

# Hybrid key exchange in TLS 1.3

draft-stebila-tls-hybrid-design

<https://dstebila.github.io/draft-stebila-tls-hybrid-design/>

# Motivation and Goals

- Multiple sources of interest in using multiple key exchange algorithms simultaneously as part of transition to post-quantum crypto
  - Several Internet-Drafts already:
    - TLS 1.2: Schanck, Whyte, Zhang 2016; Amazon 2019
    - TLS 1.3: Shack, Stebila 2017; Whyte, Zhang, Fluhrer, Garcia-Morchon 2017; Kiefer, Kwiatkowski 2018
  - Experimental implementations: Google CECPQ1, CECPQ2; Open Quantum Safe; ...
- Need PQ key exchange before we need PQ authentication because future quantum computers could retroactively decrypt, but not retroactively impersonate
- **Goal: develop experimental framework in which key exchange in TLS 1.3 can be extended with additional keyshares**

# Non-Goals

- Selecting or specifying one or more post-quantum algorithms to actually use in TLS

# Design Parameters

1. How to negotiate which combination of algorithms to use?
2. How many algorithms can be combined?  
(2? More than 2?)
3. How should public key shares be transmitted? Combined, or individually?
4. How should the shared secrets be combined?

# Evaluating Designs

- Backwards compatibility?
  - Hybrid-aware client, hybrid-aware server
  - Hybrid-aware client, non-hybrid aware server
  - With middle-boxes
- Extra round trips?
- Duplicate information?
- Changes to TLS state machine or processing logic?

# Negotiation

## Negotiate each algorithm individually

- Extend the NamedGroup enum to include identifiers for each individual algorithm

Options:

1. Send two lists of algorithms (2nd list in extension) [SCHANCK]
2. Send all algorithms in one list, with some external (IANA) mapping onto traditional vs. next generation
3. Insert divider in the supported\_groups extension to delineate the “first” list and the “second” list

## Negotiate combination together

Options:

1. Add NamedGroups for every desired combination [KIEFER, CECPQ1,CECPQ2]
2. Use NamedGroup markers combined with an extension to negotiation combinations [WHYTE 1.3]
3. Use delimiters in supported\_groups extension

Some choices affect backwards compatibility,  
add processing logic,  
or result in sending duplicate information

# Key Combination

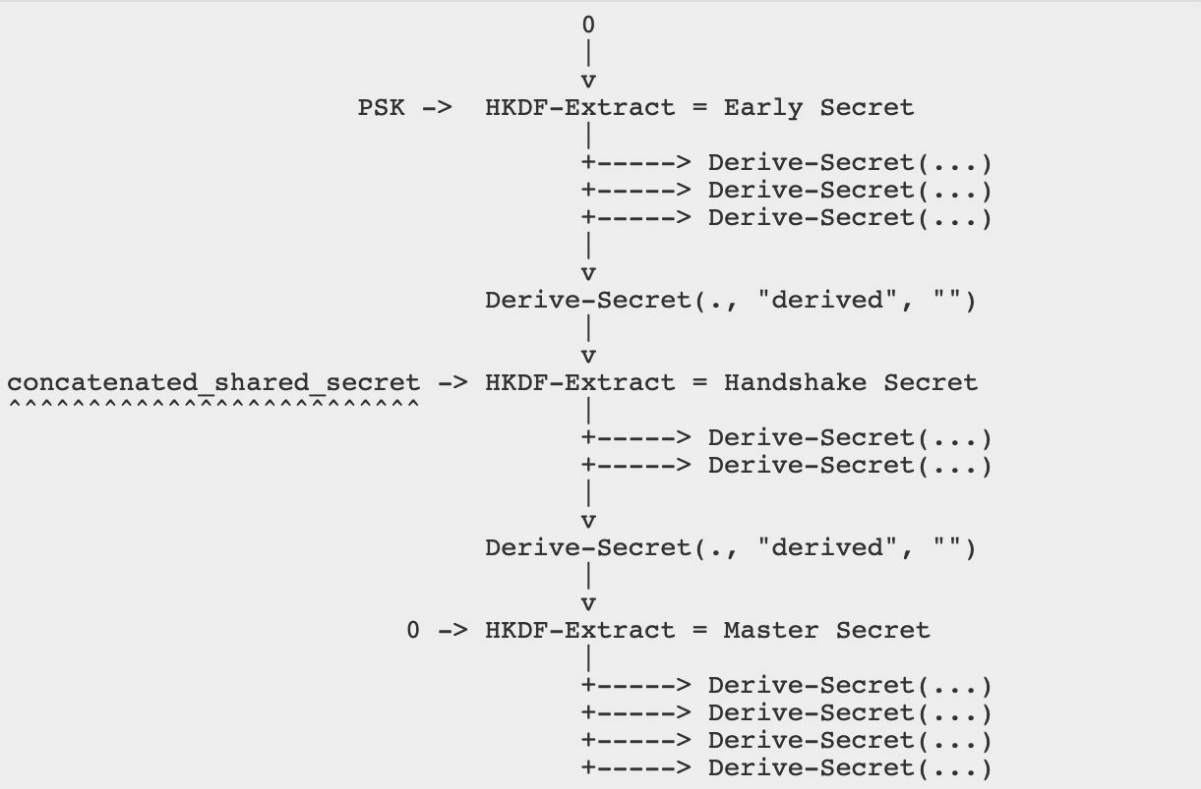
**Top requirement:** needs to provide “robust” security:

- Final session key should be secure as long as at least one of the ingredient keys is unbroken
- (Most obvious techniques are fine, though with some subtleties; see Giaccon et al. PKC 2018, Bindel et al. PQCrypto 2019, ....)

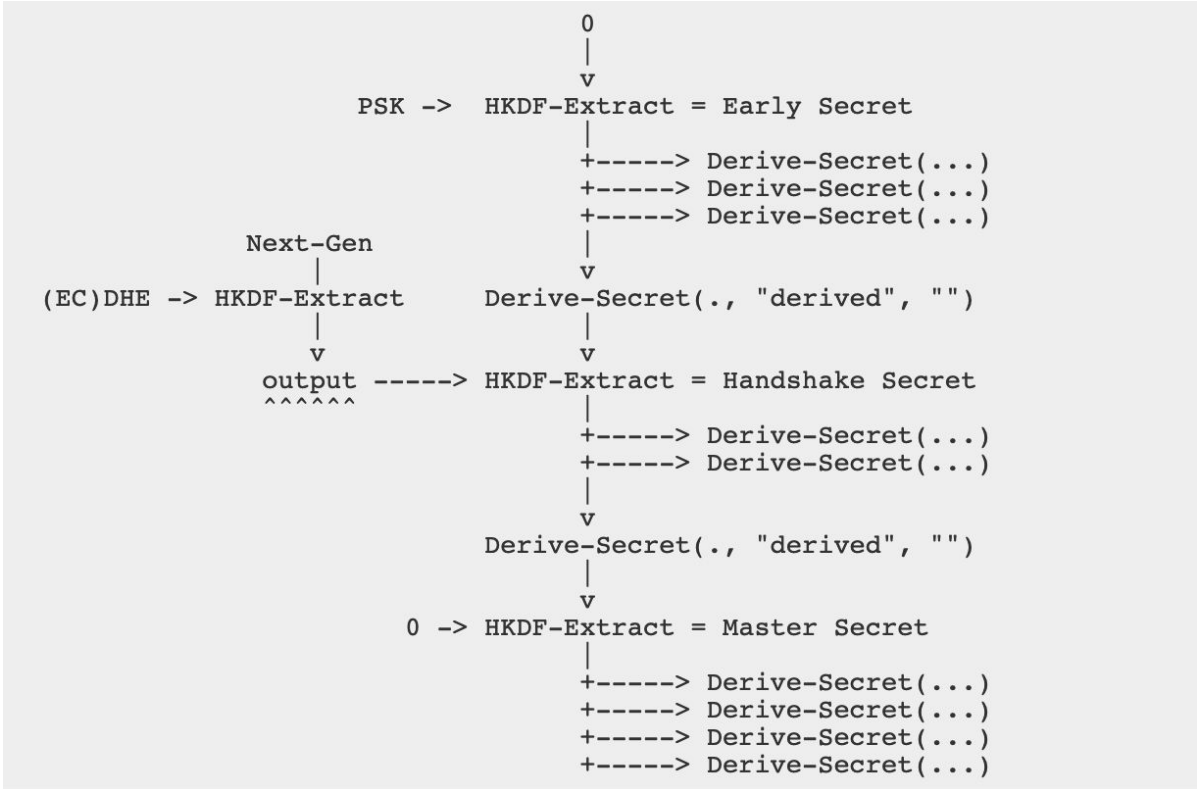
Options:

1. Concatenate keys, then feed directly into TLS 1.3 key schedule
2. KDF keys together, then feed that into key schedule
3. XOR keys together, then feed directly into key schedule
4. Add new stage of key schedule for each key
5. Stick 2nd key into a (hopefully unused?) “0” spot in the key schedule

# Key Concatenation



# Nested KDFs





# New Key Schedule Stage



# Technical Questions

Will any KEM suffice?

- Passive-secure CPA KEMs not okay with key share reuse
- Actively-secure CCA KEMs more robust but more expensive
- How to deal with KEMs which have a non-zero probability of failure?

What metrics should we use to evaluate different designs?

# Procedural Questions

1. Should the WG have a document that describes the different hybridization parameters, focus on producing a document that describes one specific design, or neither?
2. Should the WG produce a set of requirements for KEMs that would be needed for any viable key exchange algorithm?

**Questions?**

# Hybrid key exchange in TLS 1.3

draft-stebila-tls-hybrid-design

<https://dstebila.github.io/draft-stebila-tls-hybrid-design/>