A proposal

TLS requires 3 passes to complete. The client Hello is not encrypted, and neither is the first part of the server Hello, but as soon as keys are negotiated the rest of the server response is encrypted, as is the rest of the handshake.

The basic idea is that an ephemeral key exchange takes place as early as possible using Diffie-Hellman, and immediately it completes the agreed session key is used for encryption. One of first things encrypted is the server's certificate chain which provides the client with the server's authenticated long-term public key. The server then digitally signs the full handshake transcript and sends the signature to the client. The client verifies this signature using the server's public key. This has the effect of authenticating the original key exchange, at least as far as the client is concerned.

Now in a PQ world this raises a few problems. The original key exchange is manageable – PQ key exchange is probably the easiest and least resource intensive thing that PQ crypto can manage reasonably elegantly. However PQ signatures and PQ public keys can be very large, and this is a problem.

Use a KEM ([1], [2])

For convenience the Diffie-Hellman exchange can easily be replaced by a KEM, a Key Encapsulation mechanism. An ephemeral key pair is generated by the client and its public key sent to the server, who uses it to wrap up a first session key and send it back to the client, who unwraps it. As before the server next sends over its certificate chain, encrypted by this session key, from which the client can extract the servers long-term public key.

But a KEM can also be used in place of the signature. The client uses the server's public key to wrap up in a KEM a second encryption key, and by successfully unwrapping the same key, and combining it with the first key, and using that to encrypt subsequent data, the server is able to implicitly authenticate itself to the client. But this requires sending some data back to the client. You may not have noticed, but this increases the number of passes from 3 to 4. That's not good. In [4] the authors attempt to fix this, but the solution requires pre-shared public keys which leads to complications.

Why can't the client Hello be encrypted?

The client Hello cannot be encrypted with this approach, as the client does not at the start possess a public key from the server to encrypt against. However the community would dearly like the whole handshake, including that original client Hello, to be encrypted [4].

Enter IBE [3]

Identity based encryption has often been suggested as an alternative to PKI, and I think its worth another look, for reasons I will now outline.

One of the problems with TLS is that we can't do any serious encryption until we have possession of the other parties public key. This is the reason that client Hello cannot be easily encrypted. It is also the reason for that extra pass in the KEM based approach discussed above.

With IBE there is no need to wait for a public key. The public key is the identity of the server with which the client is trying to connect. And clearly the client must know this prior to attempting a connection. So if using IBE we can easily encrypt the client Hello.

(A word of warning here. Early encryption like this must suffer from a small defect – it is vulnerable to a replay attack. However the limitations of “early data” are already well understood in the context of TLS. It is unavoidable, but any encryption of the client Hello is better than nothing.)

The handshake now proceeds as follows. First the client Hello is directly encrypted using IBE against the server’s identity. Next the client initiates a KEM by generating an ephemeral key pair, and uses IBE again to encapsulate a first session key using the server identity, and sends this encapsulation along with the ephemeral public key to the server. The server decapsulates this key, and the server wraps a second session key using the provided public key, and sends it back to the client, who unwraps it. Both parties now have both keys, and combine them to derive a shared key. See [3] for a nice diagram.

So using IBE and KEMs in combination we can (a) eliminate (potentially bulky) public keys and signatures, (b) we can keep the number of protocol passes at 3, and (c) we can encrypt the client Hello. Its a win-win-win.

Questions

Can we do IBE using PQC?

Yes we can. There is a well established and very practical lattice-based method from 2014 based on the NTRU trapdoor [5]. I have implemented it.

How about pre-quantum methods?

Yes we could use a pairing-based approach. Note that using the Sakai-Kasahara IBE method the client side calculation does not involve the calculation of a pairing, and so will be as fast as standard elliptic curve cryptography. There is a mistake in [3] where they suggest that the client side computation needs a pairing – it does not. The use of a KEM rather than a signature effectively “turns around” the computation, and shifts the heavier computation from the client to the server.

How about client-side authentication?

Easily accommodated without introducing any new flows. IBE has often been touted as being a more light-weight solution for end-points like TLS clients, compared to PKI.

Key Escrow issues?

Key escrow is intrinsic to IBE. Its Trusted Authority (TA) has a role not dissimilar to the Certificate Authority (CA) of PKI. However a TA can forge private keys directly for any participant. A rogue CA can launch MITM attacks against participants, a rather more difficult attack. However in an IoT setting we would maintain that this is less of an issue.

Does this mean a new TLS?

Schwabe et al [1], [2] have already largely addressed this issue and show that only minor changes would be required. However an IBE-based encryption of client Hello would require something new (as already being being considered in [4])

[1] Schwabe, Stebila, Wiggers https://eprint.iacr.org/2020/534
[2] Schwabe, Stebila, Wiggers https://eprint.iacr.org/2021/779

[3] Banerjee, Chandrakasan “Efficient Post-Quantum TLS Handshakes using Identity-Based Key Exchange from Lattices”
[4] Rescorla et al. “TLS Encrypted Client Hello” https://datatracker.ietf.org/doc/draft-ietf-tls-esni/
[5] Ducas, Lyubashevsky, Prest “Efficient Identity-Based Encryption over NTRU Lattices” https://eprint.iacr.org/2014/794