# Chapter 6: Establishing (Secure) Communication using TCP/IP Sockets

## Objective

You need to know:

* What a socket is
* How to make, read and write sockets in WICED
* Security:
  + Symmetric and asymmetric keys
  + X509 certificate
  + TLS (transport layer security)

## Sockets – Fundamentals of TCP Communication

Almost everything on the internet runs using TCP sockets

Think of it as a hose where you put stuff in and it comes out on the other side in order

Identified with two pairs of ip/port numbers

IANA – Internet Assigned Name Authority… [www.iana.org](http://www.iana.org) … all the numbers are here

You already know some standard ports (80,25,53 etc)

Often people just do text based protocols (in fact HTTP is just plain text)

Tell them that they are going to make a TCP client which is going to attach to a TCP server.

They are going to open the socket

Then send text strings through socket

The server is going to respond with text

Explain WWEP using bullet list and examples

## WICED-SDK TCP Server & Client using Sockets

Go through the setup flow … follow the diagram #s/letters

## Transmitting and Receiving Data using Streams

Streams are functions in WICED that allow you to read/write arbitrary amounts of data without worrying about packets

(Remember- packets are the data unit between the network layer and transport layer)

Explain the stream and the firmware flow:

Stream Write: Init -> Write -> Flush -> Deinit (explain the flush)

Steram Read: Init -> Read -> Deinit

Full Stream Write: Create Socket

Bind Socket

Connect Socket

Init Stream

Write Stream

Flush Stream

Deinit Stream

Delete Socket

Explain it is common to create/delete socket to server for each “message” to the server so that the server keeps minimal sockets open

## WICED Socket Documentation

Quick mention of where the documentation is: show the TCP Sockets, Streams and TLS

### Security: Symmetric and Asymmetric Encryption: A Foundation

Explain Symmetric encryption – both sides use the same key – encrypt and decrypt with same key

Explain asymmetric encryption and the magic of the public/private key pair

Keep private key and give out public key to anyone.

Anyone can encrypt data using your public key and only you can decrypt it with private

Only you can encrypt with your private key and anyone can decrypt it with public

Explain the TLS connection setup picture

Explain MIM (man in the middle)

MIM pretends to be the server on initial setup and can steal all the data

Certificate Authorities are used to prevent MIM

How do you know the certificate is valid?

Server takes its pub key and CA pub key and does a cryptographic hash like md5 or sha-256 is just a very fancy checksum … take something big and make it unique and small

Output of the hash is called the digest

Digest is encrypted by the CA’s private key (i.e. only the CA can provide that unique encryption). This is the signature.

The signature is included in the certificate that the server sends to you.

To verify a certificate, you:

Hash the public keys for the server and the CA to recreate the digest

Decrypt the signature with the CA’s public key

Compare the two – if they are the same, the certificate has not been tampered with

Look for the CA’s public key in your known list. If so, then the signature must have been created by the CA since it is the only one that could have created that encryption using its private key

### X.509 Certificates

Certificate contains:

* The site’s public key
* An intermediate authority’s public key
* The root authority’s public key
* The valid DNS domains for this certificate
* The expiration date of the certificate
* One or more secure signatures that let you verify the authenticity of the message
* Other information

Open <https://httpbin.org> and show the certificates in the browser

**In Chrome**: three dots, more tools, developer tools, view certificate, certification path, let’s encrypt, view certificate, Details, Copy to File, Base 64

Open by double clicking and then in emacs or other text editor

Three ways to access/use certificates inside WICED firmware

### TCP/IP Sockets with TLS

TLS is the method to exchange keys, validate keys, and do the encryption

Explain that it is a simple modification to add TLS to a TCP socket

Explain wiced\_TLS\_init\_Identity is for you to send your certificate (optional)

Explain root\_ca\_certificates is for you to check the server certificate

If you open a TLS socket and you don’t check then you are subject to MIM… this is a very common mistake

## Exercise(s)

90 minutes