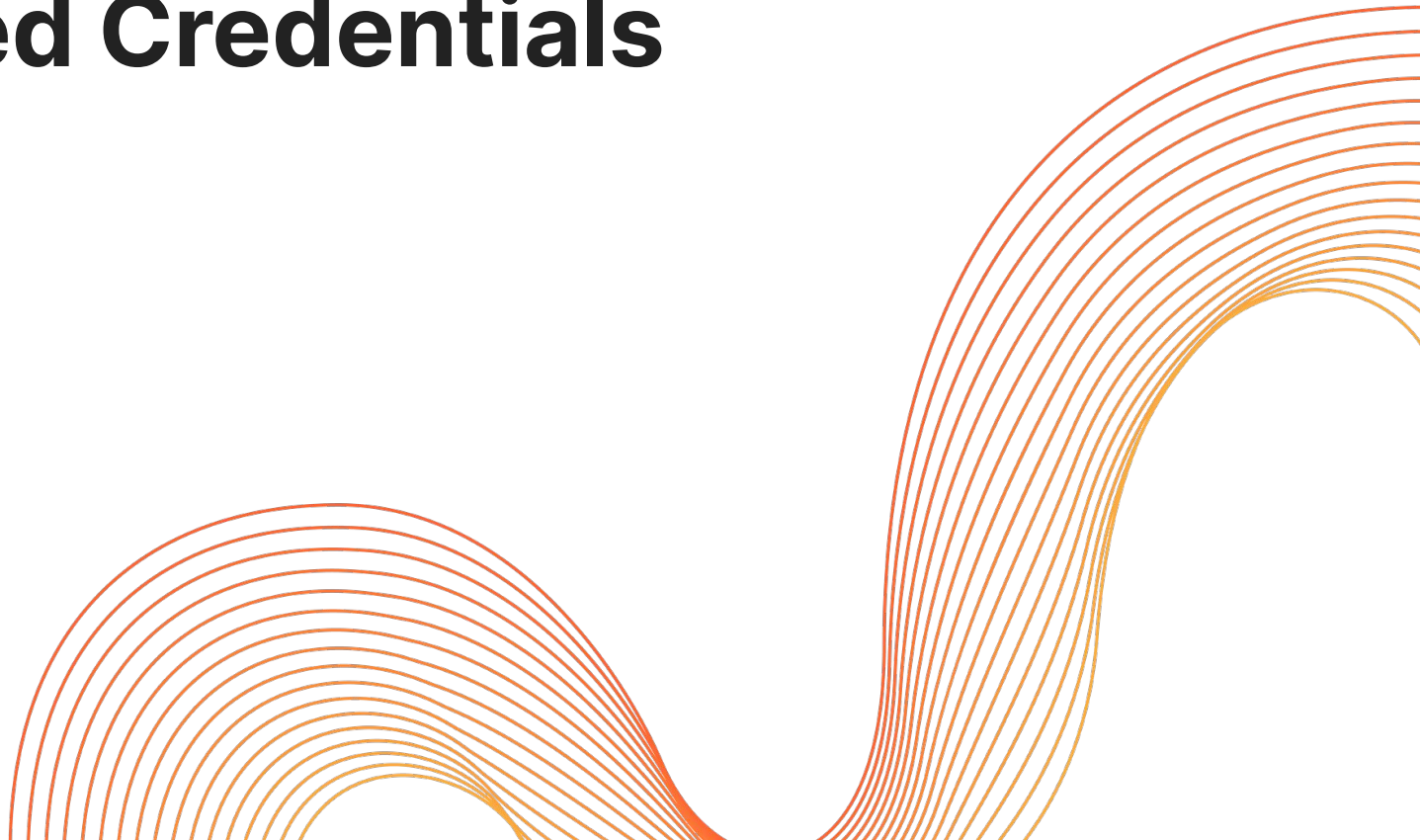
Delegated Credentials



Problems

Certificate keys live near the public Internet

Certificate revocation is broken

Deploying new signature algorithms is hard

CAs can be unreliable

CLOUDFLARE

Certificate keys live near the public Internet

- To achieve fast handshake times certificate keys are stored close to where the connection is terminated
- Sometimes those keys can leak (e.g. Heartbleed and Cloudbleed)
- We can’t make the speed of light faster
- Can we reduce the harm of a leak?

CLOUDFLARE

Certificate revocation is broken

- Certificate revocation does not work well in practice
- OCSP / OCSP Stapling is not used by all browsers
- Even when it is used, can take up to 10 days
- Browser specific revocation lists do not always include all revoked certs
- Can we make certificate lifetimes shorter?

CAs can be unreliable

- To switch to short-lifetime certs we need to be able to reliably issue certs
- Studies have found that CA uptime can be poor
- If certs have a short lifetime, and CAs are not reliably up, we could have outages

CLOUDFLARE

Deploying new signature algorithms is hard

- To experiment with new signature algorithms we need to coordinate between browsers, servers, and CAs.
- We keep signature algorithms alive for years for backwards compatibility.

CLOUDFLARE

Enter Delegated Credentials

- Instead of terminating a TLS handshake with the private key of a certificate, we can issue a delegated credential.
- The delegated credential:
  - has a public / private key pair
  - And a lifetime of less than seven days
  - Is signed by the certificate
  - And can be used in place of the certificate’s private key to sign the TLS handshake

Benefits

- Certificate long-term private keys can be stored far from the edge
- Have very short lifetimes (can be as short days or even hours)
- Are “minted” by the certificate owner, without the help of the CA
- Can use any signature algorithm the client and server support, not just those supported by the CA

CLOUDFLARE

Formal Analysis

- Extended the TLS 1.3 model
- Proves the protocol meets the same guarantees as TLS*
- Didn’t find any issues
- Hard to publish