Server-Client Communication Architecture:

The server receives information and the client sends information using ports. The server node initially “opens” a port and starts to accept packets destined to it. The client node has to know what port the server is listening to, and can start communication. It is important to emphasize that as son the data exchange is started on a server/client architecture, such roles disappear and both ends can send and receive data indistinctly. It is a full-duplex communication; therefore both ends can send or receive information.

What is a socket?

A socket is one endpoint of a two-way communication link between two programs running on the network. A socket is bound to a port number so that the TCP layer can identify the application that data is destined to be sent to.

Plugins is code that access hardware, such as camera, GPS, file systems, etc. Plugin may also do other stuff, such as socket programming. A plugin can be viewed as an external library that is not part of the main code. Anyone can write code and make a new plugin for any particular task, it does not have to be specific for hardware.

**Socket Programming**

Using plugins (external libraries) to program sockets that will open and exchange data beween the cell phone and the Raspberry Pi board.

Lab 11 – TLS

Secure Socket Layer (SSL): the first versions of cryptographic protocols that aimed to establish a two-way secure medium.

In 2015, the latest SSL version (3.0) was replaced by another set of protocols, called TLS (Transport Layer Security).

TLS is based on a handshake process that is used to establish a session key that will be used to cipher data on both ends. Basically, the session key encrypts the data and the handshake process creates it. Since it supports different algorithms, the actual types of messages that will be exchanged depend on the implementation on both ends of the communication (client and server). The negotiation of which algorithms to use and type of messages is part of the handshake process.

**Client Hello message:** from a high-level perspective, the handshake starts with a client connecting to a TLS-enabled server to request a secure connection. Then, the client sends the list of supported cipher suites (ciphering and hashing algorithms supported by the client).

**ServerHello message and one Certificate message:** from the list provided by the client (laptop user), the server (google) picks a cipher and hash algorithms that it also supports and notifies the client of the decision (it lets the laptop know what cipher suites is going to use). The server (google) usually also sends back its identification in the form of a digital certificate. This certificate contains the server name (www.google.com), the trusted certificate authority (CA) (the authority that recognizes that the server is google), and the server’s (google) public encryption key (so that the laptop can encrypt the message and only google can decrypt it).

**Client Key Exchange message:** (Integrity) before the client (the laptop) proceeds in establishing communication with the server (google), the client confirms the validity of the certificate (to make sure the laptop is actually getting information from google). In order to generate the session key (used to encrypt the message and, therefore, have a secure connection (Confidentiality), the client (the laptop) usually chooses a pre-master key, which is a random number (the laptop generates a random number). The client ciphers the pre-master key (the laptop encrypts the random number) with the server’s public key (with google’s public key, so only google can decrypt it with its private key) and sends it to the server (google).

**Final Step:** both parties (google and the laptop) then use the random number (pre-master key) to generate a unique session key (it will change every time the session is closed and a new session is started). This unique session key is used for subsequent encryption and decryption of data during the session (the unique session key is used the entire session to encrypt and decrypt data, basically is a shared key). The server (google) usually sends the last message called **New Session Ticket** to finalize the handshake. From now on, all the messages are encrypted using the session key (key generated by the pre-master key, random number, and it is used as a shared key).

**MQTT (MQ Telemetry Transport):** it is a lightweight publish/subscribe messaging protocol. It is useful for use with low power sensors, but it is applicable to many scenarios.

**Describe what CoAP is?**

CoAP: CoAP (client-server communication architecture) means Constrained Application Protocol. It is an Internet Application Protocol for constrained devices. It enables those constrained devices to communicate with the wider Internet using similar protocols.

**What benefits it provides for IoT application development?**

Smaller packets take less bandwidth (data transfer) and require less memory to store. This also reduces the need for complex, time-costly parsing algorithms, which is particularly useful in IoT application development, where memory and time are variables that are always optimized based on the requirements of the device/system. CoAP runs on UDP, a lightweight transfer protocol, and is overall made to be a very efficient application layer protocol. Therefore, embedded devices that are limited in their processing power and memory can employ it.

**What are some of the key differences between HTTP and CoAp?**

* CoAP uses UDP as opposed to HTTP which uses TCP. Using TCP ensures messages are successfully delivered using HTTP while it is not guaranteed using UDP.
* CoAP has less overhead as opposed to HTTP (because you don’t use TCP and it is a lightweight internet application protocol).
* Because you do less processing, CoAP is more energy-efficient than HTTP. It consumes less battery.

**Describe MQTT**

MQTT is a publish/subscribe communication system Application Layer Protocol for sending and receiving data. It is a lightweight messaging (sending and receiving data) protocol.

**What are the benefits it provides for IoT application development?**

The sensor (publisher) sends data to a broker, which stores the data until the subscriber (mobile phone) asks for the data they have subscribed to. The biggest benefit is that this reduces unnecessary cost of sending and receiving data when it is not required using heavy mechanism because the information is just stored in the broker and it updates when the it required by either the publisher or the broker. MQTT is a lightweight application too and serves well in the IoT environment considering memory and power.

**When would you use MQTT instead of CoAP and vice versa?**

The distinction between the two protocols is based on their scale. MQTT works well for many-to-many communication. While it incurs higher overhead because it uses TCP, it is a publish/subscribe protocol that can be run on a broker.

CoAP is more optimized and lightweight because it is based on UDP. It can be run on very lightweight servers. It is best employed in cases where one-to-many communication, or multiplexing, is needed.

**Filters:**

* Low pass: Only allows lower frequency signals to pass through the filter
* High pass: Only allows higher frequency signals to pass through the filter.
* Bandpass: Only allows frequencies within a certain range through the filter.
* Notch: Most frequencies are passed except those around a specific value (high Q value), whice are attenuated.

**How JPEG compression works?**

There are three steps involved in the process:

1. (Down grades blue and red) First, is to down sample the less used variant of colors. Basically, downgrade the colors that are not used. This is due to the fact that if we upscale it the human eye will not be able to notice it much. Thus, in the YCbCr color space, it down samples Cb and Cr channels (blue and red).
2. (It divides the image into an 8x8 grid) Then, it divides the image into 8x8 grid of similar pixels and weighs it on the amount of impact if has on the image and the detail it takes away from the video and then sorts it accordingly to compress.
3. (The 8x8 grid is turned into a linear sequence using a zig zag pattern. Then the RLE algorithm is ran on this zig zag pattern linear sequence.) Finally, this 8x8 grid is then converted into a linear sequence of integers but in a zig zag pattern as opposed to the row or column major. Finally, using RLE algorithm it is compressed.

**Define and distinguish between confidentiality, integrity, and availability:**

These definitions are all concerns in the field of security because they relate how data is packaged, sent, and stored:

1. Confidentiality: Encrypting data so only relevant parties can see it. This does not verify the source. Confidentiality between users or parties.
2. Integrity: Not fraudulent data (actual and legit data), verifying the source (using hash functions), avoiding getting fraudulent or tampered (falsificado) data. Original message is untampered. Integrity of the message.
3. Availability: Data and services and network applications are up and running whenever needed or as intended. Data (information), services (google search), and network applications (email)

**AES (Advanced Encryption Standard)**

The algorithm (AES algorithm) first breaks down the data into a 16x16 grid (JPEG uses 8x8 grid). The 16x16 grid is then xor-ed with the first-round key. Then, it expands this key into multiple different keys. They way it does it is by taking the last columsn of the key rotates it up by one block, and then it runs through the substitution box, with which it is mapped to something else. Now that it is mapped, it is xor-ed again with a round constant.

This generates a new round key which is xor-ed with original key. The key is now shuffled again in a symmetric way and then it’s a column’s bits are shuffled too and this is final key for this round. This is repeated for multiple rounds depending upon the number of bits. The substitution box makes use of XOR and multiplication where the only constants allowed are 0s and 1s and then runs it through two functions, one which makes the math uglier and two which inverses the byte.

XOR let’s you know which bites are different 0 and 0 is 0 and 1 and 1 is 0. Only 1 and 0 or 0 and 1 are 1

**How can I encrypt a message? How do I ensure the integrity of a message using asymmetric key cryptography (public/private key)**

Encrypt using public key of the person who is going to receive the message. To ensure integrity, use a hash function.

First, we hash the data using a secure algorithm like SHA1 and then add it as part of the message with an appropriate header so that the receiver can differentiate what to hash when it receives the data. Next, we encrypt the data (including the hash) using the public key. When the user who it is intended for want to decrypt it, the user can use the private key and then rehash the data to see if the data was preserved.

First step, hash the data using a secure algorithm like SHA1. Then, I will have to add the hashed data into the message with a proper overhead so that receiver can be able to find where the hashed data is located. Next, the data will be encrypted with the has using the public key of the receiver. When the receiver wants to decrypt, it will use its private key and then rehash the hased data to verify the integrity of the message (The source).

**Replay attack? How can it work even if an encrypted channel is used? How can it be prevented?**

A replay attack is when someone captures information (use capture information instead of record information) meant for someone else and repeats (or replays) the sending for malicious purposes (bad intentions). It can come to use even if the data is encrypted because the car, in this case, will use its private key to decrypt its data and when we repeat what the key sent to the car and we do it again, the car will open. It can be prevented using different nonce (random number used one single time) every time someone wants to open the car with the right key.

**If you were hosting a website that you wanted to run https, how would you get a certificate? IF I HAD MY OWN WEBSITE, HOW WOULD I GET A CERTIFICATE?**

I must get certified by a certification authority. I don’t always need to be certified directly by the certification authority. I can be certified directly or indirectly by the certification authority using the hierarchy/chain of trust. The certification can be directly or indirectly using the hierarchy or chain of trust.

Sha1: cryptographic hash function or algorithm. Designed by the US government to serve as a processing standard. It is basically a standord for veriying the integrity of a message. Used for cryptography and data integrity.

RSA (Rivest, Shamir, and Adleman): One of the first practice public key crypto-systems, asymmetric, public key encrypts while private key decrypts. It is one of the first practical public-key systems of cryptography. This system is widely used for secure data transmission.

DDoS: cyber-attack in which perpetrator makes a machine/network unavailable to usrs by disrupting services of a host connected to the Internet. Flooding the targeted machine with requests to overload system and prevent legitimate requests from being fulfilled.