The Record Layer

Time to revisit the lowest layer in the protocol – the record layer. Here messages are broken down into records and transmitted.

Until now we have sent each message in a single record. Which is simple, and allowed. And we have re-used the same buffer for input and output – which of course would be a problem if we wanted full-duplex operation (see an earlier blog). Fortunately TLS is itself a half-duplex protocol, but once a link is established the application may want to switch to a full duplex mode, with separate threads handling input and output.

Note that a full record must be buffered before it can be decrypted and checked for authenticity. That requires memory. TLS1.3 does allow a mechanism for a client to ask nicely that a server reduce its record size, so that a constrained client will find it less resource demanding to consume each record. Up until now our server wasn't really respecting these requests.

For input we already allow for records of any size, because we cannot control what the other party in the protocol may throw at us. But when outputting records, well, we do have control over that.

And there are security implications. The record traffic can be eavesdropped, and the size of each record determined by an attacker. This is called "traffic analysis", and you would be surprised how much can be gleaned that way.

The Fix

To fix all of these issues we create a dedicated output buffer. It does not have to be big, certainly it can be a lot smaller than the input buffer. For our IoT client a size of 1024 bytes seems appropriate. The session also maintains a pointer into this buffer. Once it fills, it is automatically encrypted and transmitted. But we need to be careful – if different messages are encrypted with different keys, then we must force a flush before the key change. When we need to flush we first pad the buffer with nulls to its full length, and then flush it out.

For an observer all encrypted records are now exactly the same length. Traffic analysis comes away empty. And since we are also using a separate buffer for output, full duplex operation is now possible.

Our server now respects client requests for smaller record sizes. For example it may start with an output buffer of 4096 bytes, but it will throttle that back to 1024 bytes if the client sends a **`max_fragment_length`** extension request for that length in its client Hello.