Small IoT Devices

Well finally time to bite the bullet and see how low we can go in terms of supporting small IoT devices. But first to make things easier I went through the root CA certificates and use a compiler option to only include the most popular “top ten” CAs. When I went through them all I found lots of rather strange ones which I suspect are rarely used. My choice of most popular ones will obviously have to be refined later on. One thing I did notice – no Arab CAs? Might be an idea for some-one in Abu Dhabi to set up such a CA – there will always be a cryptographic need for some sort of “root of trust”, even if its not classic PKI. I noticed that there were lots of what I might call “national” CAs. There is even a Catalonian CA – Marc will be pleased.

I have three candidate devices to hand, all of which support their own independent WiFi.

ESP32

First up is the ESP32 board, the one I have been waving around at recent meetings. Generously endowed with 4M of ROM and 512K of RAM, it uses an Xtensa LX6 processor, and can be clocked from 20MHz to 240Mz. I choose 40MHz. The LX6 processor has a pretty good integer multiplication instruction – a good indicator of how fast the public key crypto will be.

This board also boasts a FREERTOS operating system, which takes up a lot of ROM space (630K) and a small amount of RAM. The TLS client takes up a further 240K bytes of ROM. The overall RAM requirement is for 26K of RAM.

The client is impressively fast. But rather than give timings, its best to do a live demo, which I will do at our next meeting. I suspect that the crypto takes up only a small percentage of the overall time – its mostly I/O.

Arduino Nano 33 IoT

This is a much feebler device. It uses an ARM M0+ processor which has a hopeless multiply instruction. Its the least powerful processor in the ARM range. The board has 256K of ROM and 32K of RAM, and is physically less than half the size of the ESP32. The TLS client takes 192K of ROM, and the 32K RAM proved to be sufficient. This time there is no operating system overhead.

Again it is surprisingly fast, especially considering that it is doing a full handshake followed by a resumption. Based on this experience I suspect that performance of the crypto on the client side is unlikely to be of critical importance.

Fishino Piranha

This nice board is made by an innovative Italian company. It uses a MIPS32 processor, which has a decent multiply instruction. It has 512K of ROM and 128K of RAM. Clocked at only 40MHz, it is again very fast. Note that since we are using no assembly language, we don’t really care about the processor.

Idea

Maybe we should open source the client, and close source the server?

Interesting Fact

Some servers actually include the root CA certificate in their certificate chain, most do not (as it should be found in your root CA store, and there is no need to duplicate it in the chain).

Our current crude algorithm for certificate chain handling, is (a) if the first cert is self-signed, then look no further (whether to accept it or not is a policy decision), (b) Check that the second cert's public key verifies the signature on the first, if not return failure, (c) look no further – if the second cert was not issued by a cert in the root cert store, or if the public key of the root cert does not verify the signature on the second cert, then return failure. Otherwise return success.