The Secure Web: TLS and HTTPS

Introduction to Computer Security

Naercio Magaia and Imran Khan

Contents

- Diffie-Hellman Exchange
- Secure Sockets Layer and Transport Layer Security
  - Protocol Structure
  - Record Layer
  - Handshake Protocol
- HTTPS

Diffie-Hellman Key Exchange

- First published public-key algorithm
- By Diffie and Hellman in 1976 along with the exposition of public key concepts
- Used in a number of commercial products (SSL/TLS, WhatsApp Signal protocol, etc.)
- Practical method **to exchange a secret key securely** that can then be used for subsequent encryption of messages
- Security relies on **difficulty of computing discrete logarithms**

Modular maths (non-examinable)

- Recall the modulus operator, where $a \bmod q$ gives the remainder when a is divided by q
- Modular arithmetic is where the answers wrap around in a circle
  - $12 + 18 \bmod 9 \ = 30 \bmod 9 = 3$
- Modular exponentiation $a^x \pmod q$ is quickly calculated even if a and x are large
  - $a^x (\text{mod } q) \ = \ (ay(\text{mod } q) * a^z(\text{mod } q)) \ (\text{mod } q) \ \ (\text{where } y + z = x)$
- Modular logarithms are difficult to calculate $\log_a(y) \pmod q$ – the *discrete logarithm problem*

Diffie-Hellman Key Exchange

- Pick (large) prime number q and α such that $\alpha<q$ and α is primitive root to q (there exists a power of α such that all the relatively prime numbers y to q have $\alpha^{z} \pmod{q} = y$). α and q are public
- User A pick X_A such that $X_A < q$ and makes public $Y_A = \alpha^{X_A} \pmod{q}$
- User B pick X_B such that $X_B < q$ and makes public $Y_B = \alpha^{X_B} \pmod{q}$
- The secret key for A is calculated by $Y_B^{X_A} \pmod{q} = (\alpha^{X_B})^{X_A} \pmod{q} = \alpha^{(X_A * X_B)} \pmod{q}$
- The secret key for B is $Y_A^{X_B} \pmod{q}$ - the same number
- And remember that calculating logarithms is hard
- Takeaway is that Diffie Hellman **allows two parties to compute a secret key** whilst publicly passing the necessary information

Diffie-Hellman Example

Have

- Prime number $q = 353$
- Primitive root $\alpha = 3$

A and B each compute their public keys

- A computes $Y_A = 3^{97} \bmod 353 = 40$
- B computes $Y_B = 3^{233} \bmod 353 = 248$

Then exchange and compute secret key:

- For A: $K = (Y_B)^{XA} \bmod 353 = 248^{97} \bmod 353 = 160$
- *For B:* $K = (Y_A)^{XB} \bmod 353 = 40^{233} \bmod 353 = 160$

Attacker must solve:

- $3^{\alpha} \bmod 353 = 40$ which is hard
- Desired answer is 97, then compute key as B does

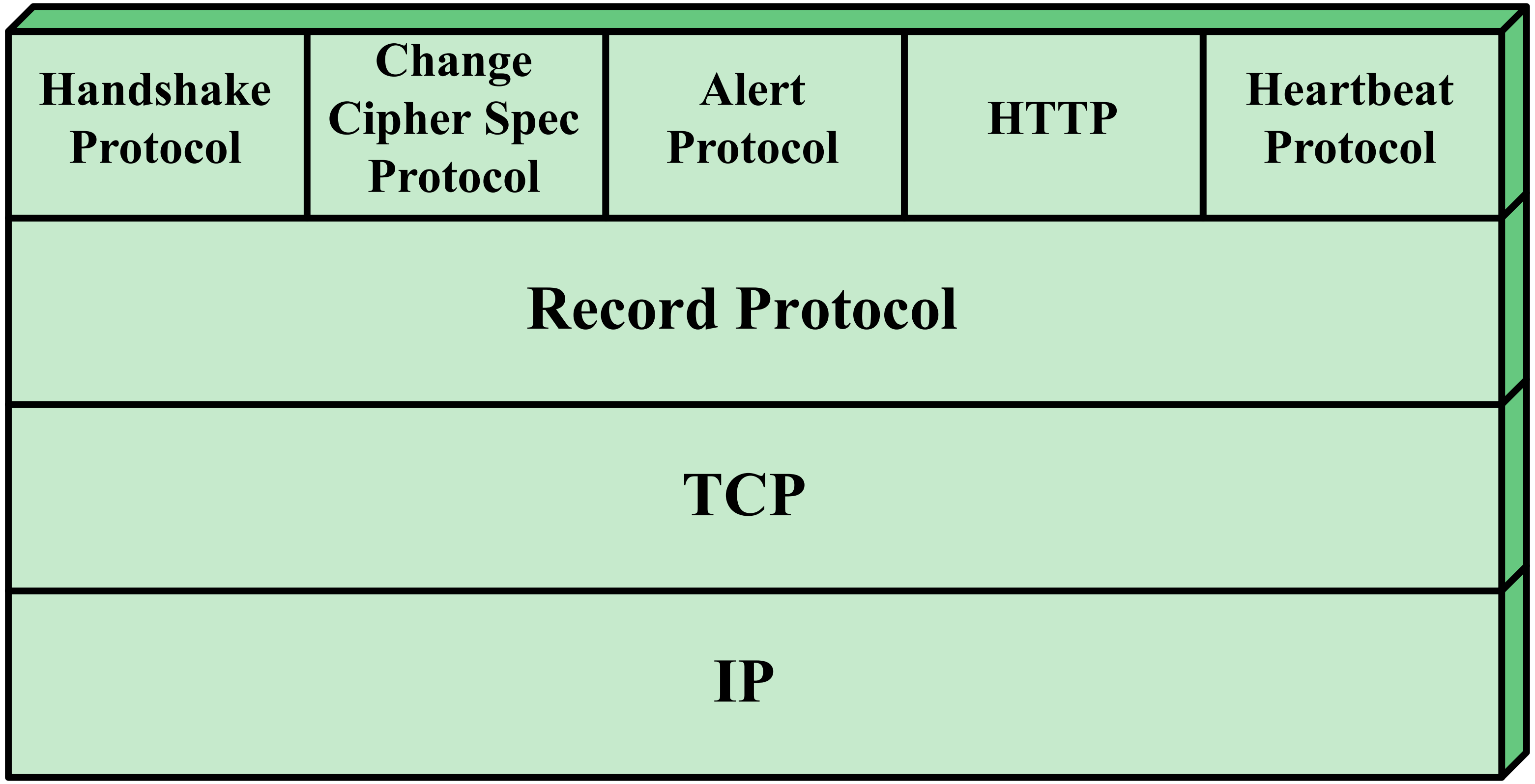
Secure Sockets Layer (SSL) and Transport Layer Security (TLS)

- One of the most widely used security services
- General-purpose service implemented as **a set of protocols** that rely on TCP
- Subsequently became Internet standard RFC4346: Transport Layer Security (TLS)

Two implementation choices:

- Provided as part of the underlying protocol suite
- Embedded in specific packages

SSL/TLS Protocol Stack



TLS Concepts

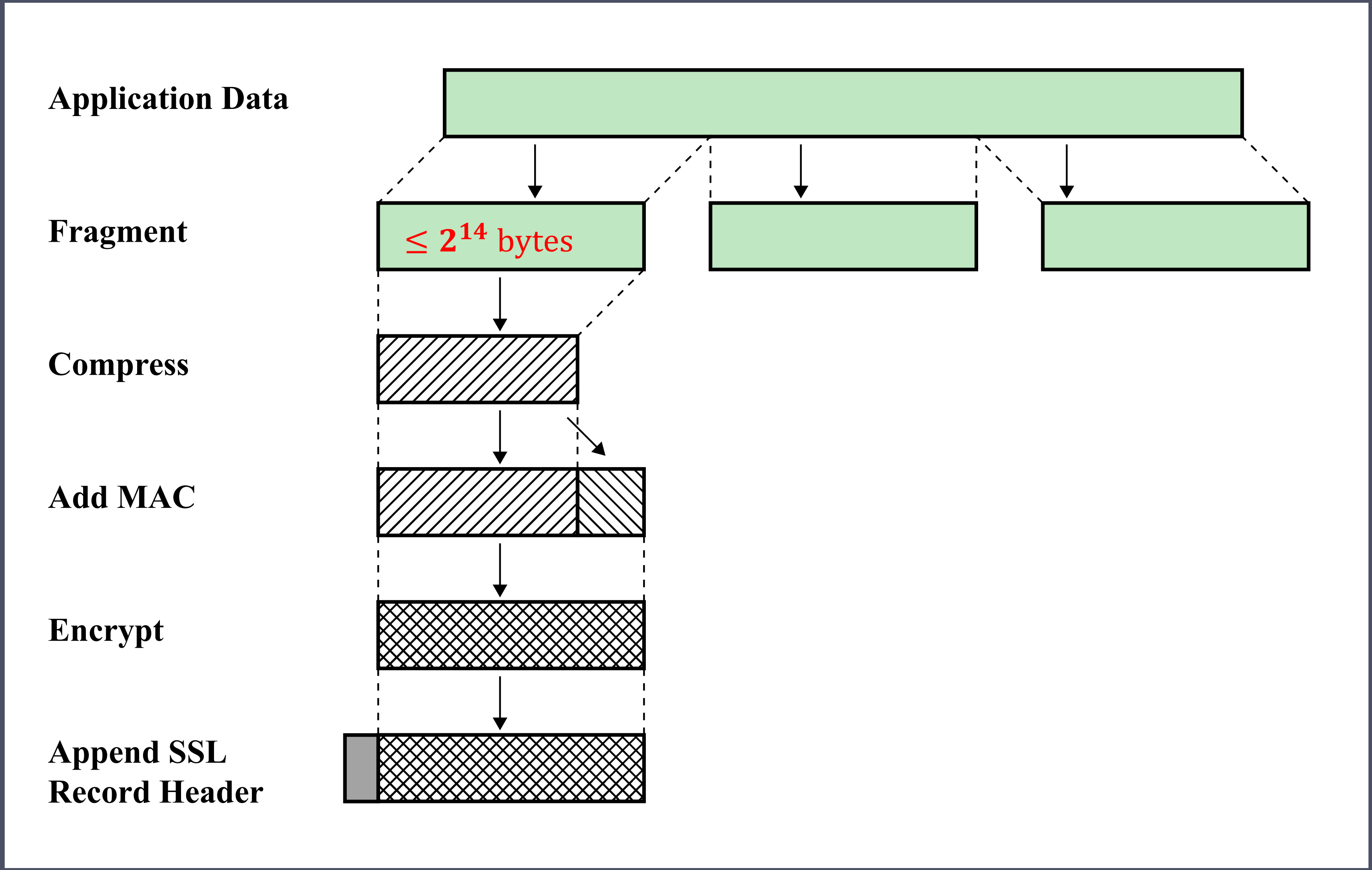
TLS Session

- An association between a client and a server
- Created by the Handshake Protocol
- Define a set of cryptographic security parameters
- Used to **avoid the expensive negotiation** of new security parameters for each connection

TLS Connection

- A transport (in the OSI layering model definition) that provides a suitable type of service
- Peer-to-peer relationships
- Transient
- Every connection **is associated with one session**

TLS Record Protocol Operation



Change Cipher Spec Protocol

- One of four TLS specific protocols that use the TLS Record Protocol
- Is the simplest
- Consists of a **single message which consists of a single byte** with the value 1
- Sole purpose of this message is to **cause pending state to be copied into the current state**
  - Hence **updating the cipher suite to be used in the connection**

Handshake Protocol

- Most complex part of TLS
- Is **used before any application data are transmitted**
- Allows server and client to:

Authenticate each other → **Negotiate encryption and MAC algorithms** → **Negotiate cryptographic keys to be used**

- Comprises a series of messages exchanged by client and server
- Exchange has four phases

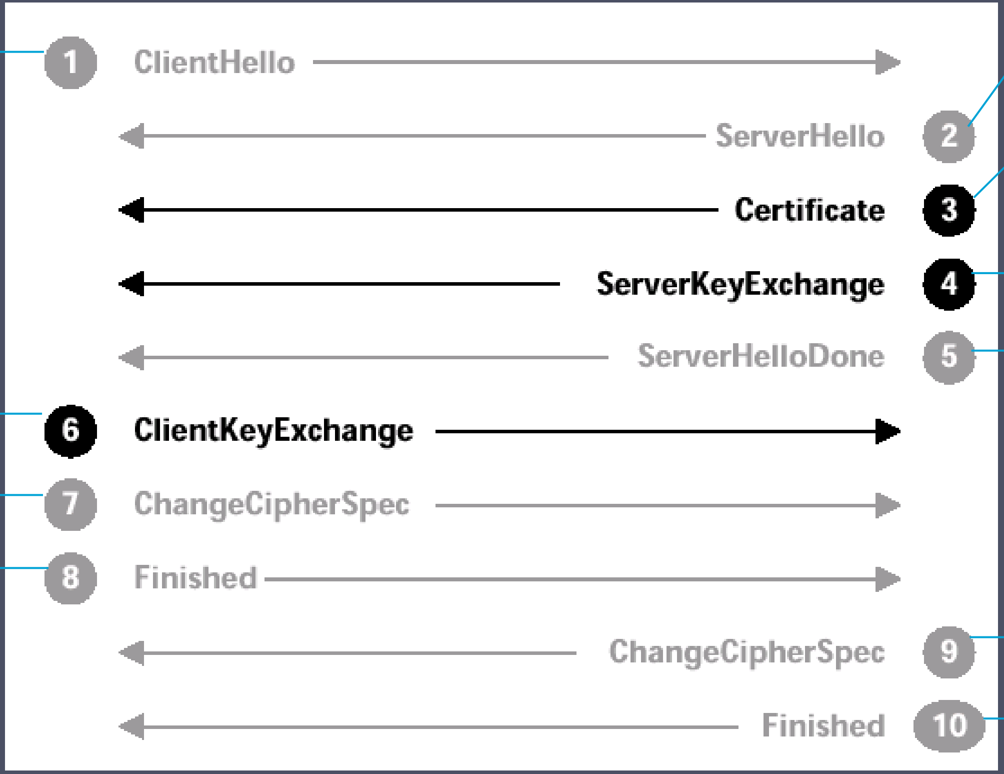
TLS Messages

CLIENT SIDE

SERVER SIDE

Phase 1

OFFER CIPHER SUITE MENU TO SERVER



SELECT A CIPHER SUITE

SEND CERTIFICATE AND CHAIN TO CA ROOT

Phase 2

SEND PUBLIC KEY TO ENCRYPT SYMMETRIC KEY

SERVER NEGOTIATION FINISHED

Phase 3

SEND ENCRYPTED SYMMETRIC KEY

Phase 4

ACTIVATE ENCRYPTION

CLIENT PORTION DONE

(SERVER CHECKS OPTIONS)

ACTIVATE SERVER ENCRYPTION

(CLIENT CHECKS OPTIONS)

SERVER PORTION DONE

NOW THE PARTIES CAN USE SYMMETRIC ENCRYPTION

SOURCE: THOMAS, *SSL AND TLS ESSENTIALS*

Alert Protocol

- Conveys **TLS-related alerts** to peer entity
  - Alert messages are **compressed and encrypted**
- Each message consists of two bytes:
  - First byte takes the value **warning** (1) or **fatal** (2) to convey the severity of the message
    - If the level is fatal, TLS **immediately terminates** the connection
    - Other connections on the same session may continue, but **no new connections** on this session may be established
  - Second byte contains a code that **indicates the specific alert**

Heartbeat Protocol

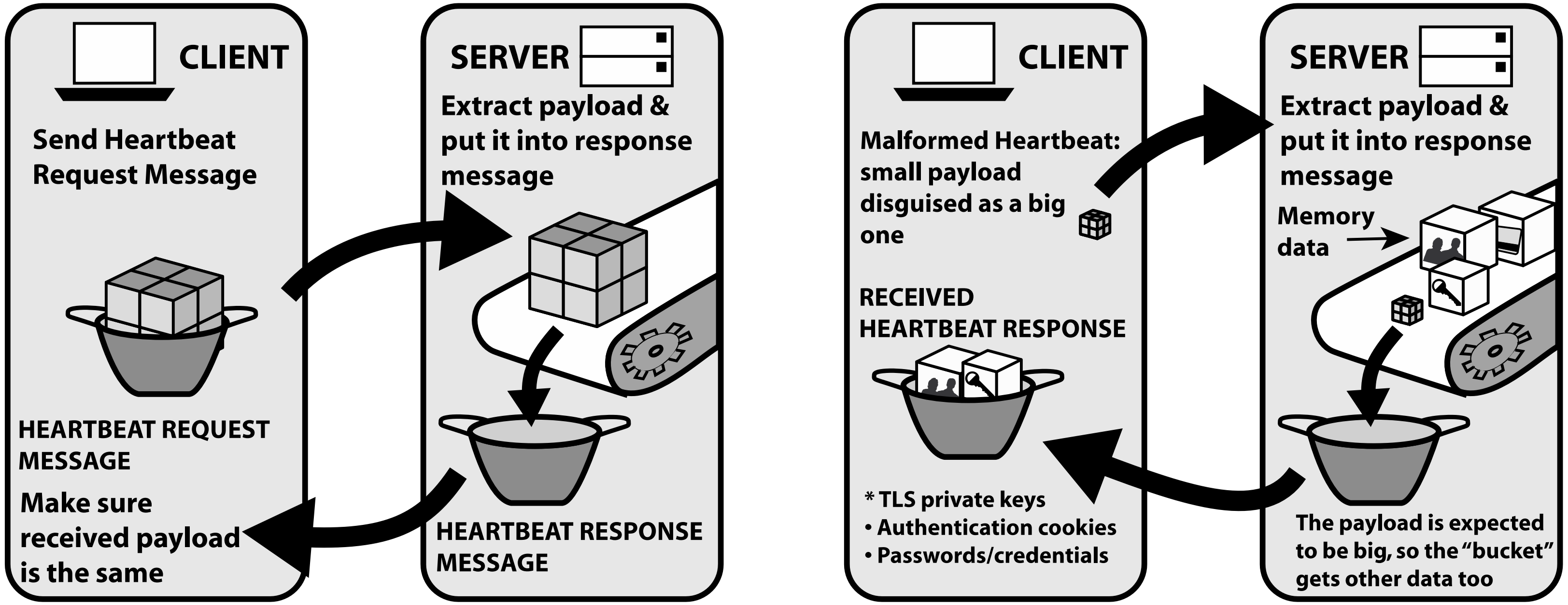
- A periodic signal generated by hardware or software **to indicate normal operation or to synchronize** other parts of a system
- Typically used to monitor the availability of a protocol entity
- Defined in 2012 in RFC 6250
- Runs on top of the TLS Record Protocol
- Use **is established during Phase 1** of the Handshake Protocol
- Each peer indicates whether it supports heartbeats
- Serves two purposes:
  - Assures the sender that the recipient **is still alive**
  - **Generates activity** across the connection during idle periods

SSL/TLS Attacks

Four general categories:

- Attacks on the Handshake Protocol
- Attacks on the record and application data protocols
- Attacks on the PKI
- Other attacks

The Heartbleed Exploit



(a) How TLS Heartbeat Protocol works

(b) How TLS Heartbleed exploit works

HTTP over TLS (HTTPS)

- Combination of HTTP and SSL **to implement secure communication** between a Web browser and a Web server
- Built into all modern Web browsers
  - URL addresses begin with **https://**
- Documented in RFC 2818, HTTP Over TLS
- Agent acting as the HTTP client also acts as the TLS client
- Closure of an HTTPS connection requires that TLS close the connection with the peer TLS entity on the remote side, which will involve closing the underlying TCP connection

Summary

- SSL and TLS
  - TLS architecture
  - TLS protocols
  - TLS attacks
  - SSL/TLS attacks
- HTTPS
  - Connection institution
  - Connection closure