A Surprise..

And I wasn't expecting to be surprised…

After the initial exchange of Hello messages from client to server and server to client, both sides have calculated "handshake" keys, and will use these to encrypt everything that comes next using AES-GCM, a modern AEAD mode of operation. The server, without missing a beat, continues after its hello with its X.509 certificate and some other stuff which it encrypts and transmits to the client. The server then goes on to calculate its final "application" key that will be used to encrypt "application" traffic (so I guess the end is in sight?) This application key is calculated from a hash of the communication transcript to date combined with the handshake key from above.

So I have gotten this far, and my application keys and IVs as calculated by the Server match those from https://tls13.ulfheim.net/, so I am obviously doing it right. Clearly the client can now go on to calculate the same application key (since it will have the same information as the server, same hash, same AEAD key), and we are nearly done. It has been gratifying to observe that all of the cryptographic tools I have needed have been available from https://github.com/miracl/core, and that their output has matched the test vectors from https://tls13.ulfheim.net/.

But wait a minute. At no stage did I use any information from the certificate in the calculation of the application keys! I thought that the session key (aka application key) was supposed to be encrypted by the client using the server's public key extracted from the certificate, and that the server could then decrypt it with its private key? What happened to the Public Key Encryption! What have I missed?

Explained

Well that is how it used to be done with TLS1.2, but in TLS1.3 is done quite differently. In fact the server didn't just transmit its X.509 Certificate (remember that "other stuff" I mentioned?), it also used its private key to digitally sign a hash of the communications transcript to date (including the cert), hence authenticating everything that has gone before. This is called the "Server Certificate Verify" step. And when the client receives this information, it needs to verify that signature using the public key extracted from the certificate. So there is no public key encryption any more in TLS1.3, just public key signature!

Next

Next I need to parse out the server's Cert and signature on the client side, check the AEAD authentication tag, extract the public key from the certificate, verify the signature and go through the identical procedure as on the Server side to calculate the application keys. As my mother used to say, that's just programming.

Peeking ahead

It is interesting that TLS1.3 requires only key exchange and digital signature, and no longer requires public key encryption. This might make it easier to migrate to post quantum primitives, as less primitives are required. Hash based signature might be interesting here…