Post quantum TLS

Already the first proposals are coming in. One of the first is KEMTLS, a proposal from Schwabe, Stebila and Wiggers. Just last week we were lucky enough to have a talk from Peter Schwabe on this proposal. I know Peter of old – he is one of the more energetic, effective and productive cryptographic researchers.

The KEMTLS proposal has two main components. The first is to switch TLS to post quantum primitives. But in order to do this they also propose significantly tweaking the TLS protocol.

Recall that in TLS1.3 the two parties first perform an ephemeral unauthenticated key exchange followed by the Server's transmission of its certificate chain, which includes its public key. The server then digitally signs a hash of the entire communications transcript with the associated private key, which authenticates the original key exchange to the client. Optionally the client may respond with a certificate chain of its own, to provide full two-way authentication.

One of the key motivations in the design of TLS1.3 was to minimize the number of over-and-backs (or "passes") in the protocol, and to allow transmission of application data as early as possible. There is a price to be paid for each pass in terms of latency increases and communication delays.

The main cryptographic operations that will need PQ replacement will be

1. The original key exchange
2. The processing of the certificate chain
3. The signature of the transcript and its verification

The Key exchange

This is probably the easiest to deal with, and we have lots of PQ choices. Myself I quite like SIKE for this, due to its small key sizes. I know its quite compute intensive. But in an IoT setting hardware support for compute intensive calculations is often available. And there are faster lattice based alternatives, although the key sizes do increase a lot.

The Certificate chain

There seems to be an assumption that drop-in replacements of PQ primitives into PKI will be used to provide PQ certification. That means that PQ signature schemes will have to be supported, with primitives for signing and verification. So there is no escaping the requirement for an acceptable PQ signature scheme. And here things becomes a little more challenging, as the current PQ signature proposals leave a lot to be desired when compared with their pre-quantum equivalents. Elliptic curve cryptography has spoiled us when it comes to fast compute time, and small key sizes! But there is no avoiding it – if we are going to stick with PKI, then we are going to need a good PQ signature scheme. This almost certainly means larger certificate chains, with bandwidth implications that will probably dwarf other aspects of the TLS handshake.

The Transcript Signature

On the face of it this would appear quite straightforward. Use one of the signature schemes which we already must support for PQ certificate PKI signature. But it is here that KEMTLS proposes a radical change – replace the transcript signature with a KEM (Key Encapsulation Mechanism). The motivation is that currently PQ KEMs are much more efficient than PQ signature. Note that a KEM is a standard method for doing the original key exchange. But its use as a substitute for the

transcript signature is a lot more controversial. And the bad news is that it requires one or more extra “passes”.

I just don’t buy it

One of the issues raised by Peter was the size of the TCB, or Trusted Computing Base. Basically it makes sense to minimise ones trust exposure, or trust “surface”. The less components that need to be trusted, the more secure is your system. The TCB also affects the size of the implementation. Digital signature requires code for performing the signature, and distinct code for verifying it. In a TLS setting the Server may only need to do signature, and the client may need to only do verification. Part of the case for KEMTLS is that only certificate authorities will have to do signatures, and if KEMTLS were adopted then online signatures by the server would not be required. It is claimed that “signatures are only generated in the more confined and secured environment of certificate authorities”. But if signatures can be safely carried out by a CA, why exactly are they a problem for servers? I just don’t get it.

And those extra passes are definitely a performance problem. One mitigation is to use pre-distributed public keys. But this sounds complicated to me, especially for a poorly resourced client in an IoT setting.

A lot of thought and years of effort has gone into the design of TLS1.3. I would be very slow to dump all of that without a much more compelling case being made. I also would be aware that the OpenSSL “keepers” of the TLS/SSL flame are a pretty conservative bunch, so I would expect a lot of push-back from that quarter.

I remain to be convinced that the KEMTLS proposal is superior to the simpler recourse of just dropping PQ primitives into the existing TLS1.3 structure. Can’t wait to start trying it out!