Refactoring the Code

Now that all the hard stuff has been overcome, its time to step back and tidy up the code. However with a deeper understanding of how TLS1.3 works, I am now in a better position to do that tidy up.

And by tidy up, I mean restructuring the code, putting in more comments, introducing some simple data structures.

The example **client.cpp** code now consists primarily of two functions, the first of which does a full TLS1.3 handshake, and the second of which attempts a session resumption.

In both case after the handshake the client sends a standard html GET command to the server, to which the server should respond with a flood of HTML that describes its web page. Once it receives the first blob of HTML, the client exits.

The first full handshake may result in a handshake retry request from the server. The code handles this properly. On resumption this will not happen again as we now know the server's preferences.

On resumption the client tries to send the GET as "early data", but this rarely works as early data is not much supported. If this fails the GET is sent after the resumption handshake has completed.

A log file – **logger.log** – logs in detail the handshake protocol. Therefore the terminal interaction is no longer cluttered with this information. Here is an example

```
./client swifttls.org
Sending Application Message

GET / HTTP/1.1
Host: swifttls.org

Waiting for Server input
Got a ticket
Waiting for Server input
Application data (truncated HTML) =
485454502f312e3120323030204f4b0d0a5365727665723a205377696674544c53
0d0a5374726963
Full Handshake succeeded
... after handshake resumption

Attempting resumption
Sending Application Message

GET / HTTP/1.1
Host: swifttls.org

Waiting for Server input
Application data (truncated HTML) =
485454502f312e3120323030204f4b0d0a5365727665723a205377696674544c53
0d0a5374726963
Resumption Handshake succeeded
Early data was accepted
```

Of course it does not always work. The most common cause of failure is a website that only supports TLS1.2 or lower. That’s about 60% of all websites. And sometimes session resumption fails – I suspect because it is not properly implemented on the server side.