Semi-Static Diffie-Hellman Key Establishment for TLS 1.3

`draft-rescorla-tls-semistatic-dh-02`

Eric Rescorla, Nick Sullivan, **Christopher A. Wood**

IETF 106 - TLS WG - Singapore

Overview

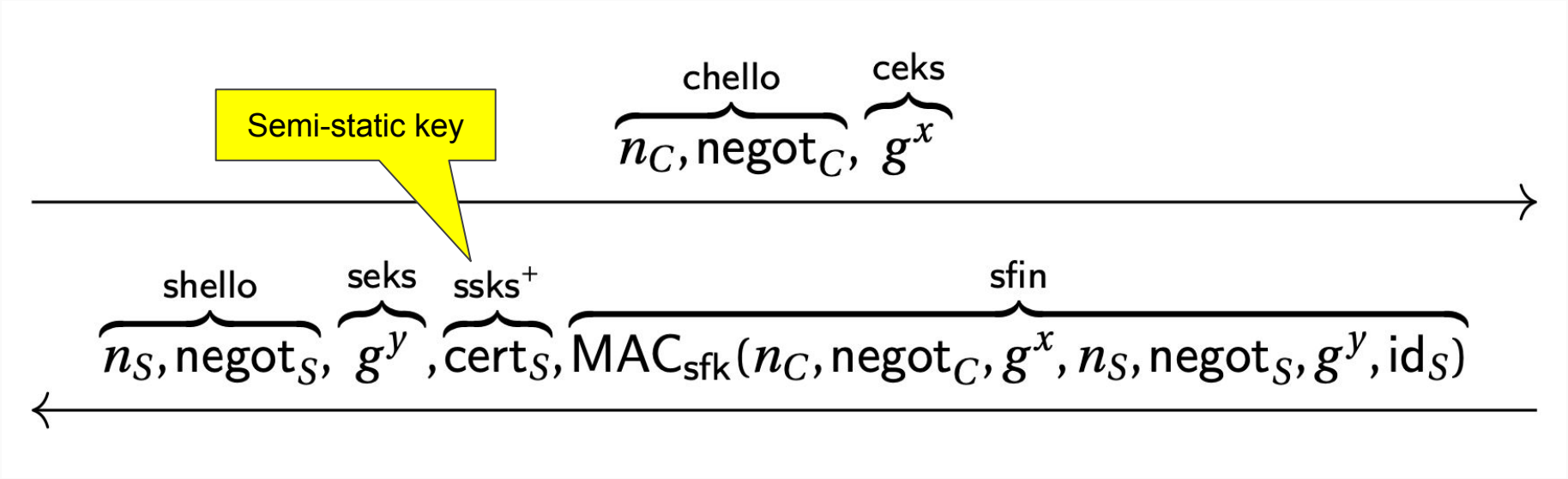
Existing TLS 1.3 key schedule is similar to OPTLS, supporting:

- 1-RTT, PSK-based resumption, ECDHE+PKE-based resumption

Missing: **1-RTT semi-static** mode

- Server supplies signed semi-static key share (in a delegated credential or certificate)
- Peers generate static secret from client ephemeral and server semi-static key shares
- Replace `CertificateVerify` with a MAC computed from the static secret

Semi-Static Flow



Krawczyk, Hugo, and Hoeteck Wee. "The OPTLS protocol and TLS 1.3." *2016 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2016.

Motivation

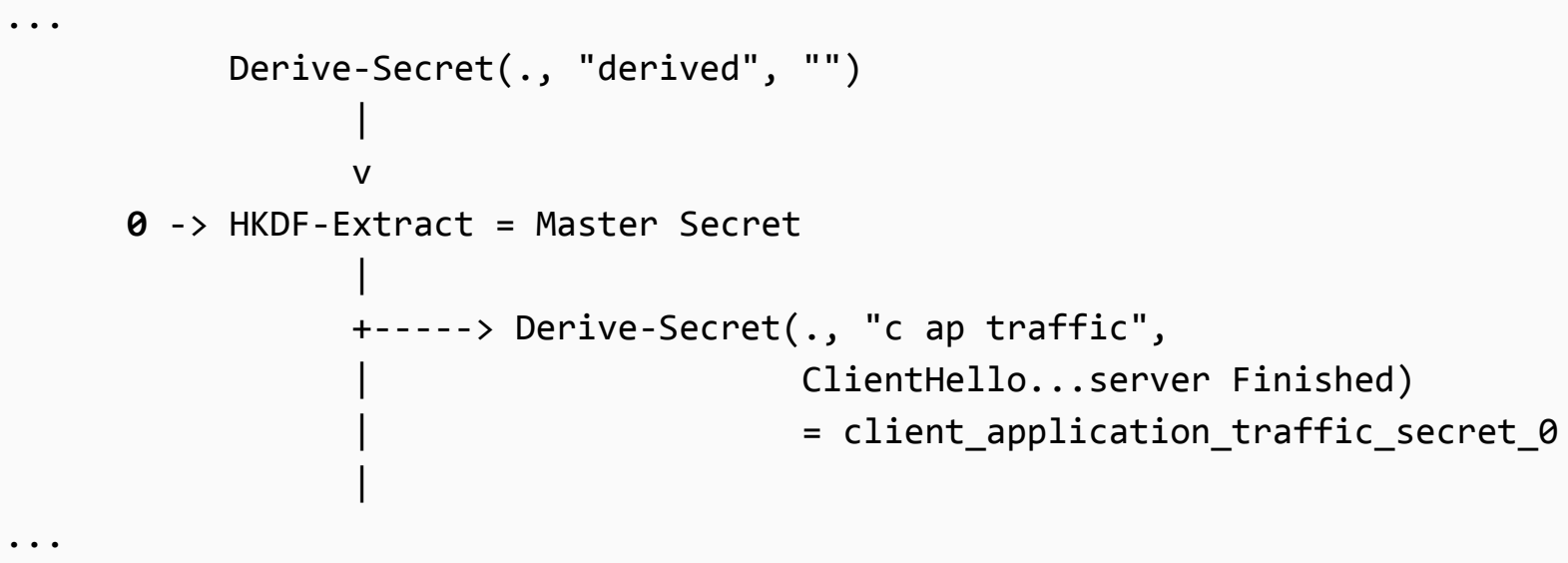
Single-primitive protocols

- Some implementations have code size restrictions
- Server only needs Diffie-Hellman, client can get by with DH-only + key pinning

Lighter key exchange variant (e.g, for LAKE)

Mixes long-term server key into master secret

TLS 1.3 Non-Semi-Static Key Schedule



TLS 1.3 Semi-Static Key Schedule

Negotiation Details

Use a new signature scheme

```
enum {
   sig_p256(0x0901),
   sig_p384(0x0902),
   sig_p521(0x0903),
   sig_x52219(0x0904),
   sig_x448(0x0905),
 } SignatureScheme;
```

These values **MUST NOT** appear in "signature_algorithms_cert"

Open Questions

Should we keep a CertificateVerify message, or just have Finished?

Should client authentication happen the same way?

Should we also support 0-RTT? Would the client ephemeral and server static secret be a PSK?

Diffie-Hellman group must support multiple uses of the same key (PQC?)

Next steps

Seeking adoption

- Is this work that people are interested in pursuing?
- Are there people willing to review the draft?

Semi-Static Diffie-Hellman Key Establishment for TLS 1.3

`draft-rescorla-tls-semistatic-dh-02`

Eric Rescorla, Nick Sullivan, **Christopher A. Wood**

IETF 106 - TLS WG - Singapore