Client-side authentication

As is well-known, TLS in the Webserver context just authenticates the server to the client. So you know you are visiting the correct Website. However this does not authenticate the client to the server. And indeed many Websites do not really care who you are. Even on e-commerce sites the only authentication the server is interested in is the authentication that arises from a credit card from which a payment goes through successfully.

But in other contexts client authentication is important. And the method of authentication natively supported by TLS is certificate-based authentication. So the client needs to have a certificate, a supporting certificate chain, and the private key associated with their certificate.

In fact since we are developing both a client and a server, we need the basic capability to authenticate with a certificate, so it makes sense to also support this mechanism from the client side. Problem is that as a developer we now need to test our client against a server which requests a client certificate. And obviously websites don't do that.

A solution is to use the OpenSSL command line tool to set up a server which does request a client certificate. Now this can be easily done. Just use OpenSSL to create a self-signed certificate and private key, and run the server as

```
openssl s_server -tls1_3 -verify 0 -key key.pem -cert cert.pem -accept 4433 -www
```

The only problem here is that we cannot completely control the debug output from OpenSSL, so we are reduced to implementing on the client side and shooting blind at the server hoping that our code is correct. Time-consuming, but ultimately we succeeded. Bouncing `client` off this server now gets the expected correct response.

For now the client certificate and private key are simply embedded in the client code. They could also be easily picked up from files, if a file system exists. There is no private key protection in place, and although passphrase protection could be easily implemented this would not really be viable in an IoT environment. Some kind of secure memory module for storing and using the private key would be best.

Some work was also required on the crypto library side to support X.509 parsing of private keys.

Getting a real certificate

Self-signed certificates are a little unsatisfactory. So I decided to spend a few dollars to get myself kitted out with a "real" certificate. I went to the comodo website and shelled out $100 for a certificate for my own small company. Not so easy – they demanded that my accountant write them a letter confirming my company address, and that my accountant does this by visiting my place of work! Clearly comodo take their work seriously. I am still in negotiations for a certificate, but did not realise just how difficult it would be.