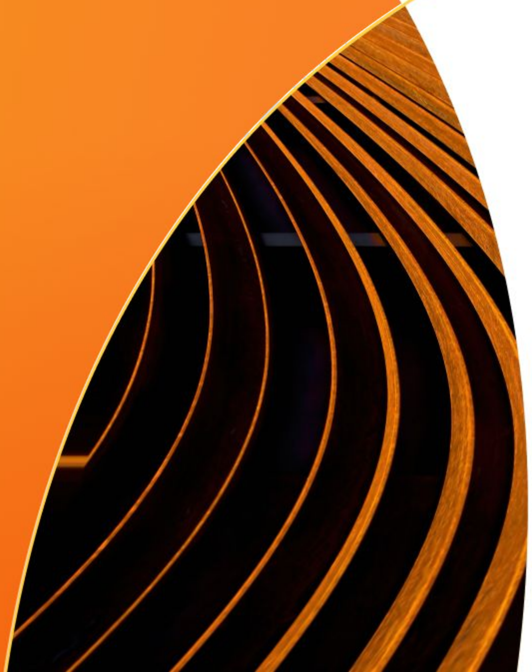
CLOUDFLARE

Requirements for Threshold TLS

Armando Faz Hernandez
Research Engineer
ask-research@cloudflare.com

MPTS 2023:
NIST Workshop on Multi-party Threshold Schemes
September 27th, 2023



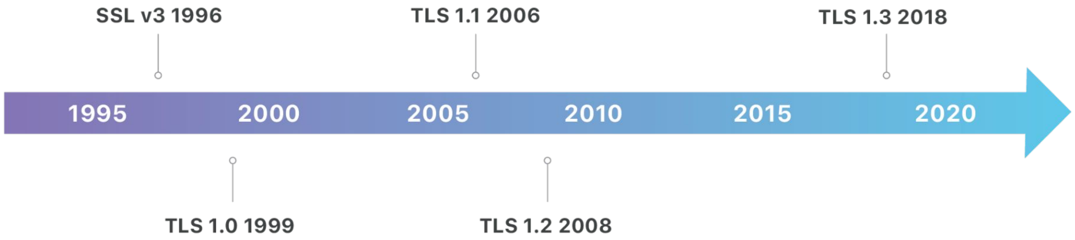
CLOUDFLARE

Agenda

- ❏ What is TLS?
- ❏ TLS termination
- ❏ Protecting TLS Keys
- ❏ Signature Algorithms
- ❏ Distributed CA

Transport Layer Security (TLS) Protocol

- Allows confidential communication between a Server and a Client.
- After protocol ends:
  - Client gets convinced that is talking to an authenticated Server.
  - Client and Server derive a shared secret used to encrypt subsequent messages.
- Client can be authenticated too (Mutual TLS).
- Relies on a Public Key Infrastructure (PKI).



TLS Protocol

Main cryptographic components:

Confidentiality

Hides the data being transferred from third parties.

Authentication

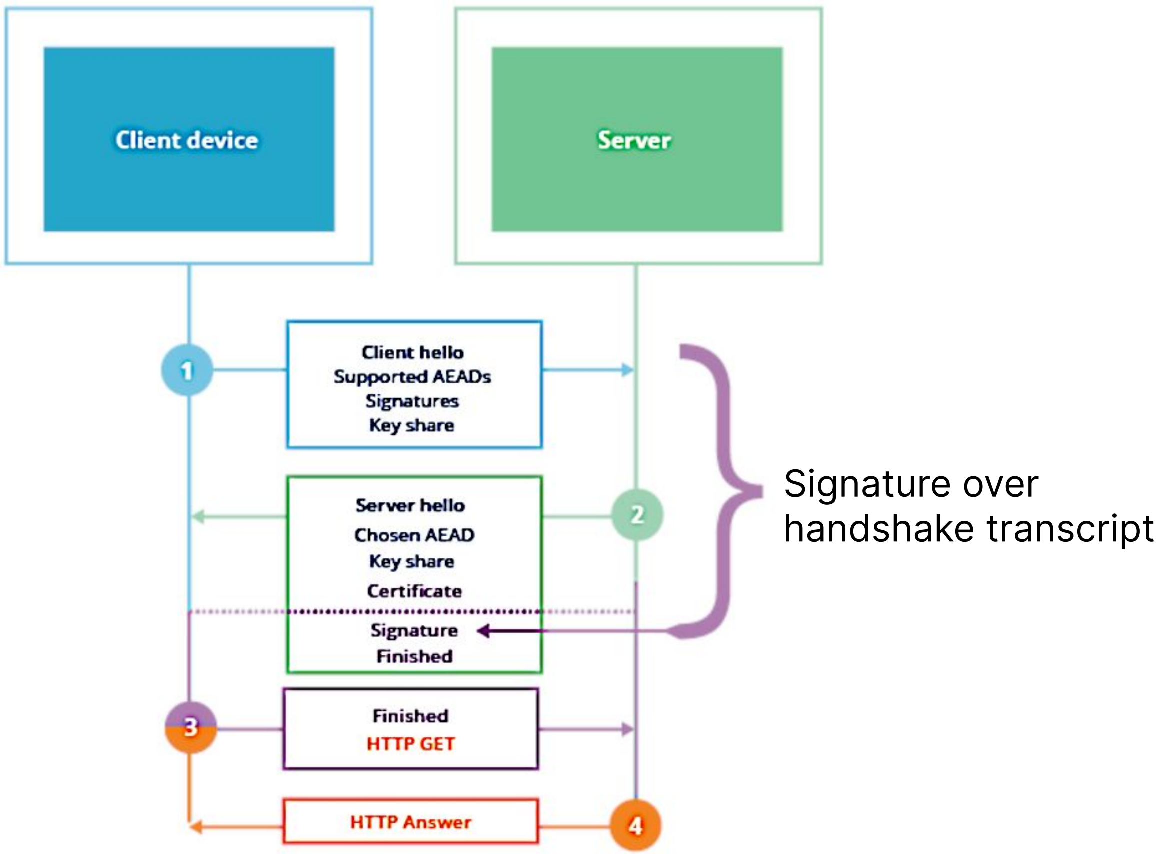
Ensures that the parties exchanging information are who they claim to be.

Integrity

Verifies that the data has not been forged or tampered with.

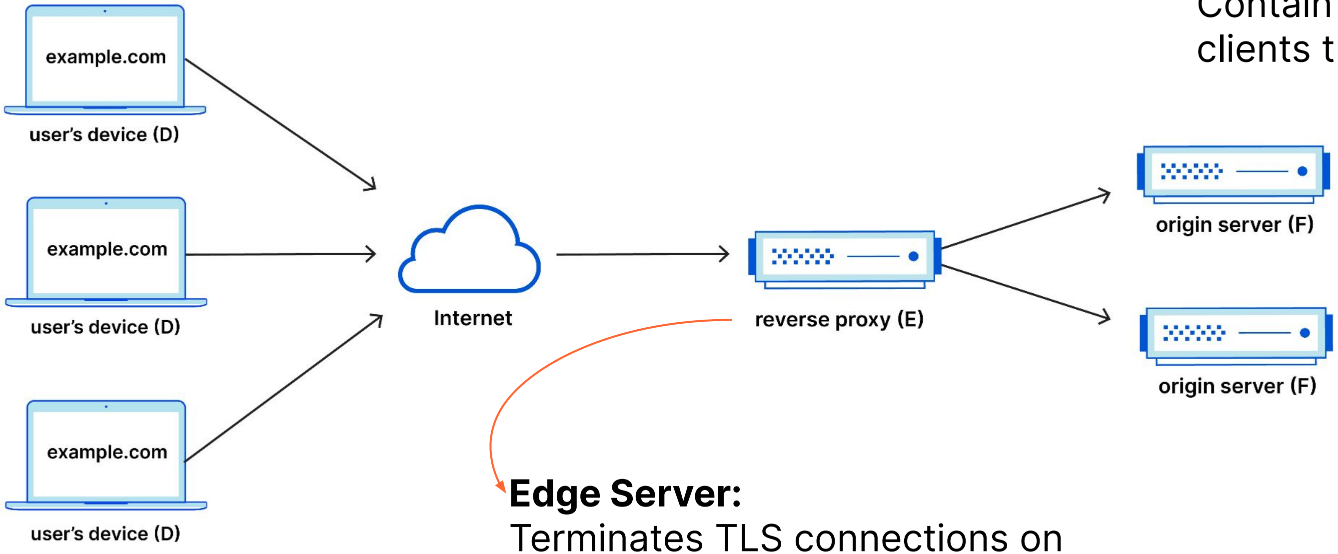
TLS v1.3

The server uses a **digital signature** to prove that the key exchange hasn't been tampered with.



TLS Termination

Origin Servers:
Contain assets to be served to clients through the edge server.

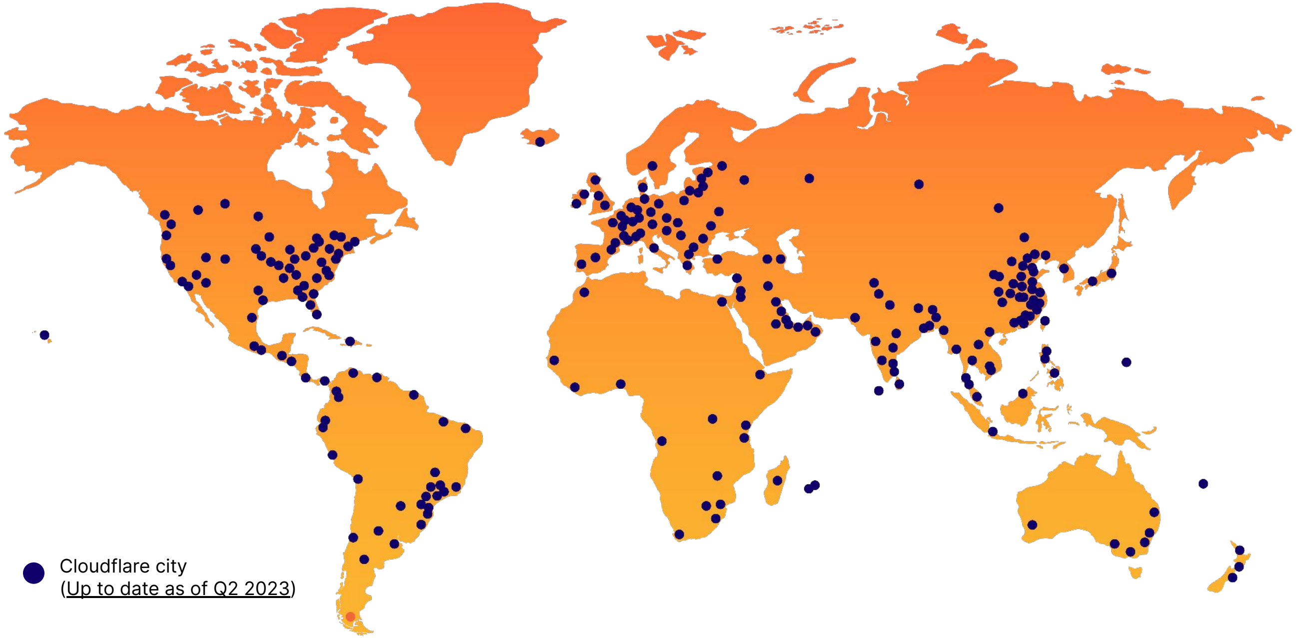


Edge Server:
Terminates TLS connections on behalf of origin servers.

That is, it produces digital signatures using the TLS private key.

CLOUDFLARE

Edge Servers



Cloudflare city
(Up to date as of Q2 2023)

46 M rps

HTTP Requests per second, on avg.

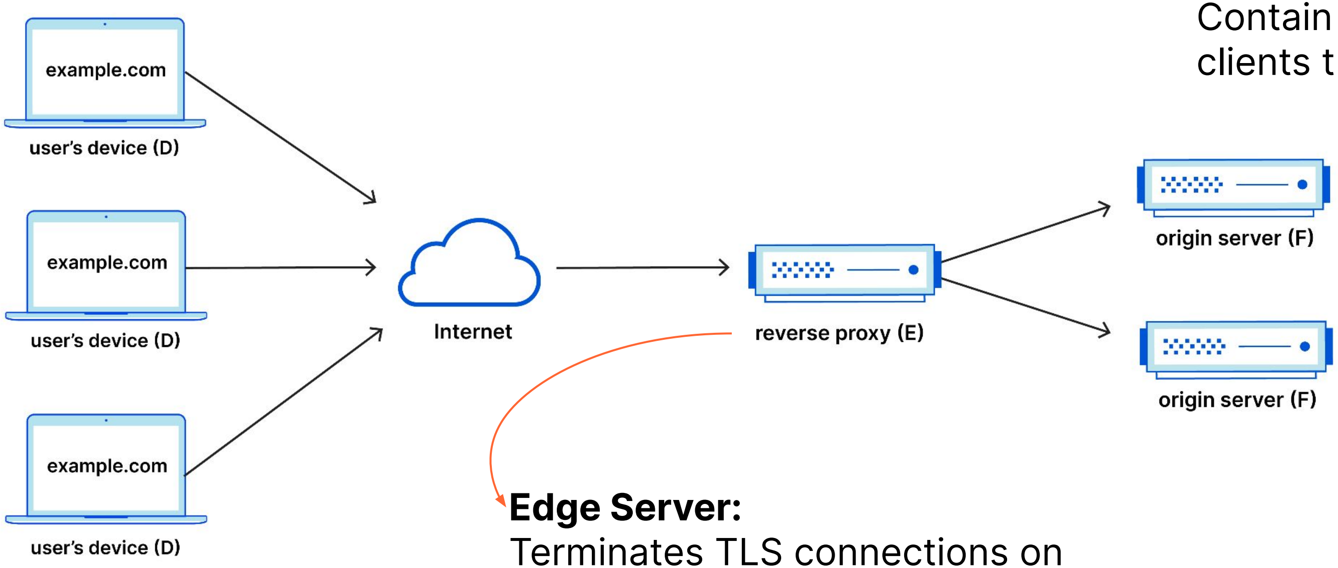
300

cities in 100+ countries, including mainland China

~50 ms

from 95% of the world's Internet-connected population

TLS Termination

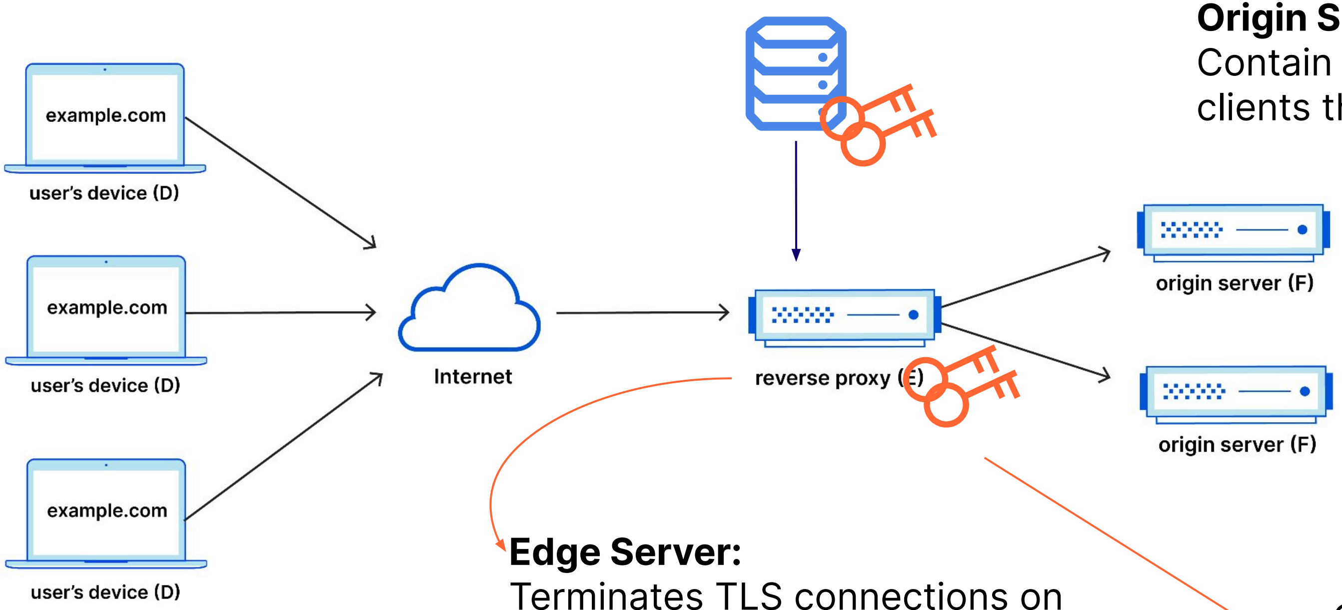


Origin Servers:
Contain assets to be served to clients through the edge server.

Edge Server:
Terminates TLS connections on behalf of origin servers.

That is, it produces digital signatures using the TLS private key.

TLS Termination - Universal Mode



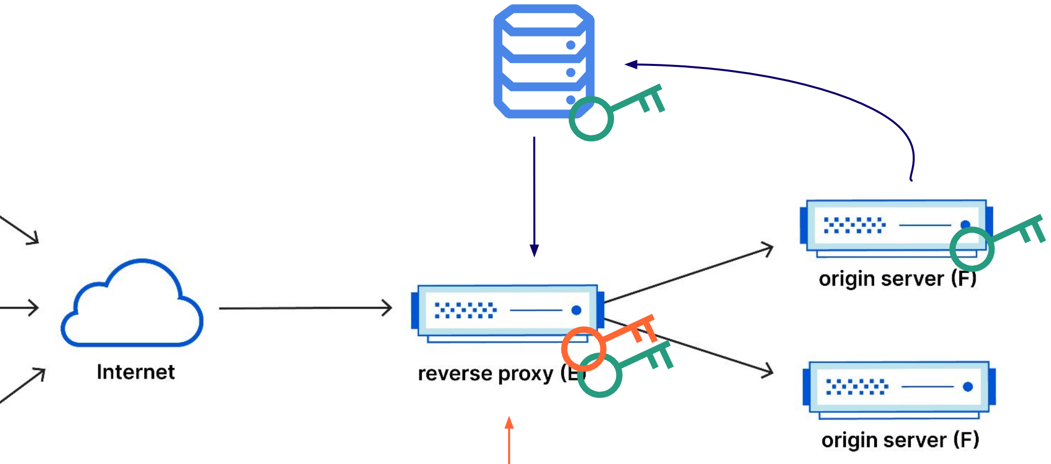
Origin Servers:
Contain assets to be served to clients through the edge server.

Edge Server:
Terminates TLS connections on behalf of origin servers.

That is, it produces digital signatures using the TLS private key.

Origin's keys are (created) and managed.

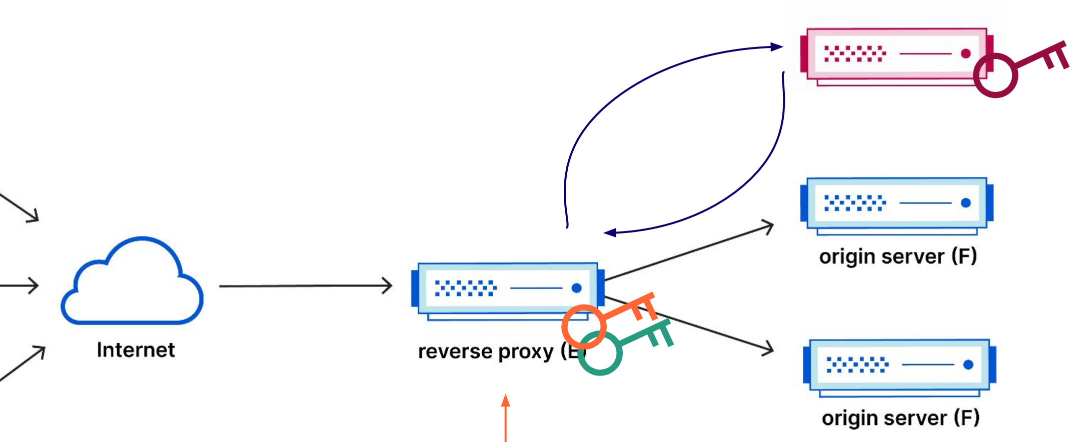
TLS Termination - Custom Certificates Mode



Origin Server:
Origins have keys created in advance and upload them to a central server.

Edge Server:
Terminates TLS connections with origin-provided keys.

TLS Termination - Keyless Mode



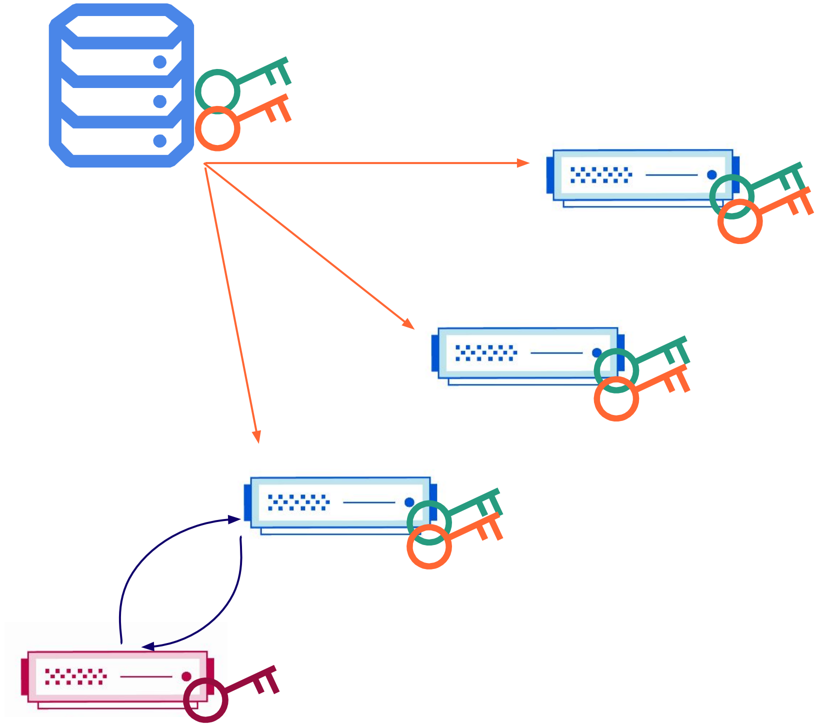
Key Server:
Its sole purpose is TLS signing.

Origin Server:
Its private key is not exposed.

Edge Server:
Terminates TLS connections with help of a key server.

CLOUDFLARE

TLS Termination



What additional measures could help protect against possibly compromised edge servers?

Keyless:
Private keys are not present in edge servers.

Threshold TLS:
Only *key shares* are located in edge servers.

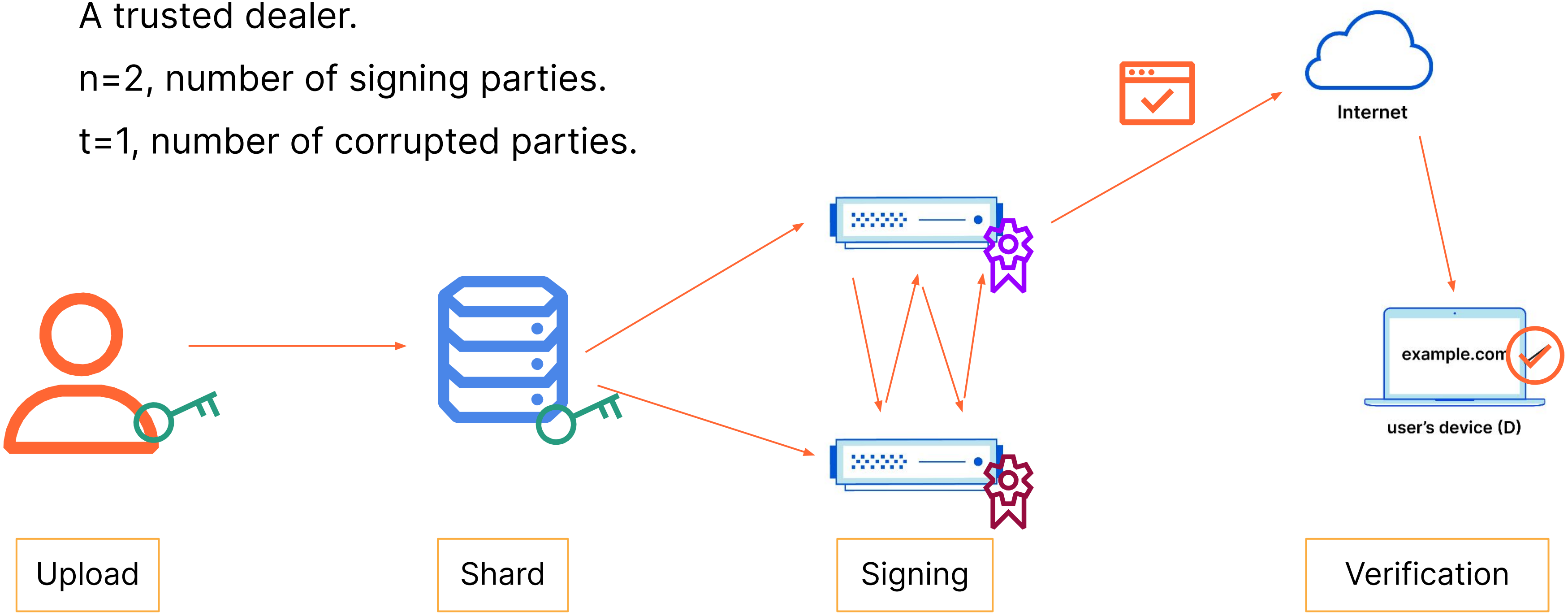
Threshold Signing

Assumptions:

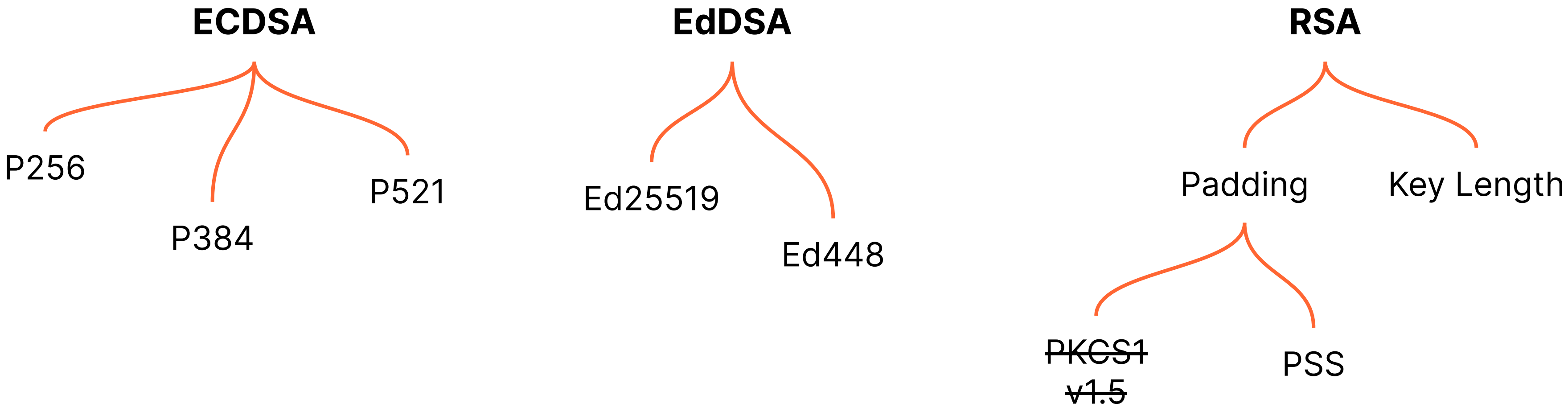
A trusted dealer.

n=2, number of signing parties.

t=1, number of corrupted parties.



Digital Signature Algorithms



CLOUDFLARE

RSA – Threshold Signing

"Practical Threshold Signatures" by Victor Shoup

KeyGen
RSA modulus must be the product of safe primes:

- $N = p*q$
- $p = 2p' + 1$
- $q = 2q' + 1$

Signing
Two full exponentiations.
If (safe primes):
appends a DLEQ proof of discrete logarithm equivalence.

Sharding
Evaluates the secret polynomial.
Generates verification keys.

Combining
Verifies the DLEQ proof.
Multi-exponentiation to combine signature shares.

https://www.iacr.org/archive/eurocrypt2000/1807/18070209-new.pdf

RSA – Threshold Signing

Pros:

- Easy to implement.
- No additional assumptions.
- Slow key generation, but ok as it happens offline.
- One round trip (two rounds).

Cons:

- Cannot apply Chinese Remainder Theorem.
    - Each party should know (p,q) – the full key
- Few values can be precomputed.
- Customer certificates do not use safe primes.
    - Most libraries do not implement them.

RSA – Threshold Signing

(t=2, n=3) Threshold RSA 2048		
	Non-safe Primes	Safe Primes
Key Generation	164 ms	66,000 ms
Sharding	10 ms	46 ms
Signing	0.84 ms	N/A
Signature Share	5 ms	10 ms
Combine Shares	0.16 ms	6 ms
Verification	0.11 ms	

Prototype Implementation

- Go language in CIRCL library.
- Safe primes generation.
- DLEQ proof.
- Still room for improvement performance-wise.

https://github.com/cloudflare/circl/tree/main/tss/rsa

Schnorr – Threshold Signing

"FROST" by Komlo-Goldberg

Pros:

- Easy to implement having a Group implementation.
- Allows precomputation of nonces and commitments.
- One round trip (two rounds).
- Works with EdDSA instances.

Cons:

- Barely use of TLS certificates with EdDSA signatures.
- Preference between Ristretto/Decaf vs EdDSA.

https://datatracker.ietf.org/doc/draft-irtf-cfrg-frost/

Schnorr – Threshold Signing

(t=3, n=5) FROST		
	ristretto255	P256
Sharding	0.061 ms	0.063 ms
Commitment	0.031 ms	0.026 ms
Signature Share	0.319 ms	0.281 ms
Combine Shares	2.286 ms	2.230 ms
Verification	0.070 ms	0.060 ms

Prototype Implementation

- Go language in CIRCL library.
- Ristretto and NIST curves supported.
- EdDSA instances under development.

https://github.com/cloudflare/circl/tree/frostyflakes/tss/frost

CLOUDFLARE

ECDSA – Threshold Signing

"Two-party Threshold ECDSA" by DKLS19

Pros:

- Fits our use case of two parties.
- No additional assumptions (ROM+ECDSA).
- One round trip (two rounds).
- Oblivious Transfer Extension - Fast using cheaper primitives.

Cons:

- Functionality is a variant of ECDSA.
- Consistency checks are not so cheap.
- Precomputation depends on the key.

https://doi.org/10.1109/SP.2018.00036

Requirements for TLS

General:

Simplicity.

Precomputation.

Key-independent & Message-Independent

Optimize time for share combination →**Fast online signing**

Optimize number of (online) round trips.

Describe explicit protocols, not only functionalities.

RSA:

Assume keys are already generated (common case).

Alternatives: Damgard-Koprowski's approach doesn't require safe primes.

ECDSA:

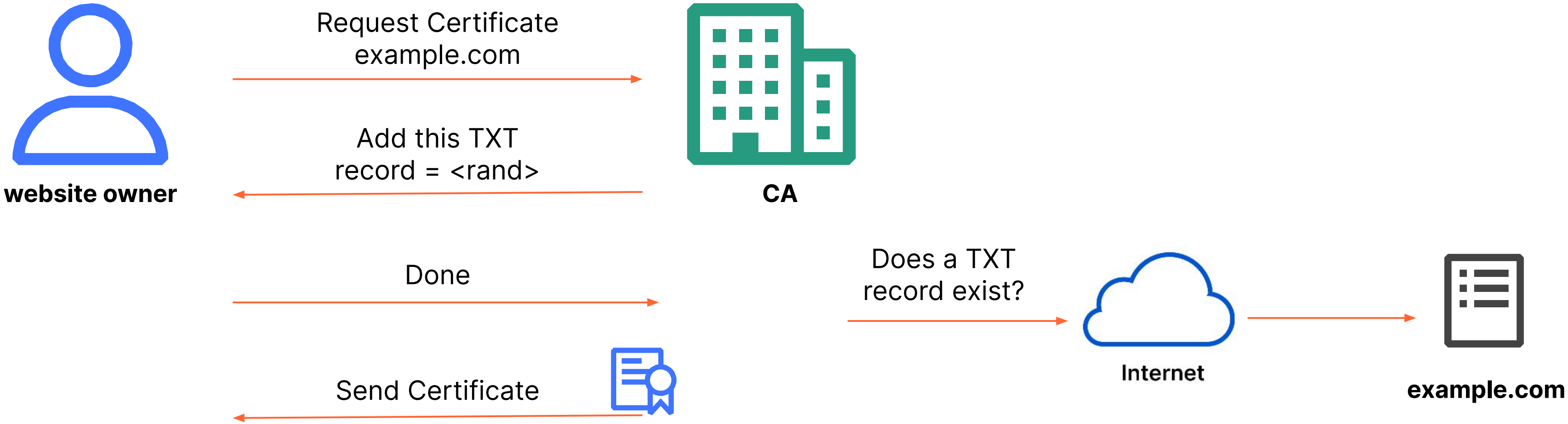
Main bottleneck is multiplication of shares.

Damgard, et al. paper requires honest majority, so (t=1, n=3).

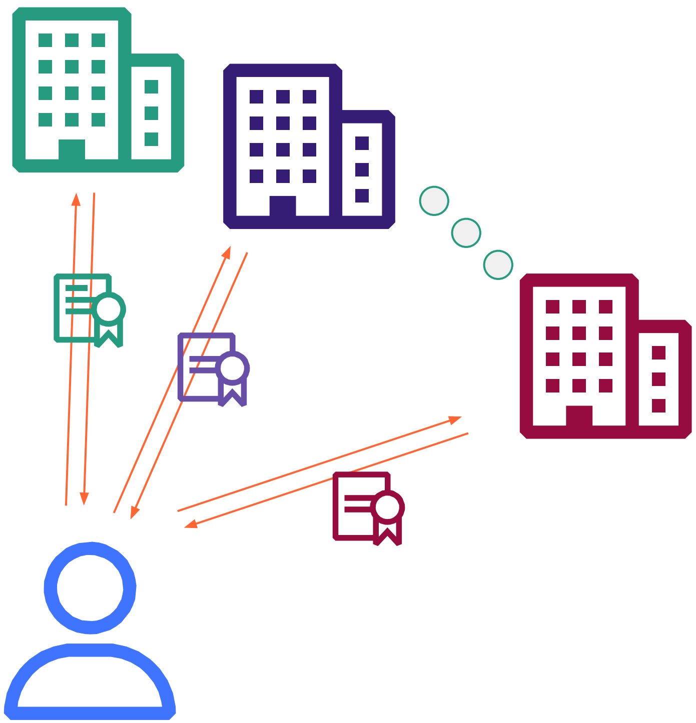
Certificate Authority (CA)

Domain Control Validation (DCV)

CA issues a certificate to a user who can prove control of a website.



Distributed Certificate Authority



- A set of *n* nodes issuing pre-certificates.
- A set of *t* of them is required to produce a certificate.
    - Nodes are operated by diverse parties: CA1, CA2,
- Each pre-certificate is signed by each CA private key.
- Shared public key for the distributed CA.
- Domain owner builds a valid certificate from pre-certificates.

Requirements for Distributed CA

- Compatibility with existing ecosystem to facilitate migration.
    - DCV, signature algorithms, etc.
- No trusted parties, thus
    - Distributed Key Generation is a must.
    - Public verifiability of pre-certificates (signature shares).
- Precomputation: minimize synchronous communication between parties.
- Identifiable aborts: detect when someone is misbehaving.
- Optimize for number of rounds.

Thanks!

Cloudflare Research

ask-research@cloudflare.com

https://research.cloudflare.com