eBOOK

How HTTPS Protocol works

bluegrid.io

The only thing that you absolutely have to know, is the location of the library.

Albert Einstein

Famous Librarian

Introduction

HTTPS (HyperText Transfer Protocol - Secure) is a secured HTTP protocol and has been set as a standard for Web apps any Web accessed service.

The most important aspect of HTTPS is the secure transfer it provides. HTTPS applies the encryption on existing HTTP protocol using the private/public encryption algorithms.

To understand the HTTPS **we need to understand the purpose of the HTTP first.**

HTTP Protocol

HTTP is the protocol for message exchange between server and client. Usually, the HTTP server "listens" on port number 80. For the purpose of testing and local/internal use, it can be changed to meet our customer needs. HTTP is an application protocol that uses TCP (Transmission Control Protocol) for connecting and ensuring communication between server and client.

GET - used to acquire specific content from the server.
HEAD - used to get information about the resource but, not the resource itself.
POST - used to send data to the server for various use. Usually, POST is used to create entries on the server-side.
PUT - used to modify/edit existing entries on the server-side.

HTTP Protocol

DELETE - used to remove existing entries on the server-side.
TRACE - used to get changes made to existing entries on the server-side.
OPTIONS - used to get a list of available request types for an existing resource on the server-side.
CONNECT - used to convert the request connection to a transparent TCP/IP tunnel.
PATCH - similar to PUT, this method partially modifies an existing server-side resource.

HTTP Protocol

Now, every single piece of the data sent or received via **HTTP exists in plain text.**

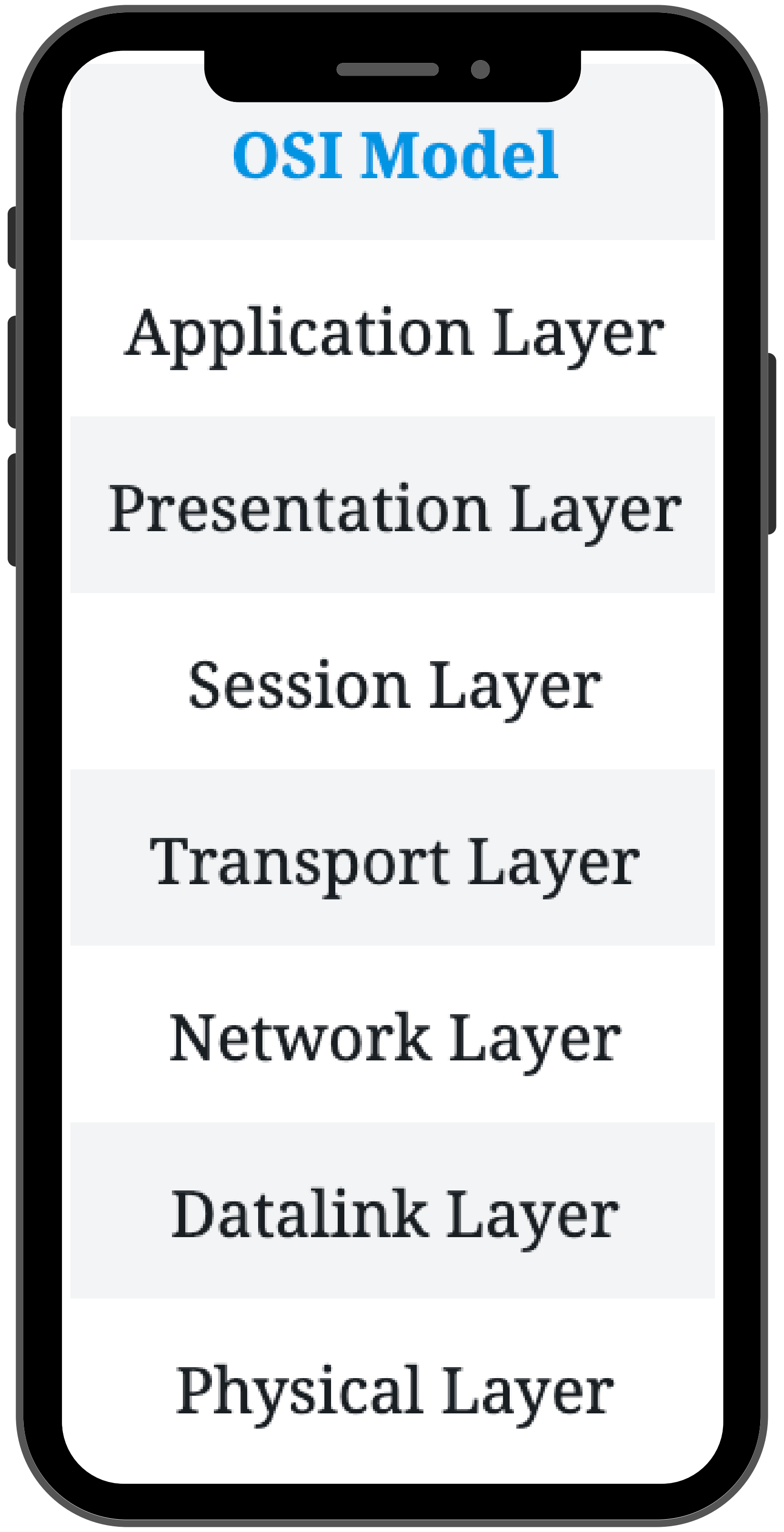
Naturally, it can be accessed without too much sweat. If the HTTP is used for, authentication it is a bad practice because all sensitive data can be seen easily.

Given that **HTTP can be used to control sessions, caching, authentication, etc.** it implies how much it needs leaks without proper security applied.

HTTPS Protocol

So, we have concluded that using HTTP for anything that requires a secure approach is not a good idea.

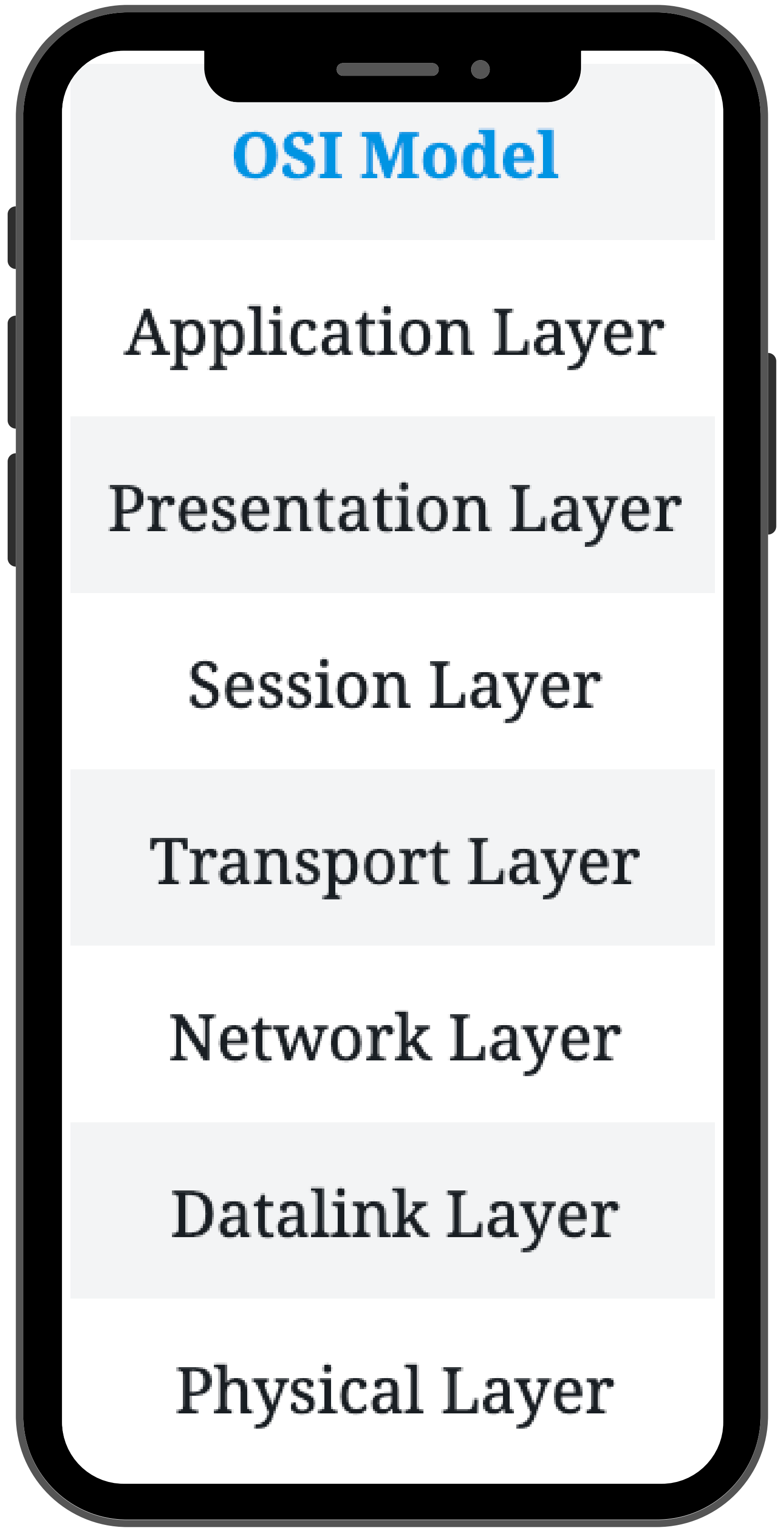
Let's take a look at the **OSI model** first:



HTTPS Protocol

HTTP is an application protocol and as such requires help from Transport Layer to secure the communication.

TLS (Transport Layer Security) that operates on the Transport Layer secures the HTTP in conjunction with three higher layers (Application, Presentation, and Session).



HTTPS Handshake

We have mentioned that HTTPS uses a private/public encryption algorithm for securing the HTTP.

Knowing how private/public key functions we understand that there has to be a certain handshake for server and client to share encryption key.

Encryption key they share will be used to encrypt entire communication between server and client:

HTTPS Handshake

Client Hello

- An indication that the server needs to respond to a client via TLS
- What is supported TLS protocol (version) and cipher settings
- client_random value generated

Server Hello

- An indication that the client needs to talk to the server via TLS
- Sending supported TLS protocol (version) and cipher settings
- Server’s public key (known as a certificate)
- server_random value generated

Auth

- The client authenticates the server’s certificate for Common Name, Issuer and Date
- The client generates a pre-master key for encryption of the communication
- The client encrypts the pre-master key with the server’s certificate and sends the encrypted data to the server

HTTPS Handshake

Master key

- The server decrypts pre-master key with its private key
- The client generates the master key by earlier agreed cipher
- The server generates the master key by earlier agreed cipher

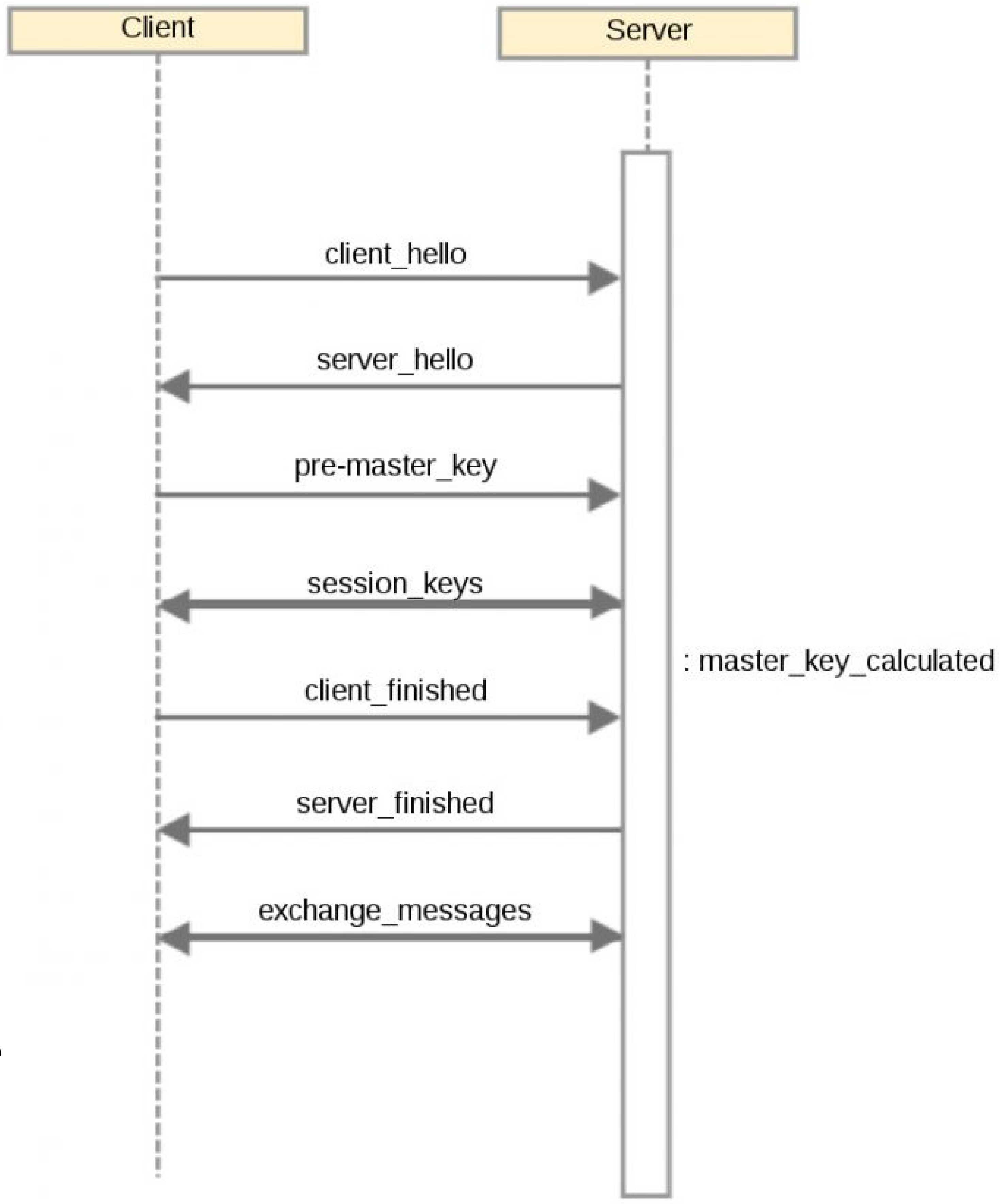
Session keys

- The client generates session key for communication encryption
- The server generates a session key for communication encryption

Encryption

- The client informs server that future communication will be encrypted
- The server informs the client that future communication will be encrypted

On diagram, it looks something like this:



Now, entire communication between server and client will be encrypted and unreadable.

Important note: Before the encryption key is generated the domain this request was initially made for stays unencrypted.

This is the only part of the request that can still be seen if the HTTPS traffic was "listened".

FOLLOW @BLUEGRID ON SOCIAL NETWORKS

OTHER TECH ARTICLES YOU CAN FIND HERE

bluegrid.io