ML-KEM for TLS 1.3

draft-connolly-tls-mlkem-key-agreement

https://datatracker.ietf.org/doc/draft-connolly-tls-mlkem-key-agreement/
https://github.com/dconnolly/draft-connolly-tls-mlkem-key-agreement
A pure-PQ ciphersuite for TLS 1.3

- No purely post-quantum ciphersuites
- Fills in the other side of draft-ietf-tls-hybrid-design
- ML-KEM-1024 supports TLS users who need to comply with the CNSA 2.0 draft: no hybrid, NIST security level V, FIPS-approved algorithm
- Clean key agreement, no hybrid duplicate shares or mixing and matching logic, if PQ-only works for your applications
New NamedGroups: MLKEM768, MLKEM1024

```c
enum { 

..., 

    /* ML-KEM Key Agreement Methods */ 
    mlkem768(0x0768),
    mlkem1024(0x1024) 

..., 

} NamedGroup;
```
Client sends encaps key, server replies with ciphertext

```c
struct {
    NamedGroup group;
    opaque key_exchange<1..2^16-1>;
} KeyShareEntry;

These are transmitted in the `extension_data` fields of `KeyShareClientHello` and `KeyShareServerHello` extensions:

```c
~~~
struct {
    KeyShareEntry client_shares<0..2^16-1>;
} KeyShareClientHello;

struct {
    KeyShareEntry server_share;
} KeyShareServerHello;
```
KEM shared secret is input to Handshake Secret derivation

```plaintext
shared_secret -> HKDF-Extract = Handshake Secret

 Derive-Secret(., "derived", "")

 Derive-Secret(., "derived", "")

 Derive-Secret(., "derived", "")
```

5

FAQs

Should this be Recommended = Y?
I don’t think so, at least not yet. It should be optional

What about PQ signatures too?
Let’s do key agreement in one document and signatures in another: the signatures are big and may be much harder to design into TLS 1.3 than KEM key agreement.

Isn’t this too early?
Considering the long timelines for adoption, I don’t think so

Just use hybrid!
Some users cannot use hybrid, and some will not do more than one PQ transition. Having a PQ-only option seems necessary eventually, let’s make a start

I don’t trust PQ crypto, it’s too young!
CRYS'TAL'S-Kyber was published 7 years ago, LWE schemes are older than that. Elliptic curves were first published in ~1985, wg adopted for TLS in 1998, NIST curves standardized in 1999, RFC in 2006! We’re older and wiser now, but even so this timeline seems in line with major crypto assumption changes in TLS in the past.
Questions?
ML-KEM for TLS 1.3

draft-connolly-tls-mlkem-key-agreement

https://datatracker.ietf.org/doc/draft-connolly-tls-mlkem-key-agreement/
https://github.com/dconnolly/draft-connolly-tls-mlkem-key-agreement