Certificate Compression

draft-ietf-tls-certificate-compression

Alessandro Ghedini, Cloudflare

Victor Vasiliev, Google

Why compress certificates:

- Reduce amplification factor during QUIC handshake.
  - QUIC combines TLS handshake and connection establishment, so the first server flight can be used for amplification in reflection attacks.
  - Explicit source address verification adds 1-RTT to handshake.
- General performance improvement (less is more).

From Victor Vasiliev's slides at IETF 98:

Based on analysis of ~30k certificate chains from popular websites:

Compressing chains with Brotli yields (rough estimate):

- -30% size reduction at median
- -48% size reduction at 95th percentile
- Chains fitting into two QUIC packets: 2% -> 54%
- Chains fitting into three QUIC packets: 55% -> 97%

Current design:

- Supports both server and client certificates compression.
- For server certificates compression, client advertises algorithms it supports in CH extension:

```
ClientHello
+ compress_certificates             -------->
                                    <--------       ServerHello
                                                    ...
                                                    {CompressedCertificate}
                                                    {CertificateVerify}
                                                    ...
```

- New CompressedCertificate message

```
    struct {
        CertificateCompressionAlgorithm algorithm;

        uint24 uncompressed_length;

        opaque compressed_certificate_message<1..2^24-1>;
    } CompressedCertificate;
```

- If compression is not desired, server sends normal Certificate message.

Current design (cont.):

- For client certificates compression, server advertises algorithms it supports in CR extension:

```
ClientHello                         -------->
                                                      ServerHello
                                                      ...
                                                      {CertificateRequest}
                                                      + compress_certificates
                                    <--------         ...
{CompressedCertificate}
{CertificateVerify}
...                                 -------->
```

- If compression is not desired, client sends normal Certificate message.

TLS 1.3 and later only:

- Extensions in CertificateRequest were introduced in TLS 1.3.
- Certificate is encrypted, so meddling middleboxes can't see it.

Next steps:

- Get early code points assignment.
- Deploy experiment in real world to gather more data.