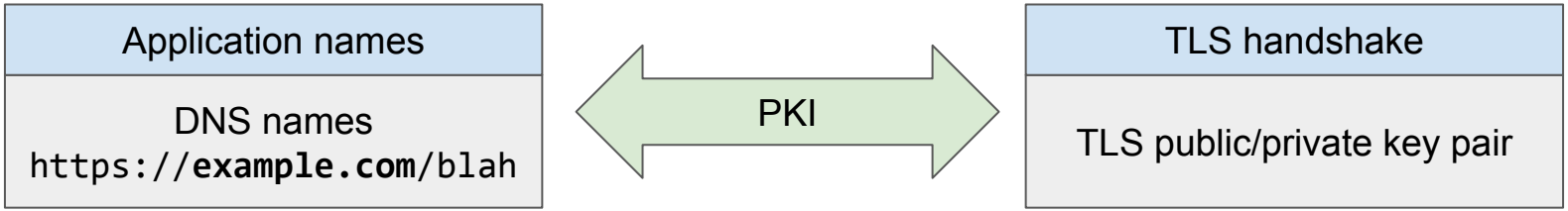
TLS Interim

Oct 1, 2024

Considerations for solving the “Trust Tussle”

- Divergence today causes widespread pain — the pain delays improvements to user security, and causes servers to break the clients they wish to support.
- This pain is increasing over time — many servers have a long tail of 10-15 year old “must support” clients, which gets larger every year.
- While a solution is crucial for the PQ transition, it is not the primary motivating factor for solving it.
- A solution will allow clients to better meet their user’s security needs, while allowing servers to choose to support a wide range of clients at their own schedule.

Background



CAs issue *certificates*, signed name-key pairs

Clients curate lists of CAs that are trusted to *only* sign correct name-key pairs

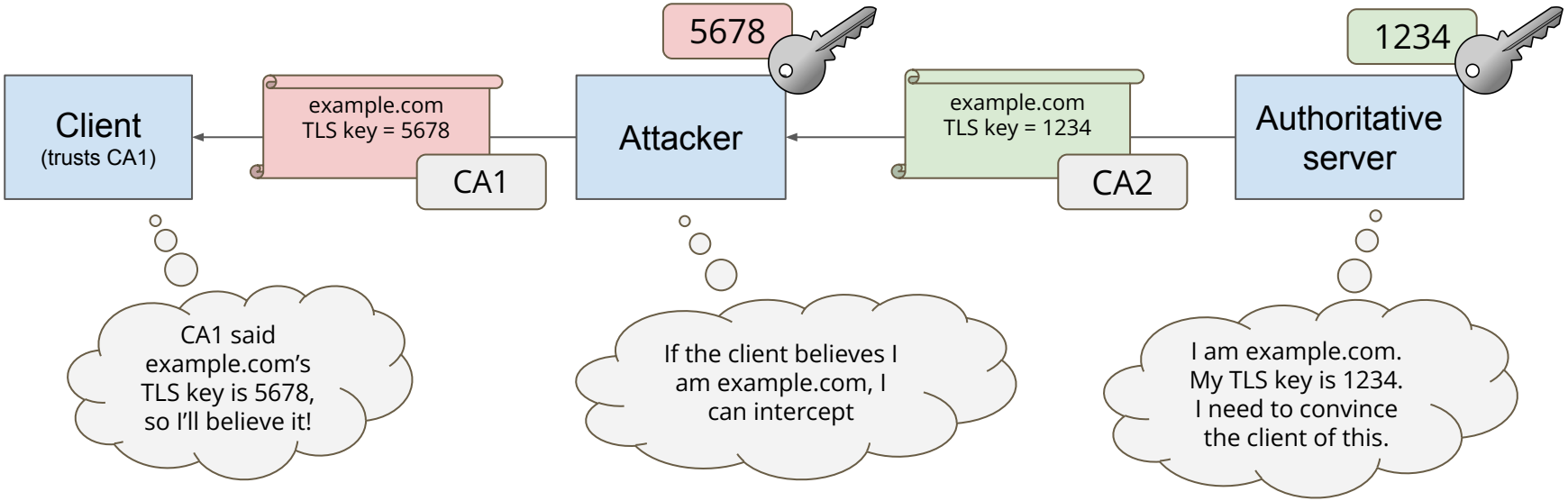
Service Availability: Client accepts correct name-key pairs
Client must trust enough CAs, so authoritative servers can obtain and present a valid certificate

User Security: Client does not accept incorrect name-key pairs
Client must *not* trust untrustworthy CAs, so users are not at risk of TLS interception from invalid name-key pairs

TLS Interception

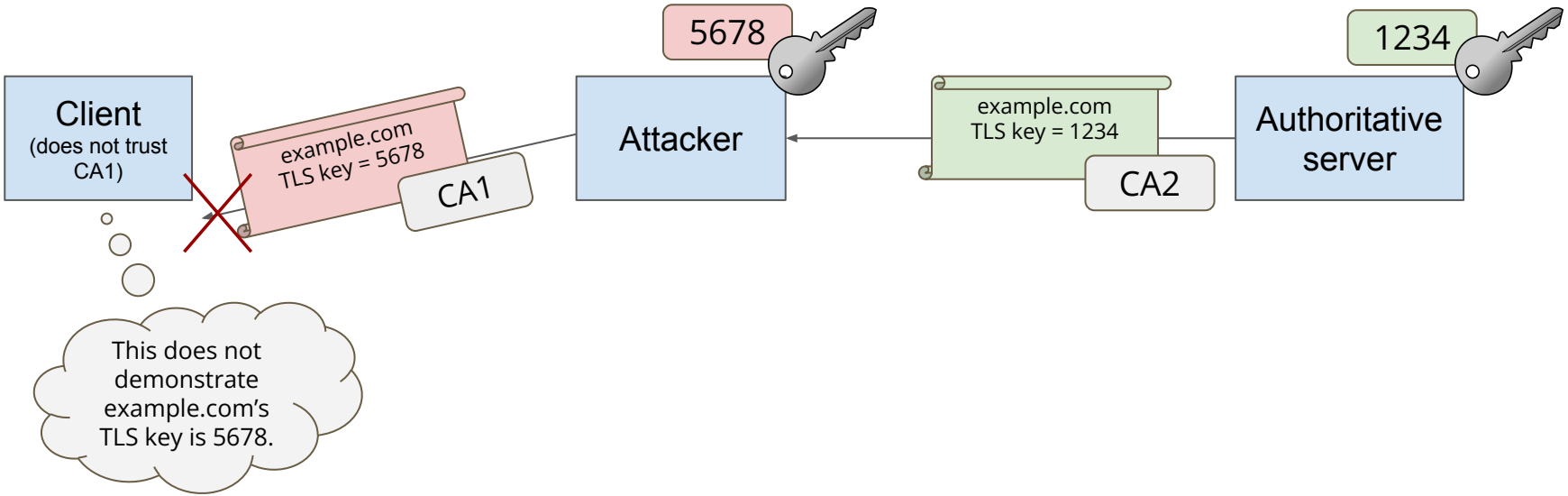
Clients trust CAs to *only* signs correct name-key pairs

If a CA signs incorrect pairs, attackers can impersonate the server



TLS Interception

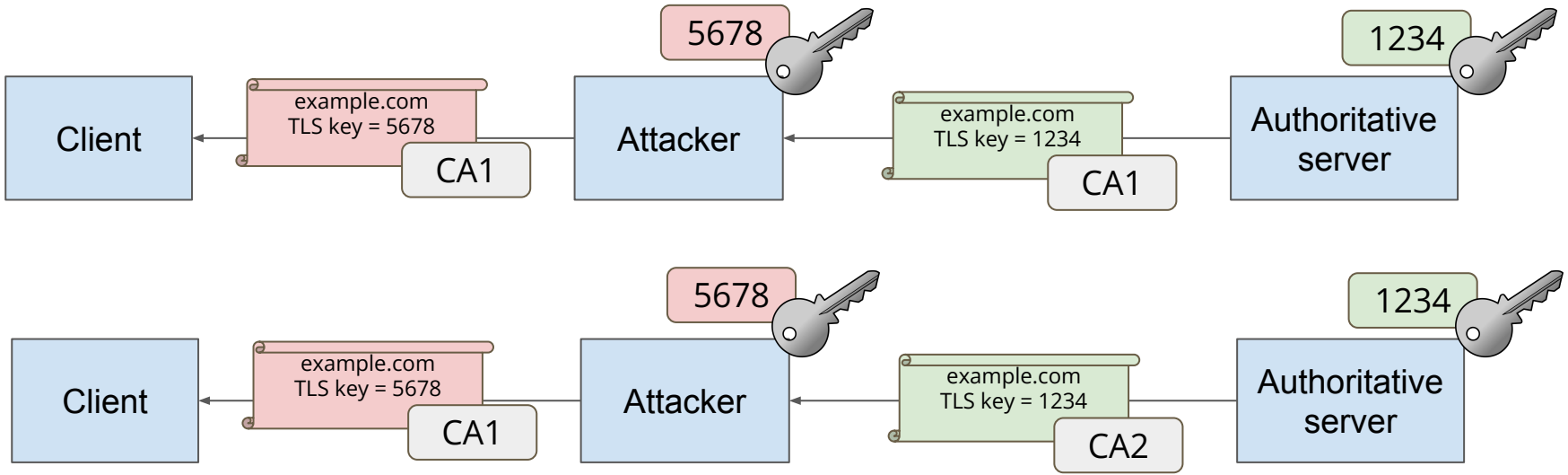
To mitigate this attack, clients *must not trust untrustworthy CAs*



TLS Interception

Which CA signed the *authoritative server's* name/key pair *does not matter*

Interception depends on the *client* accepting the *attacker's* key



The “Trust Tussle”

Over time, and across clients, PKIs face pressures toward client trust anchor diversity.

As diversity increases, if a server cannot then support all clients, the PKI must either:

- *Sacrifice user security* — clients trust CAs they do not deem trustworthy and put users at risk
- *Sacrifice service availability* — servers break unusual clients whose users they would otherwise wish to support

We wish to **avoid this conflict by enabling servers to reliably and efficiently support clients with diverse trust anchor lists**

Focused on larger PKIs — small PKIs already solved by `certificate_authorities`

Motivating Examples

*“**Changes to Google's certificates have always been painful because there's no negotiation about certificates in TLS**. All we can really do is to make a change and see what breaks. Additionally, as more and more of Google's services have moved to HTTPS, and more and more robot clients are talking to Google over HTTPS, **certificate changes are getting more painful over time**. Ten years ago we could have tested with a few browsers on a few operating systems and had pretty good confidence about a change. Five years ago, we were worrying about odd clients like feature phones, but they weren't too numerous. **Now, we have to worry about large, diverse populations of devices that we have little knowledge of**.”*

— Adam Langley, internal retrospective on Google certificate changes, 2015

Motivating Example: 2017 Symantec PKI Distrust

The domain validation of nearly all certificates from a CA called into question

- The scale and nature of the underlying incident required prompt distrust
- Coordination across relying party browsers and OSs introduced significant delays

Site operators pleaded to reverse course due to unavoidable site breakage

- Sites needed to support multiple clients with constrained trust stores
- No other CA could issue a certificate trusted by all supported clients
- Deprecating certain clients is not always possible, but is always costly
- Major sites forced to re-architect their service to accommodate client divergence

Motivating Example: Certificate Transparency Policy

Servers must serve SCTs to satisfy all supported CT-enforcing clients

- To ensure durable transparency, Chrome required 1 SCT from a Google log
- Subsequent clients mirroring this policy for themselves embrittles the ecosystem
- Clients also needed to accept same set of logs, on the same set of timelines

Subsequent CT-enforcing clients were forced into a looser policy

- Somewhere around 2016, one CA secretly acquired another CA
- Both operated “distinct” CT logs and were thus able to circumvent CT
- Chrome pressured to adopt looser policy to remove ecosystem centralization

Motivating Example: Certificate Transparency Shared Fate

Google attempted to turn down the original set of CT logs in 2019

- Their codebase was deprecated and had scaling and operational issues
- Cheaper, more reliable replacement logs had already been operated for years

Certain Apple OS clients constrained to relying on these specific logs

- Clients were still widely supported by servers but could not be updated to add new logs
- Servers could not differentiate clients, so logs remained online despite the risks and costs

Appmattus CT Library breakage (a.k.a. McDonalds-gate)

- Unknown to CT maintainers, a widely-deployed mobile networking library enforced CT
- Hardcoded dependency on Chrome’s CT log list; broke apps on schema change

Motivating Example: Root Programs Diverging

~2022: A major root program has stopped accepting new CA operators entirely

- New, modern CAs cannot independently issue trusted certificates
- Ossification reduces pressure for existing CAs to improve and modernize

2022: Launch of Chrome Root Program and certificate verifier

- Root programs launch because existing platform trust does not meet user needs
- Inherited CAs from existing root programs out of compatibility pressure
- Precipitated distrust of CAs that would never have met the bar for inclusion

~2023: CA incident response expectations between root programs beginning to diverge

- Parent root programs have changed or relaxed enforcement of their policies
- Notable examples of CA incident response quality visibly decreasing

Motivating Example: Post-quantum

Many new post-quantum CAs will have to spin up and then be accepted by clients

- Clients may include different hierarchies at different times, or not accept them at all
- At any time, clients will be in a variety of different points in this transition
- Servers will need to accommodate the resulting significant diversity of clients
- Ubiquitous set for RSA formed when both the web and the web PKI were much smaller

Clients may opt for different certificate trade-offs due to significant size costs of PQC

- This introduces a dimensionality that cannot be supported by e.g. signature_algs*
- Even if they eventually agree, the ubiquitous set will take time to converge

Benefits of Solving the “Trust Tussle”

Constrained client trust stores would not impede security decisions for agile clients

Clients could freely implement CT policies that best serve their users

One client’s CT (mis)behavior would not dictate ecosystem-wide decisions

New root programs would be free to only accept CAs that best serve their users

Servers could automatically support an increasingly diverse client base without relying on heuristics and fingerprinting

Convergence Incentives

Excessive divergence remains a burden for servers — need to consider many clients, obtain many certificates, etc.

Clients remain incentivized to converge to minimize this burden

Solving the problem means this incentive can be balanced with other needs:

- User security — remove untrustworthy CAs
- Service availability — add and make use of new CAs that provide value to the ecosystem

Consequences of the Status Quo

Ecosystem convergence is often positive, but prioritizing convergence above all else is dangerous.

Convergence can mean:

- Clients trusting untrustworthy CAs to avoid burdening server operators
- Servers being forced to deprecate clients, despite wanting to serve the users using those clients
- Ossifying the PKI to the status quo and disincentivizing improvements and new/modern entrants

Recap

The status quo today means PKIs must either:

- *Sacrifice user security* — clients trust some CA they otherwise would not deem trustworthy
- *Sacrifice service availability* — servers break unusual clients whose users they would otherwise wish to support

We wish to **avoid this conflict by enabling servers to reliably and efficiently support clients with diverse trust anchor lists**

Motivating Example: Optimizing up-to-date clients

Omit lowest-common-denominator intermediates

- Servers serve cross-signs and intermediates to keep older clients working
- CAs issue short-lived intermediates off long-lived roots
- Up-to-date clients could have trusted intermediate as a short-lived root

Merkle-Tree Certificates

- More efficient certificate structure that *only works if client is newer than certificate*
- Design required a negotiation mechanism, which is where TE and TAI came from

Servers must support older clients as long as they wish to serve those users

Every older client was once a newer client

Motivating Example: Certificate Pinning

Some clients want to protect connections to high-value servers, at the expense of generality. Examples include:

- An update server for some client
- A mobile app that primarily talks to one server

Today, this must work with a server's existing certificate, so must pin to existing CA

Implicitly assumes that server's choice of CA *never changes*

This causes outages as PKIs and servers evolve, outages increasing over time[1]

No way to use client-specific private PKI, which could evolve independently

[1] https://blog.cloudflare.com/why-certificate-pinning-is-outdated/