cTLS

draft-rescorla-tls-ctls-03

Eric Rescorla
Richard Barnes
Hannes Tschofenig

-03 update

Mostly just editorial

Focus on the template-based specialization and the DH-based handshake

… This is the one where there is the most bloat

Template system is generic

Can add new specialization keywords in the future

Varint Encoding

cTLS introduced a varint encoding

It doesn't actually save much space

… Especially with template-based specialization

But it's a hassle, especially for existing TLS 1.3 stacks

Proposal: Remove varint encoding

Semi-Static DH

1 or 2 x CertificateVerify containing a MAC instead of a digital signature
E.g. 20 bytes (HMAC-SHA256) instead of 64 bytes (P256r1)

TLS 1.3 focuses on signature-based (SIGMA-style) modes

Static DH-based (OPTLS-style) modes can be more compact

Existing (but slow) work on this for TLS 1.3 (draft-ietf-tls-semistatic-dh)

cTLS can just inherit this

Proposal: Do nothing cTLS specific
(but resurrect expired draft)

Point Compression

2 x for the omitted y coordinate
(depending on the key size)

TLS 1.3 ECC public keys use uncompressed mode (for NIST curves)

This is compatible with what people were doing

… but wastes space

Easiest plan: add new code points for compressed curve

Do people want this? Volunteers?

Proposal: Do nothing cTLS specific

Finished Message

2xMAC output
(Depending on the size of the negotiated hash fxn)

TLS 1.3 handshake does not rely on record layer to deliver a secure handshake

Finished messages provide a MAC over the entire handshake

… but then everything is AEAD-encrypted

May be possible to omit Finished MAC (if you rely on the record layer!)

Proposal: This needs analysis. Get some.

Shorter (Omitted?) Random Values

Max.
2x32 bytes

Random values are a nonce

If we do DH, then the KeyShare values are random

… can *theoretically* remove[1] the nonce (SIGMA proofs don't rely on it)

If we don't do DH, you need the Random

Proposal: This needs analysis. Get some.

1. Probably actually nonce = Hash(KeyShare)

Encryption for CH/SH (Ben Schwartz)

We could encrypt CH/SH (a la QUIC)

Potentially prevent: ossification, NAT slipstreaming

Needs some thoughts about exactly how

- Fixed key?
- Per-specialization key?
- ECH-based key?

Interest in this?

Next Steps

Remove Varint (if agreed)

Spin -04

Implementation and interop testing

Analysis of open cryptographic issues and add new keywords as they are validated