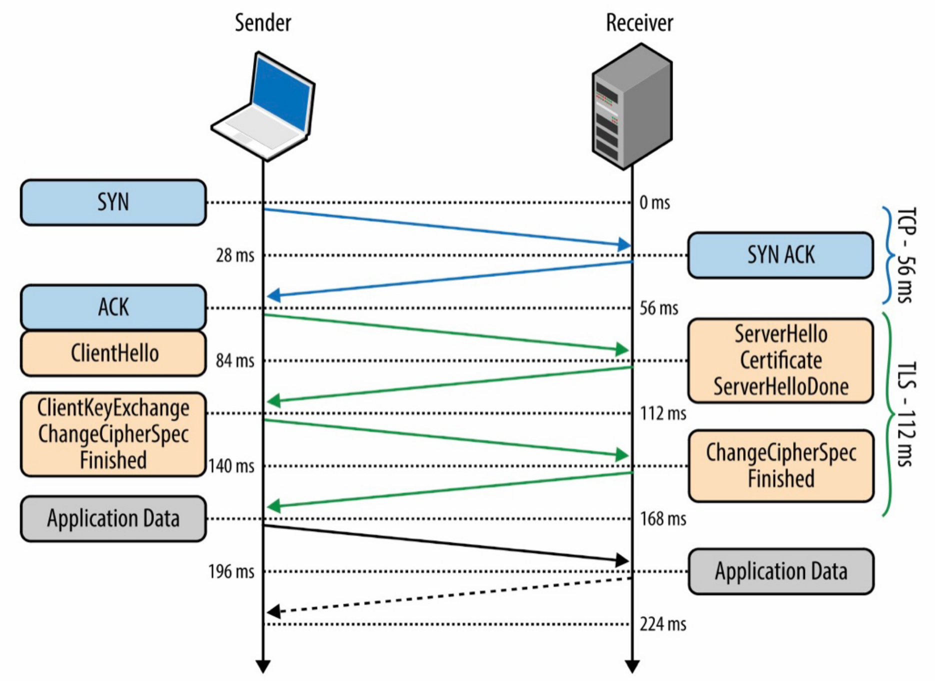


- The client sends " client hello " message (This includes SSL version, clients order of preference, clients supported cipher suites). This also contains a random byte string that is used in subsequent messsages
- Server responds with "Server Hello" . This includes (the CipherSuite chosen by the server from the list provided by the client, the session ID, and another random byte string). The digital certificate is also sent
- If the server requires a digital certificate for client authentication, the server sends a "client certificate request" that includes a list of the types of certificates supported and the Distinguished Names of acceptable Certification Authorities (CAs).
- The SSL or TLS client verifies the server's digital certificate

1. The SSL or TLS client sends the random byte string that enables both the client and the server to compute the secret key to be used for encrypting subsequent message data. The random byte string itself is encrypted with the server's public key.
2. If the SSL or TLS server sent a "client certificate request", the client sends a random byte string encrypted with the client's private key, together with the client's digital certificate, or a "no digital certificate alert". This alert is only a warning, but with some implementations the handshake fails if client authentication is mandatory.
3. The SSL or TLS server verifies the client's certificate. .
4. The SSL or TLS client sends the server a "finished" message, which is encrypted with the secret key, indicating that the client part of the handshake is complete.
5. The SSL or TLS server sends the client a "finished" message, which is encrypted with the secret key, indicating that the server part of the handshake is complete.
6. For the duration of the SSL or TLS session, the server and client can now exchange messages that are symmetrically encrypted with the shared secret key

Unfortunately, each of these steps requires new packet roundtrips (fig belowe) between the client and the server, which adds startup latency to all TLS connections.

- Due to the number of messages that must exchanged the above handshake adds up too 200 ms of overhead
- In TLS 1.3 the messages have been streamlined/optimized to reduce this



Issues With TSL

- The same public-private key pair is used both to authenticate the server and to encrypt the symmetric key. As a result,
    - If an attacker gains access to the server's private key then they can decrypt the entire session.
    - They can still record the encrypted session and decrypt it at a later time once they obtain the private key. (Forward secrecy)

False Start

This is a TLS extensiion which allows clients and servers to start transmitting encypted messages when handshake is only partially complete.

- i.e: Once change cipher spec abd finished messages are sent. But without the other side to do the same.
- This optimization reduces handshake overhead for new TLS connections to one roundtrip

In computer security, a chain of trust is established by validating each component of hardware and software from the end entity up to the root certificate.

Chain of trust and certificate authorities

- Authentication is an integral part of establishing every TLS connection.
- To understand how we can verify the peer's identity, let's examine a simple authentication workflow between Alice and Bob:
    - Both Alice and Bob generate their own public and private keys.
    - Both Alice and Bob hide their respective private keys.
    - Alice shares her public key with Bob, and Bob shares his with Alice.
    - Alice generates a new message for Bob and signs it with her private key.
    - Bob uses Alice's public key to verify the provided message signature.
- Trust is a key component of the preceding exchange. Specifically, public key encryption allows us to use the public key of the sender to verify that the message was signed with the right private key, but the decision to approve the sender is still one that is based on trust. In the exchange just shown, Alice and Bob could have exchanged their public keys when they met in person, and because they know each other well, they are certain that their exchange was not compromised by an impostor—perhaps they even verified their identities through another, secret (physical) handshake they had established earlier!
- Next, Alice receives a message from Charlie, whom she has never met, but who claims to be a friend of Bob's. In fact, to prove that he is friends with Bob, Charlie asked Bob to sign his own public key with Bob's private key and attached this signature with his message (Figure 4-4). In this case, Alice first checks Bob's signature of Charlie's key. She knows Bob's public key and is thus able to verify that Bob did indeed sign Charlie's key. Because she trusts Bob's decision to verify Charlie, she accepts the message and performs a similar integrity check on Charlie's message to ensure that it is, indeed, from Charlie.

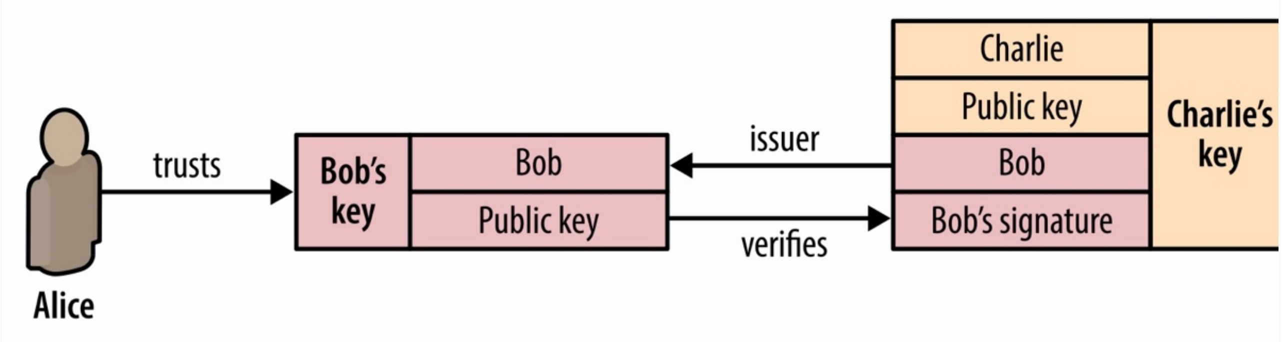


Figure 4-4. Chain of trust for Alice, Bob, and Charlie

This establishes a chain of trust. Authentication on the Web and in your browser follows the exact same process as shown. Which means that at this point you should be asking: whom does your browser trust, and whom do you trust when you use the browser? There are at least three answers to this question:

- Manually specified certificates: It is possible to import certificates that I can trust to the browser. How the import happens and how the verification happens is upto the developer
- Certificate authorities : CAs are a trusted thrid party by both subjects
- The browser and the OS : Every operating system and most browsers ship with a list of well-known certificate authorities. Thus, you also trust the vendors of this software to provide and maintain a list of trusted parties.

Every browser allows you to inspect the chain of trust of your secure connectionusually accessible by clicking on the lock icon beside the URL.

Certificate transparency

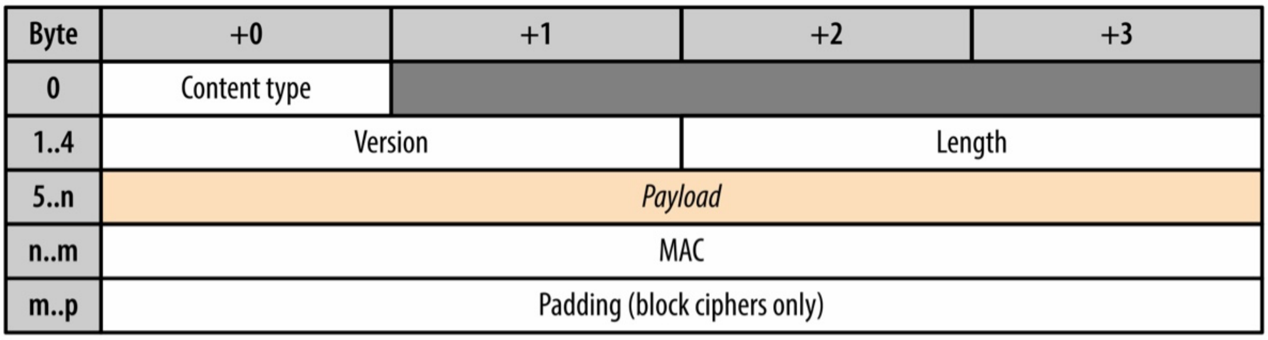
Every OS and browser provides a public listing of all the certificate authorities they trust

Certificate revocation

Occasionally the issuer of a certificate will need to revoke or invalidate the certificate due to a number of possible reasons: the private key of the certificate has been compromised, the certificate authority itself has been compromised, or due to a variety of more benign reasons such as a superseding certificate, change in affiliation, and so on. To address this, the certificates themselves contain instructions on how to check if they have been revoked.

TLS Record Protocol

The TLS Record protocol is responsible for identifying different types of messages (handshake, alert, or data via the "Content Type" field), as well as securing and verifying the integrity of each message.



Byte	+0	+1	+2	+3
0	Content type			
1..4	Version		Length	
5..n	*Payload*			
n..m	MAC			
m..p	Padding (block ciphers only)			

Figure 4-8. TLS record structure