TurboTLS IETF 118

Faster handshaking over UDP
The problem

We want to cut TLS latency (specifically, connection establishment)
QUIC provides a nice solution for this. Migrate to QUIC :)
…unless you can’t! Upgrading legacy software can be difficult
Can we construct a minimally invasive technique to make fast TLS connection establishment available to all?
The idea

Conduct TLS handshake via UDP while TCP connection is being set up in parallel

Switch back to TCP once connection is up

This eliminates one round trip from a TLS handshake in the best case scenario

If UDP fails, fallback to TLS-over-TCP after a small timeout

Does not affect TLS security model (transport only)
TurboTLS vs QUIC (it’s not a competition!)

QUIC provides many benefits that TurboTLS does not intend to replicate.

TurboTLS only modifies transport for handshaking, with techniques for fallback.

Some firewalls struggle to perform Deep Packet Inspection due to encrypted transport headers. Thus QUIC is sometimes blocked.

QUIC supports only TLS1.3. TurboTLS works for any version.

TurboTLS can be implemented as a transparent proxy, thus in legacy applications no code needs updating.
The results so far…

Tested in GCP→GCP environments

- long distance (intercontinental) 50% improvement
- short distance (same postcode, city) no improvement
- National-distance: significant improvement
- Compared with QUIC, near identical performance (expected)
Implementation + remaining questions

An experimental implementation is included in Sandwich.

An transparent proxy will be released in the near future.

We propose a technique called client request-based fragmentation to mitigate against middlebox filtering. We would like your opinions.

Merging UDP and TCP streams raises questions (e.g. TCP fallback).
Draft

Internet-Draft: https://datatracker.ietf.org/doc/draft-joseph-tls-turbotls/

Github Repo: https://github.com/PhDJsandboxaq/draft-joseph-tls-turbotls
Questions?

David Joseph       dj@sandboxaq.com
Deirdre Connolly   deirdre.connolly@sandboxquantum.com

https://datatrapper.ietf.org/doc/draft-joseph-tls-turbotls/
Batch signatures IETF 118

Higher online TLS throughput via Merkle Trees
The problem

Load balancers, reverse proxies, etc have to deal with many incoming TLS requests per second

Post-quantum signatures have much higher latency than ECC/RSA. This has the potential to cause performance issues when signing threads get clogged

Can we make post-quantum signatures more performant?
The idea

Construct Merkle tree from message transcripts

Sign the root with a base signature algorithm

Signature is the Merkle tree sibling path (to reconstruct the root) plus the root signature

Sibling path varies for each signature

Same root signature for all in same batch

Idea first proposed in:

draft-davidben-tls-batch-signing-00

Figure 1: Merkle tree batch signature construction
Updates

In Batch Signatures, Revisited, the authors provide security proofs for the construction, including privacy notions.

We based security on Target Collision Resistance, instead of Collision resistance. This enables halving of the sibling path length. Technique from SPHINCS+

Development in PQC (and acknowledgement of its performance problems) makes high throughput signing increasingly appealing.

Different use case and applications to Merkle tree certs, however the TCR/CR update would be applicable there.
Draft

Internet-Draft: https://datatracker.ietf.org/doc/draft-joseph-tls-batch-signatures/

Github Repo: https://github.com/PhDJsandboxaq/draft-joseph-batch-signatures
Questions?

David Joseph  
dj@sandboxaq.com

Deirdre Connolly  
deirdre.connolly@sandboxquantum.com