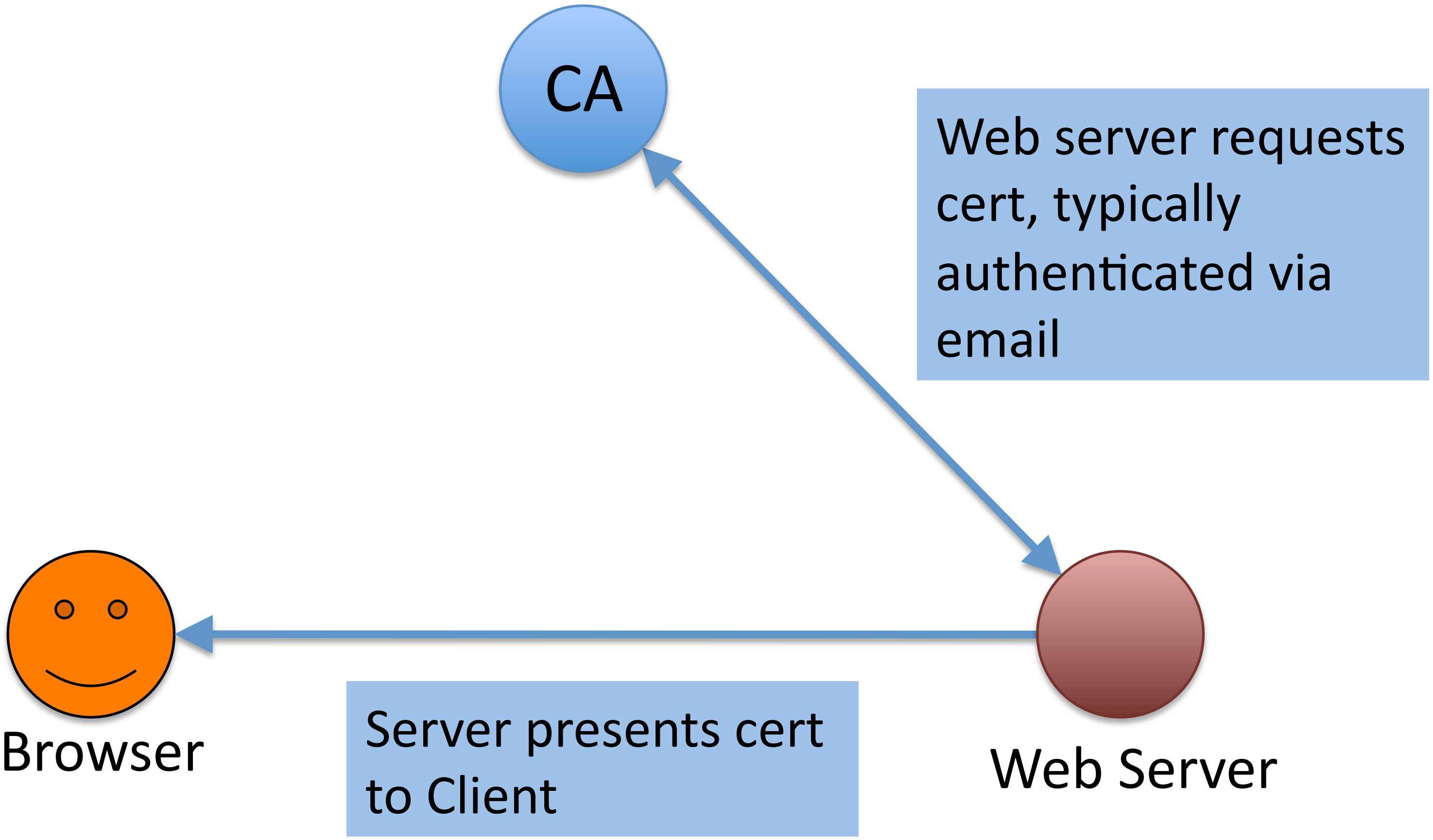
Transparency, Trust Agility, Pinning

(Recent Developments in Server Authentication)

Trevor Perrin <trevp@trevp.net>

Certificate Authorities



Web server requests cert, typically authenticated via email

Server presents cert to Client

Web PKI

- 50+ Root CAs, unknown number of Sub CAs
- Most CAs can issue certs for any domain
- Known CA failures in last 2 years:
  - Comodo - hacker issued bad certs
  - Diginotar - hacker issued bad certs for MITM
  - Trustwave - issued sub CA to customer for MITM
  - Turktrust - issued sub CA by mistake, used for MITM

Can we recover from bad certs?

Revocation

- Online lookups (CRLs, OCSP)
  - Slow
  - Leaks browsing history
  - Connection could fail (security/reliability tradeoff)
- Fresh signatures from CA (e.g. OCSP stapling)
- Out-of-band update (software update, crlsets)
  - (Chrome current crlset = ~24000 entries, ~250 KB)

Change who we trust?

DNSSEC/DANE

- DNSSEC adds key and signature records to DNS

- DANE adds records for application keys

- Considered as a PKI:
  - Fewer trusted parties (ICANN root, TLD registry, registrar, and your own DNSSEC keys)
  - Builds on existing authentication relationships

DNSSEC/DANE challenges

- "Last mile" problem: getting DNSSEC to clients
  - Fetching DNSSEC records over DNS has reliability and latency problems
  - Stapling needs universal deployment before a "fail-if-absent" client policy
- DNSSEC is not widely deployed on domains
  - More complex than cert requests

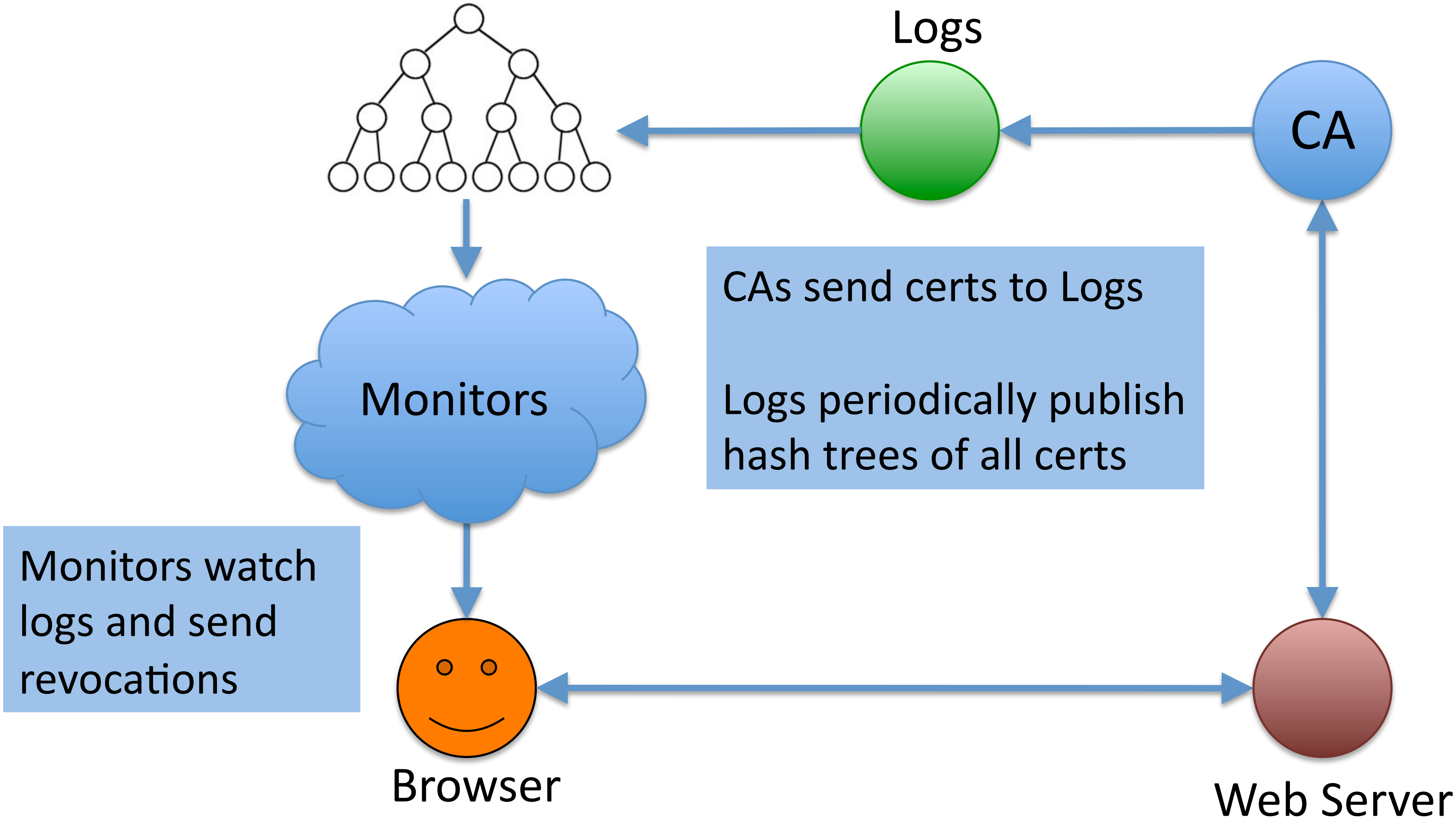
Change *how much* we have to trust anyone?

Certificate Transparency

- Goals
  - CAs publish all certificates
- Challenges
  - What if they don't?

    (mistakes, hacks, intentional, etc.)
- Laurie and Langley et al, Google, started 2011
  - IETF draft in progress

Logs and Monitors

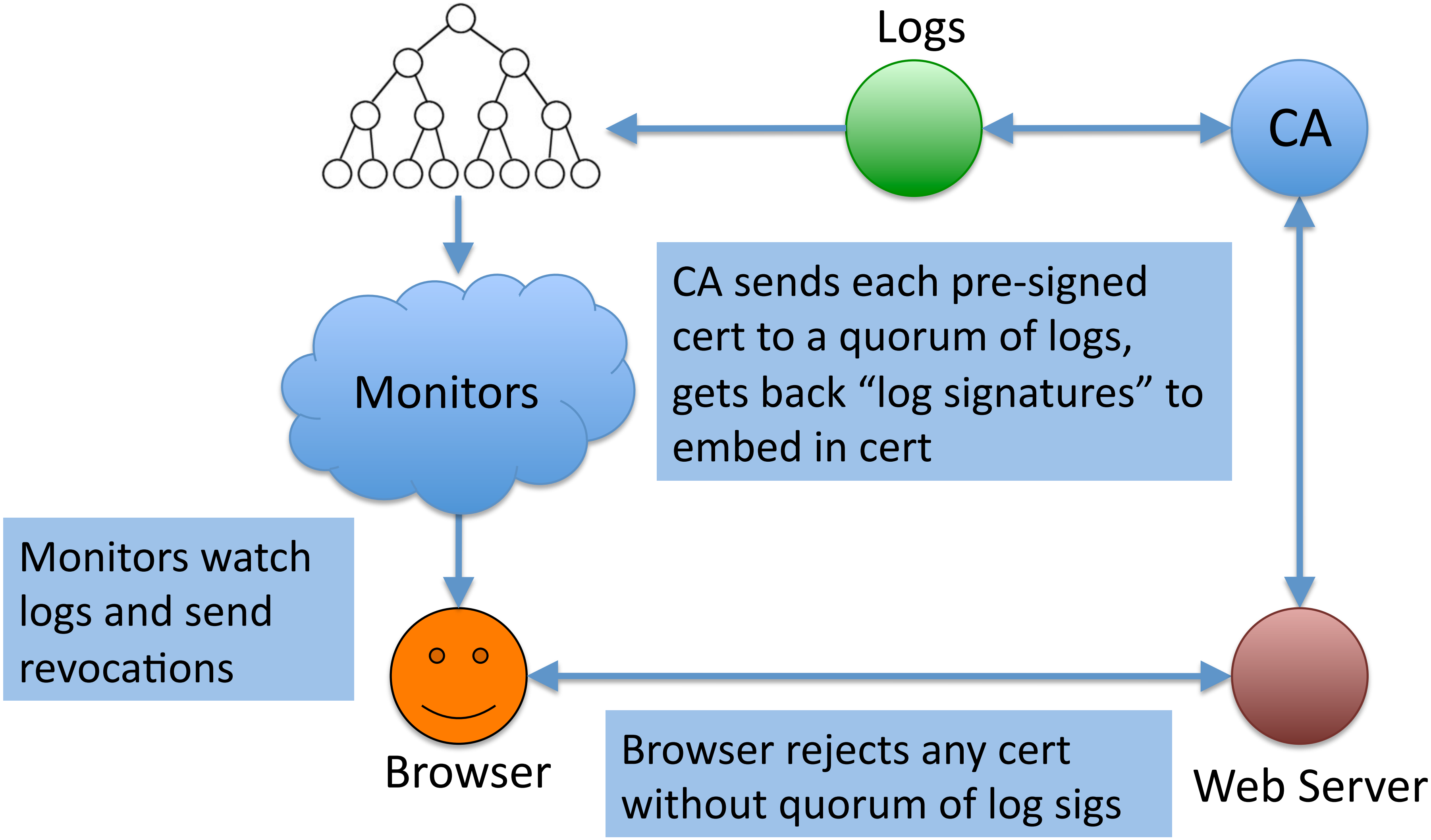


CAs send certs to Logs

Logs periodically publish hash trees of all certs

Monitors watch logs and send revocations

CT Part 1 – Log Signing

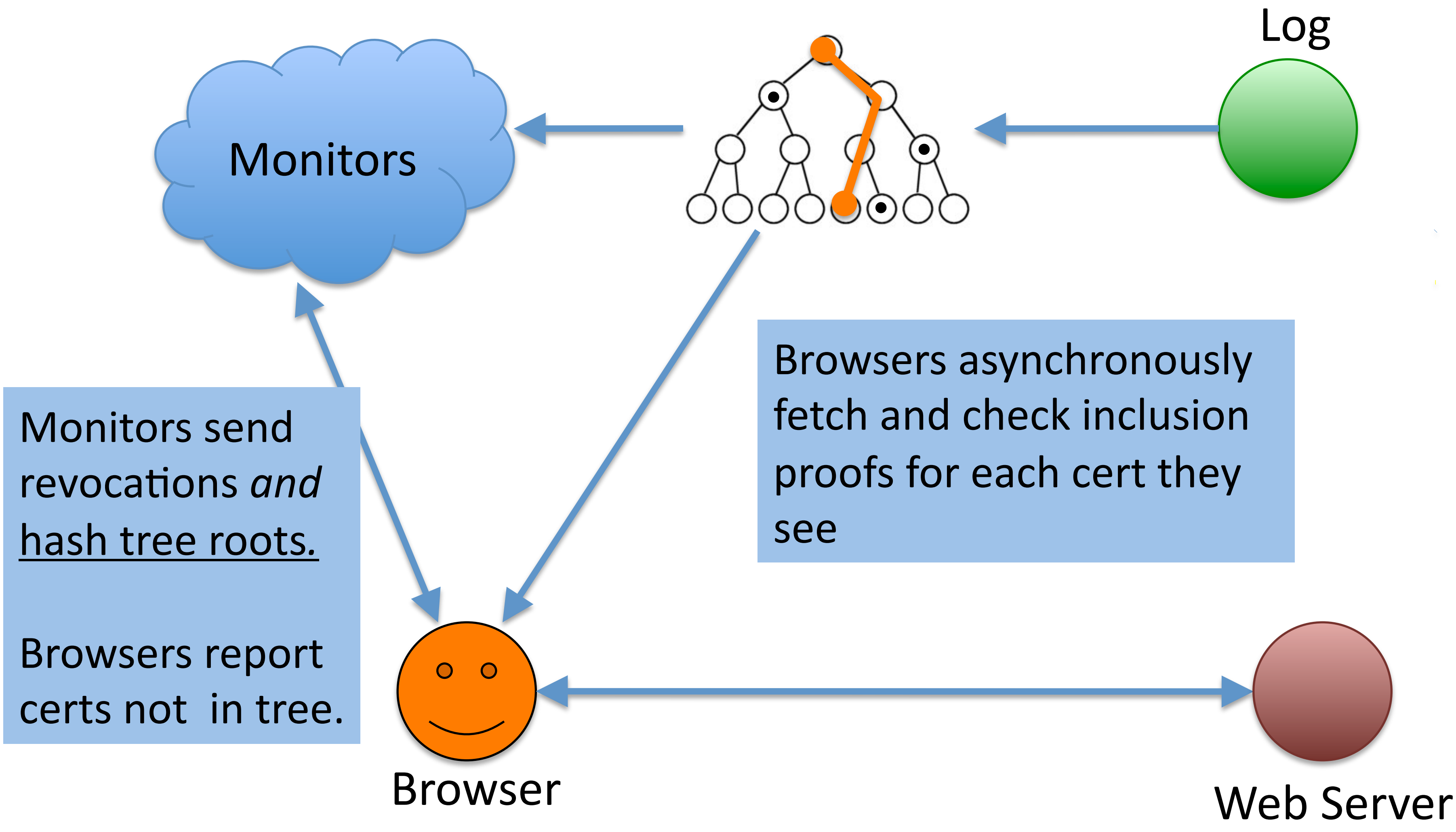


CA sends each pre-signed cert to a quorum of logs, gets back "log signatures" to embed in cert

Monitors watch logs and send revocations

Browser rejects any cert without quorum of log sigs

CT Part 2 – Online Log Checking



Browsers asynchronously fetch and check inclusion proofs for each cert they see

Monitors send revocations *and* hash tree roots.

Browsers report certs not in tree.

Cert Transparency Challenges

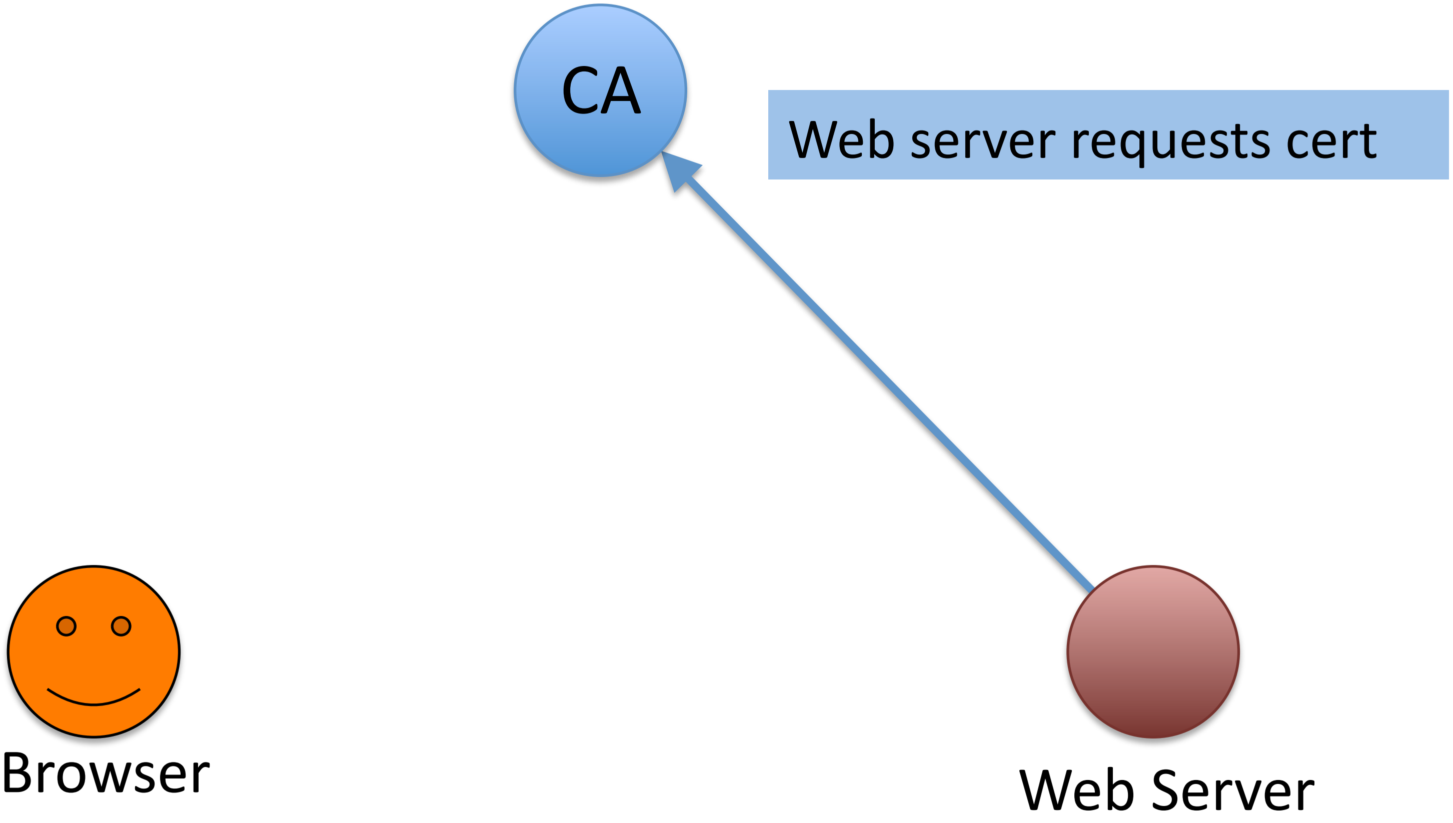
- Requires multiple high-availability logs

- Log signatures need universal deployment before a "fail-if-absent" policy
  - But can be done by CAs

- Requires good monitoring and revocation, and an infrequently-breached CA system

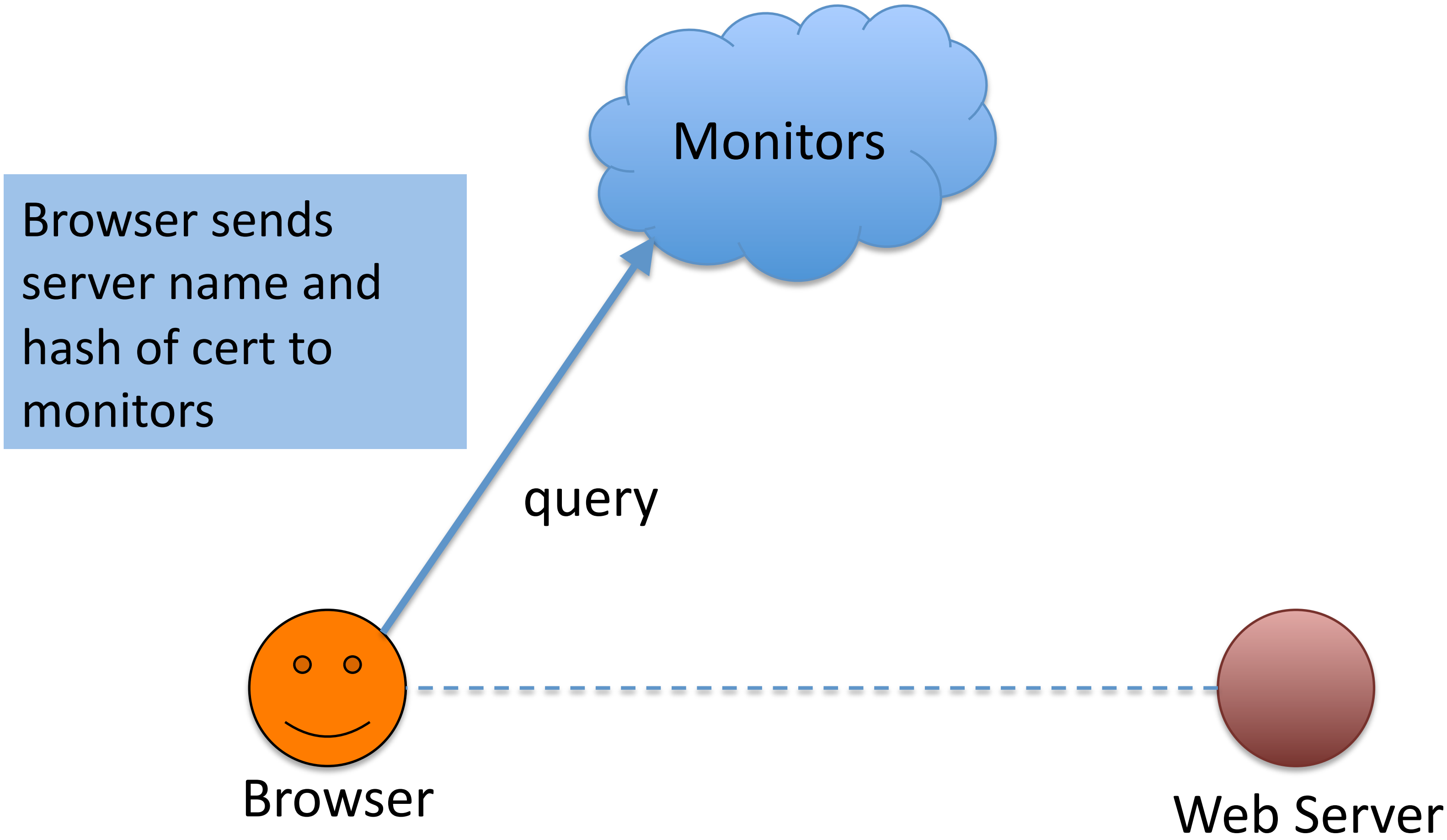
Don’t use CAs?

CAs again

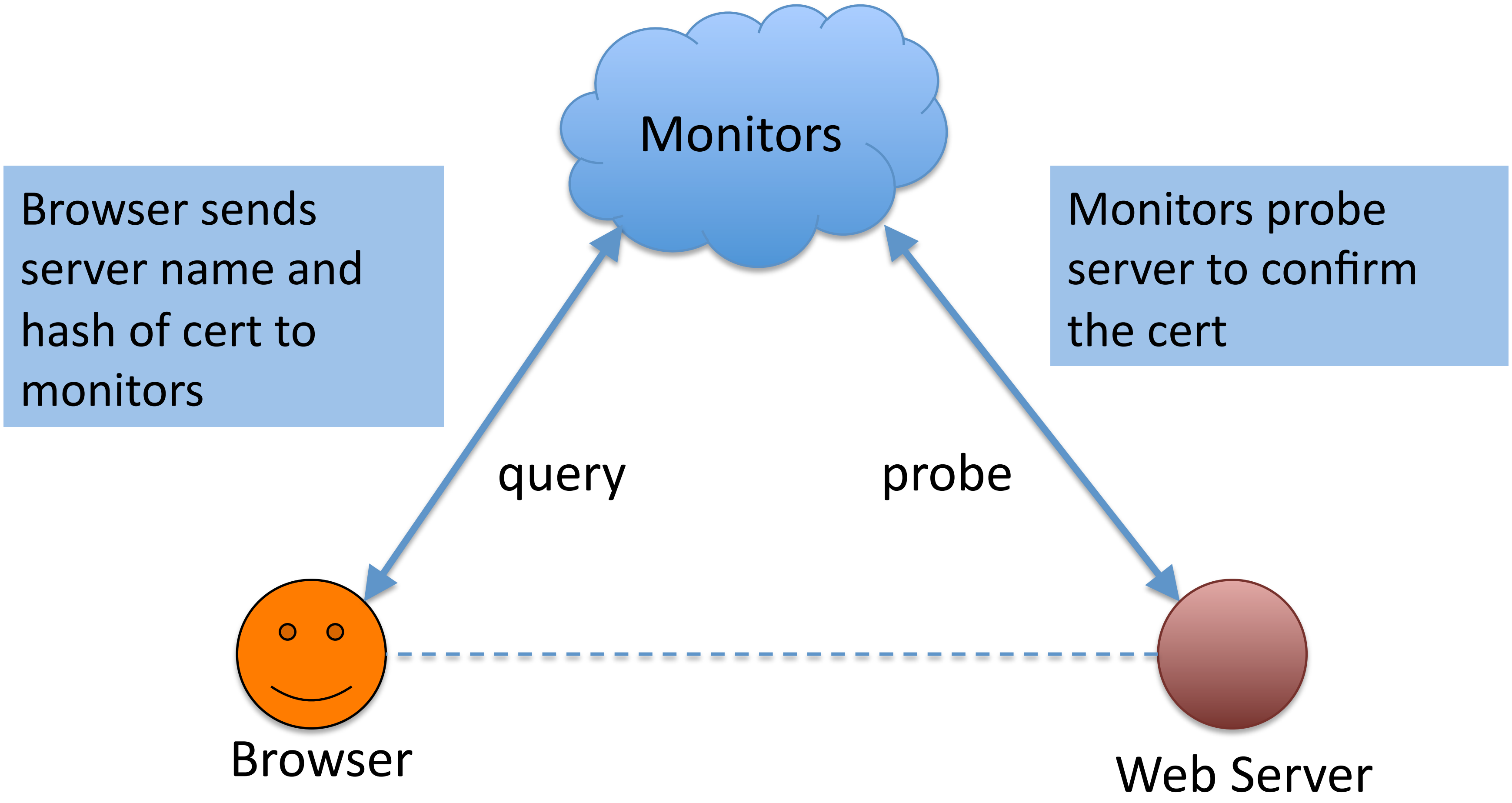


Web server requests cert

Convergence


Browser sends server name and hash of cert to monitors

Convergence



Browser sends server name and hash of cert to monitors

Monitors

Monitors probe server to confirm the cert

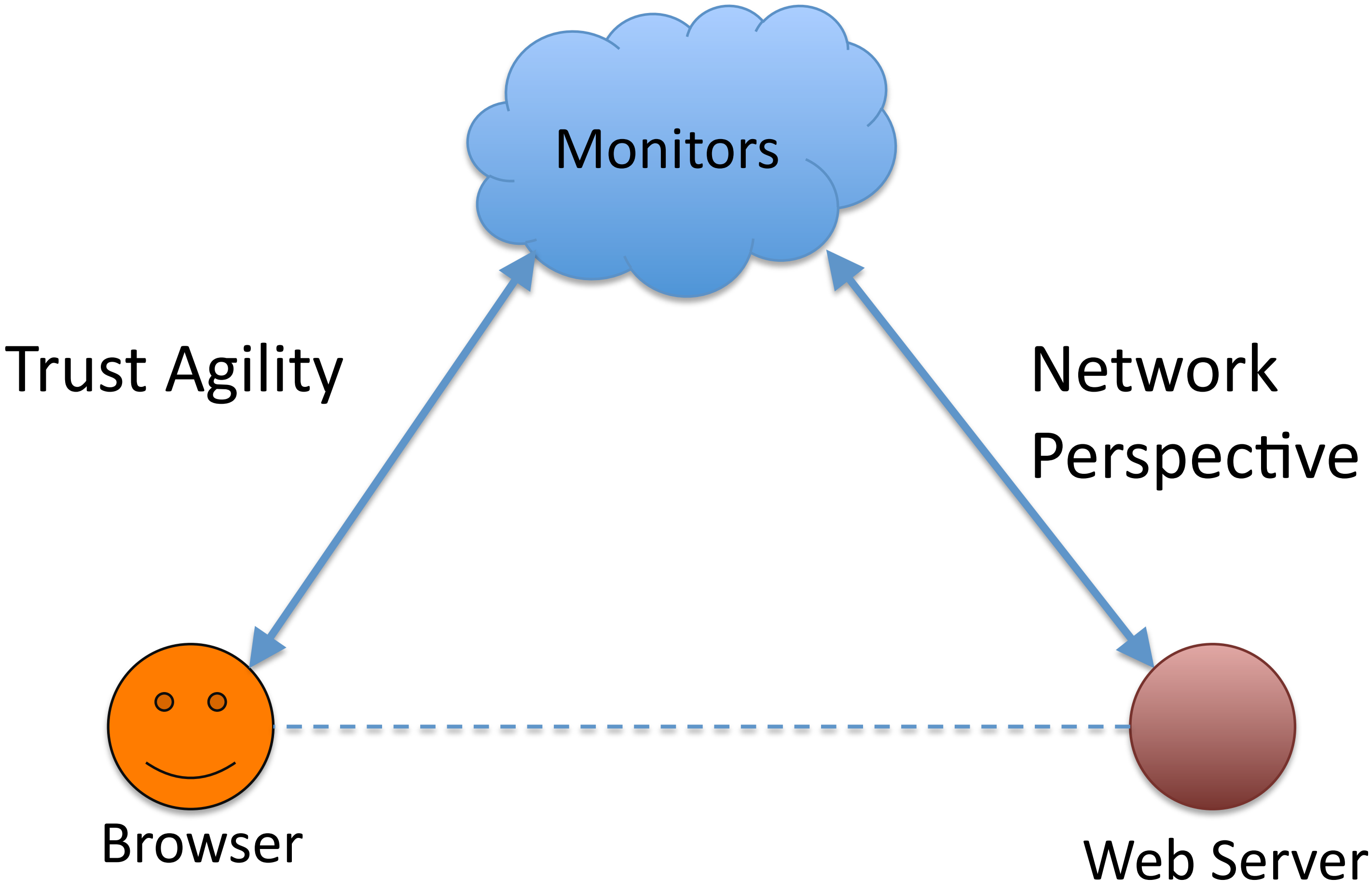
query

probe

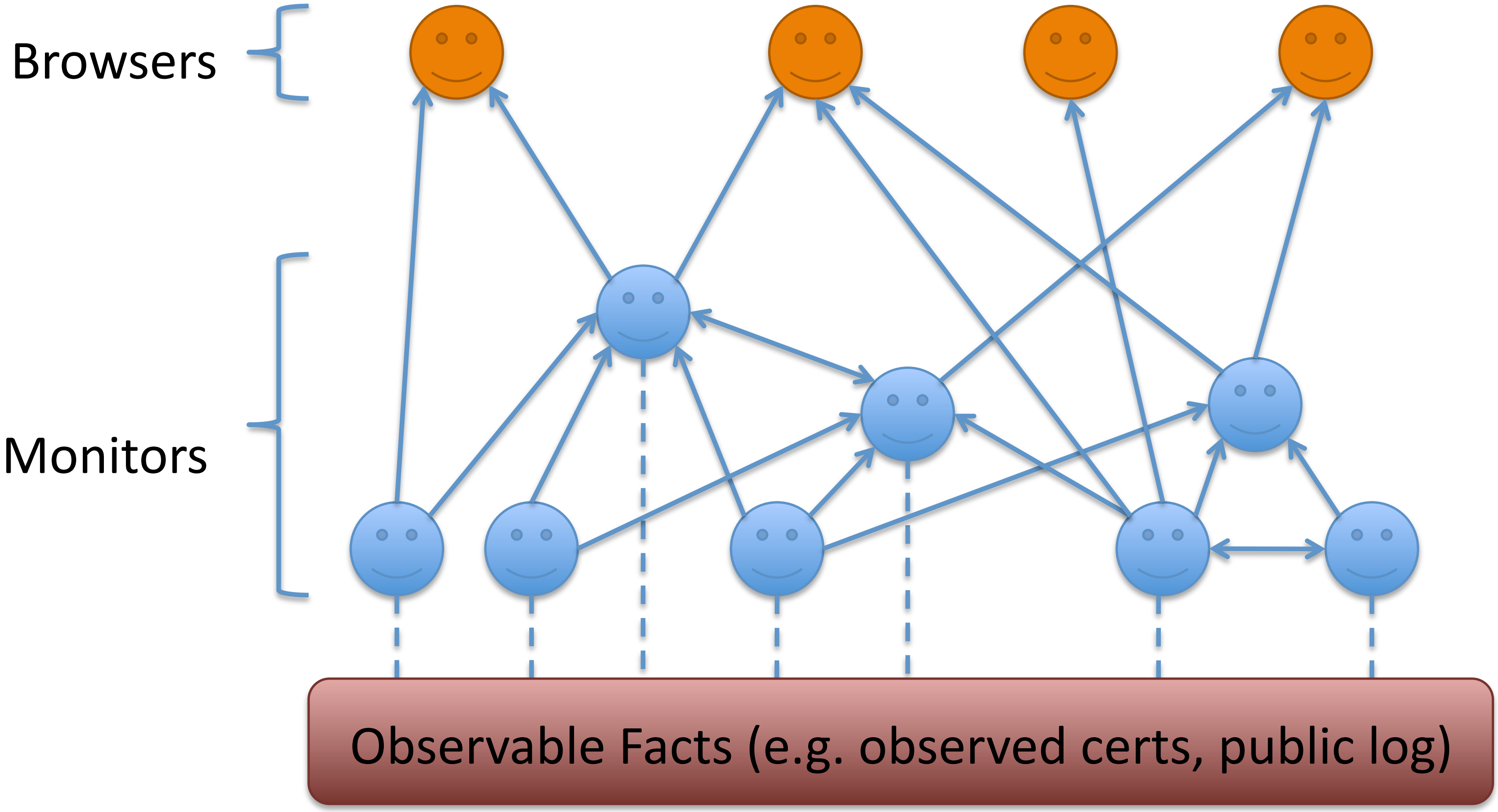
Browser

Web Server

Convergence



Trust Agility in action



Observational Trust Modes

- Net Perspective: “Do you see what I see?”
- Key Continuity: “Is this the same as before?”
- SSH, Convergence, Perspectives, etc.
- Rationale: Internet works for most people most of the time

Convergence Challenges

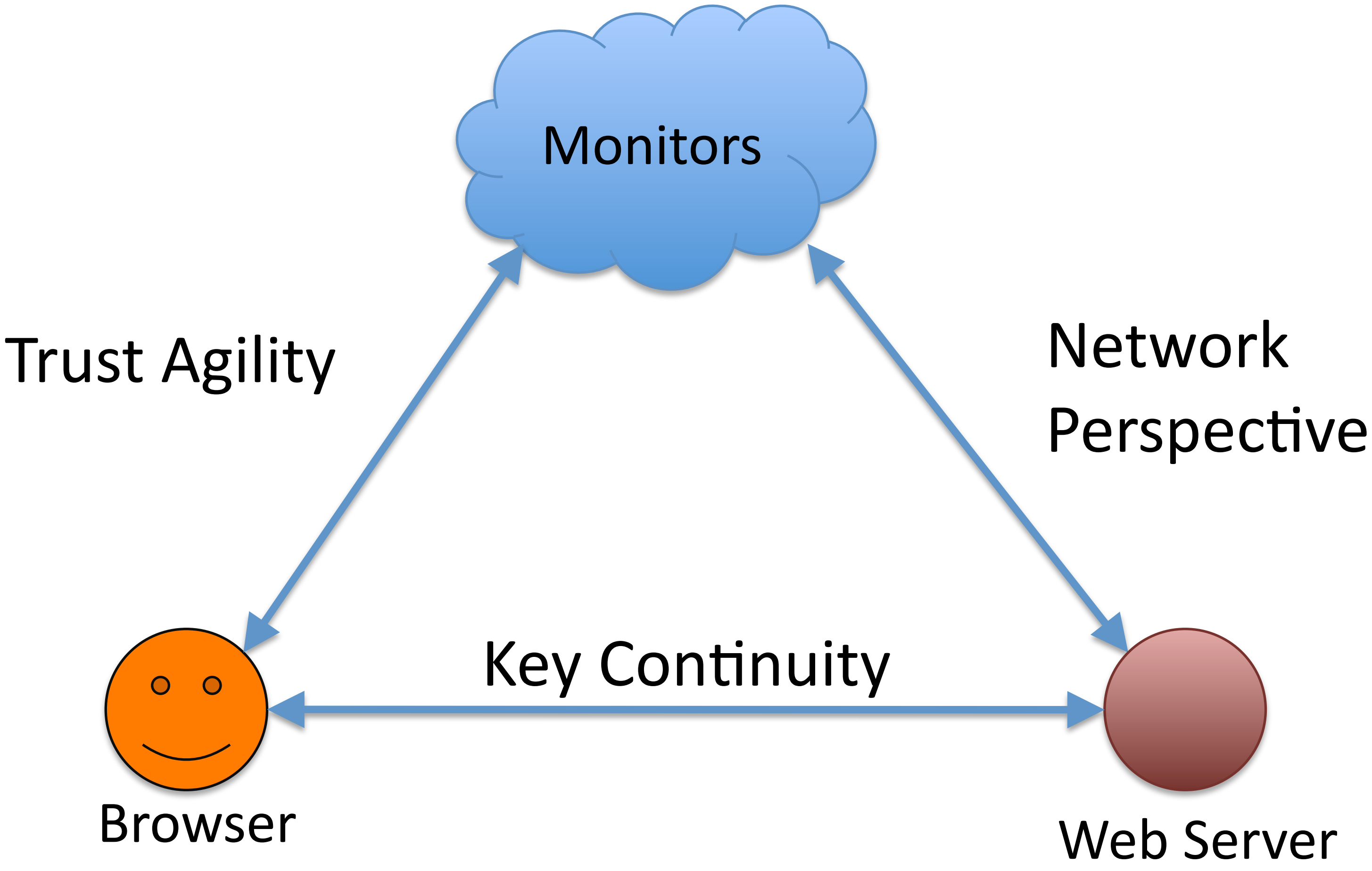
- Online lookups
  - Performed on first connection or key discontinuity
  - Costly infrastructure
  - Performance and reliability risk

Observational Trust Challenges

- Key Continuity
  - Doesn't protect initial connection
  - Doesn't handle key changes well

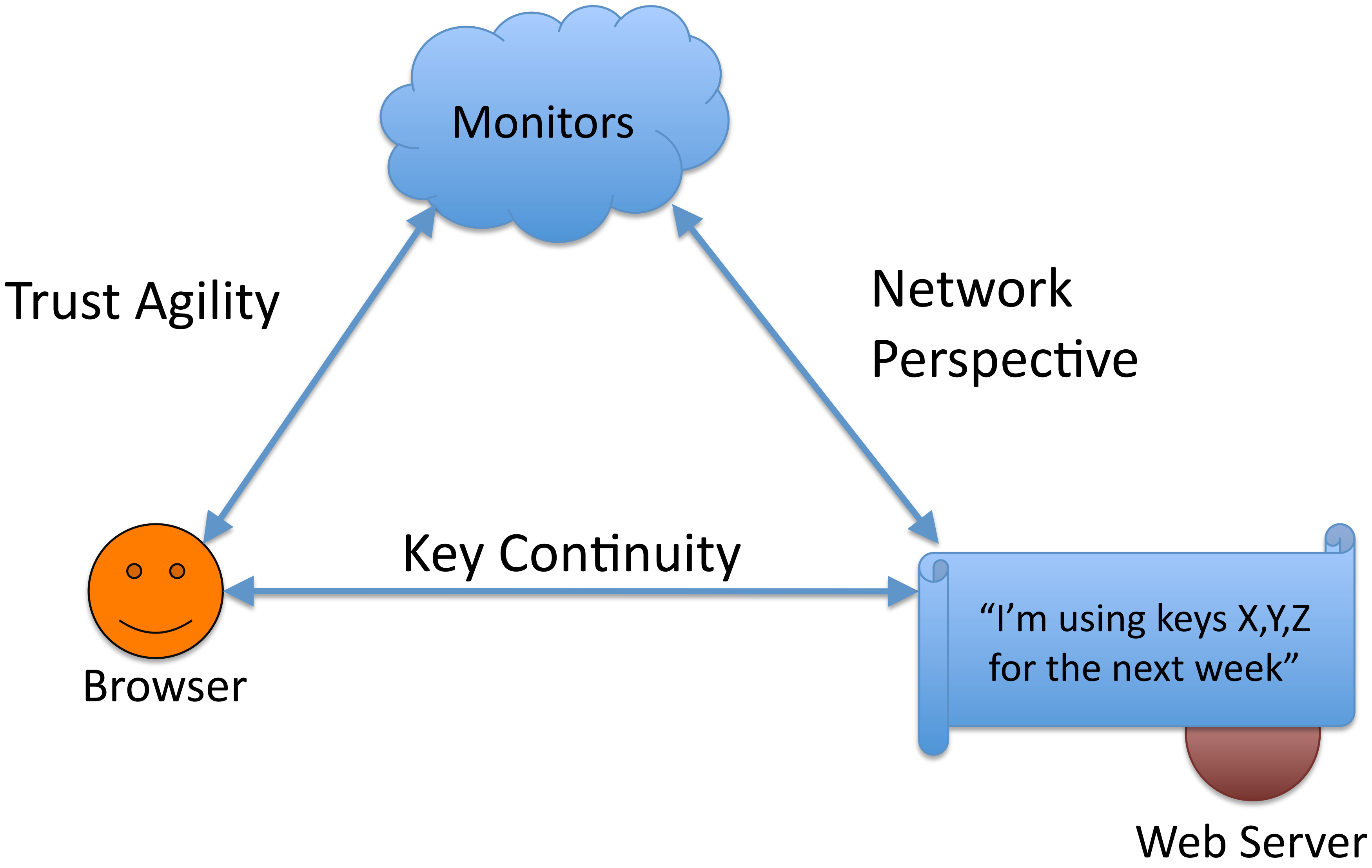
- Network Perspective
  - Handles initial connection and key changes at cost of online lookups
  - Doesn't handle multiple-keys-per-site well

Observational Trust



Can we improve observational trust…

…with some help?



Server Asserted Pinning

- Improves reliability (server has made a commitment)
  - Regardless of multiple-keys-per-site or key change

- Can help with initial connection / online lookup
  - Gives us longer-lived "tokens" which can be distributed in different ways

Pins

- Pin = (Name, Authentication Data, Expiration)
- Authentication Data
  - Public key(s)
  - Opt-In (HSTS, DNSSEC, Certificate Transparency)

- How are pins asserted?

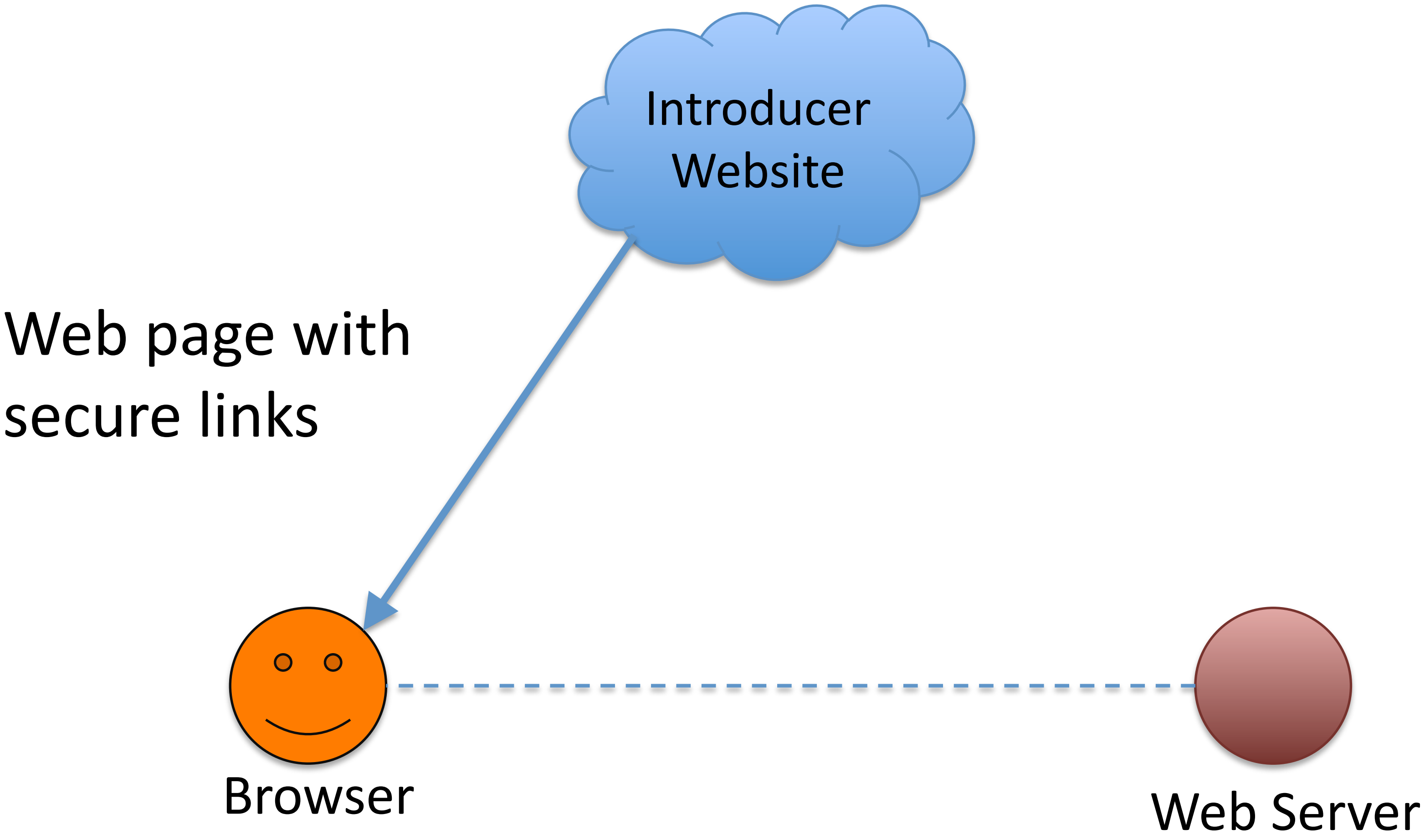
- How are pins distributed?

Distributing Pins

- Preloaded pins

- Key continuity

Secure Links



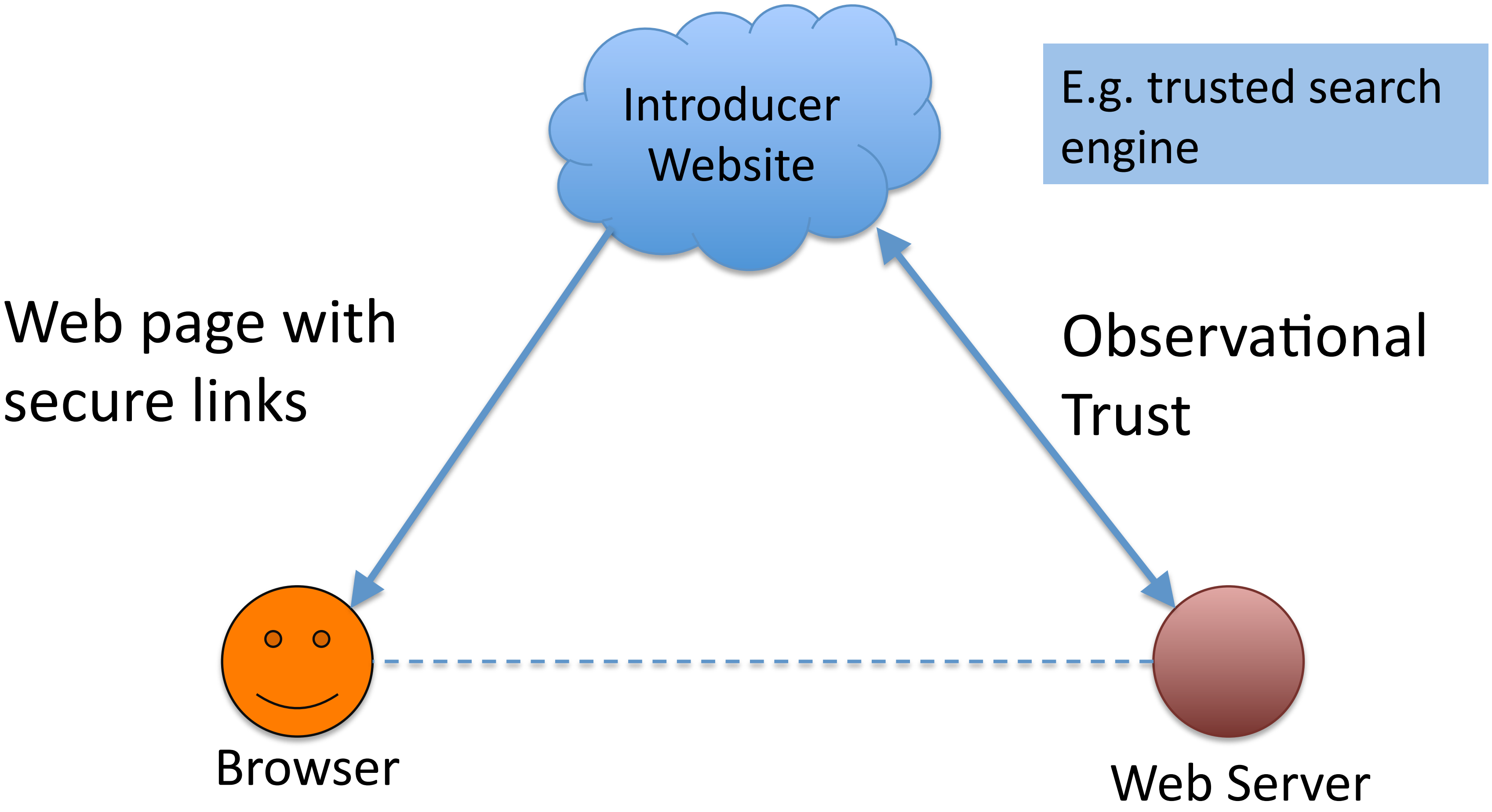
Introducer Website

Web page with secure links

Browser

Web Server

Secure Links



Secure Links

```
<a link-security="expiry=1357849989;
pin-sha256=YWRmYXNkZmFzZGZhc2RmcXdlcnF3ZXJxd2VycXdlcnF=;
pin-sha256=LPJNul+wow4m6DsqxbninhsWHlwfp0JecwQzYpOLmCQ=;"
href="https://www.example.com">a secure link!</a>
```

Secure Links

- Use current trust model on the web
  - A broken link is the introducer’s fault
- Build on trust in the web’s major “hubs”
  - Search engines, social networks, link shorteners
- Also useful for loading page resources securely
  - i.e. JavaScript libraries
- Feedback welcome: www.secure-links.org

Asserting Pins

- HPKP
  - HTTP layer, pins to EE keys and/or CA keys
- TACK
  - At TLS layer, pins to self-chosen signing key

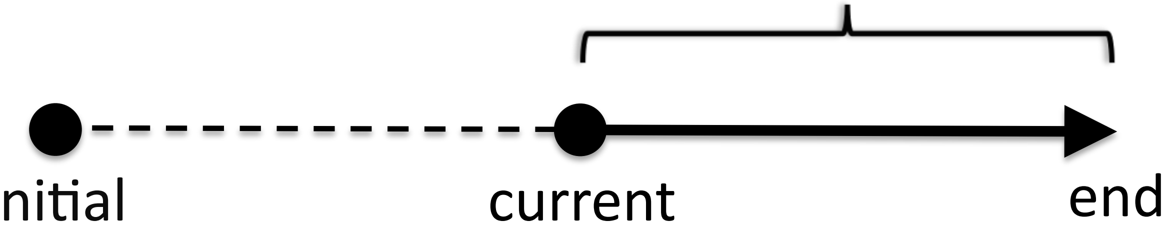
Pin Assertion Challenges

- Risks to relying party
  - Bad pins

- Risks to asserting party
  - Key loss
  - Key compromise
  - Inflexible / impossible key changes

Pin Activation

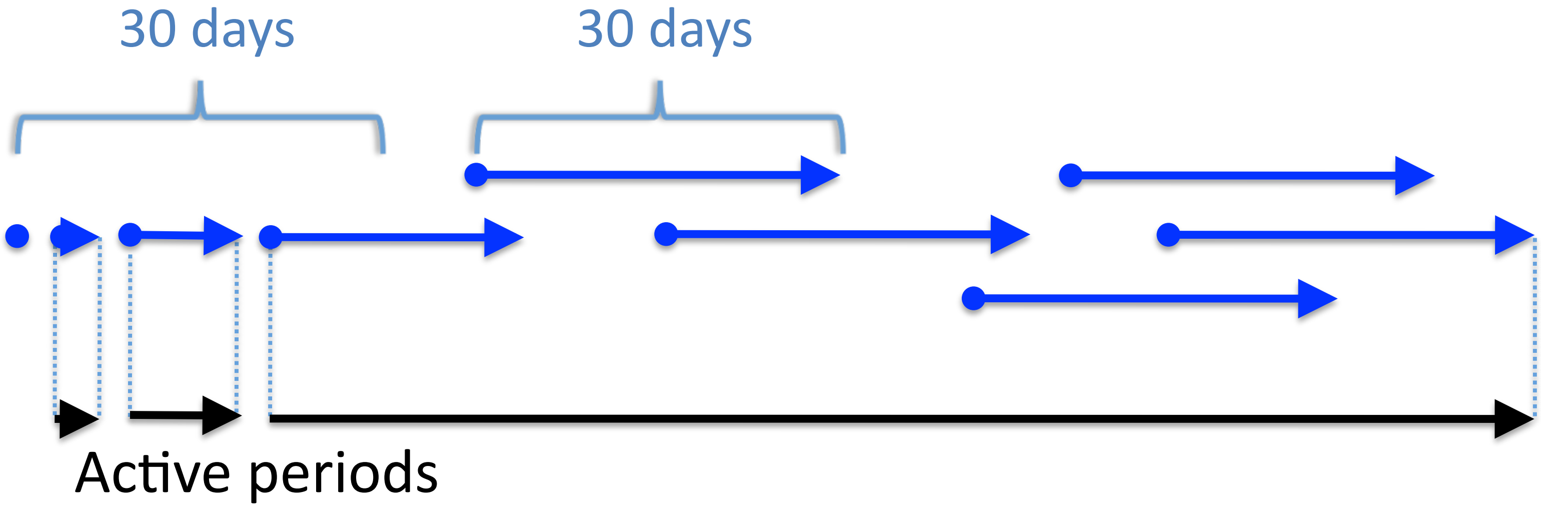
Active period duration = MIN(30 days, current – initial)



initial

current

end

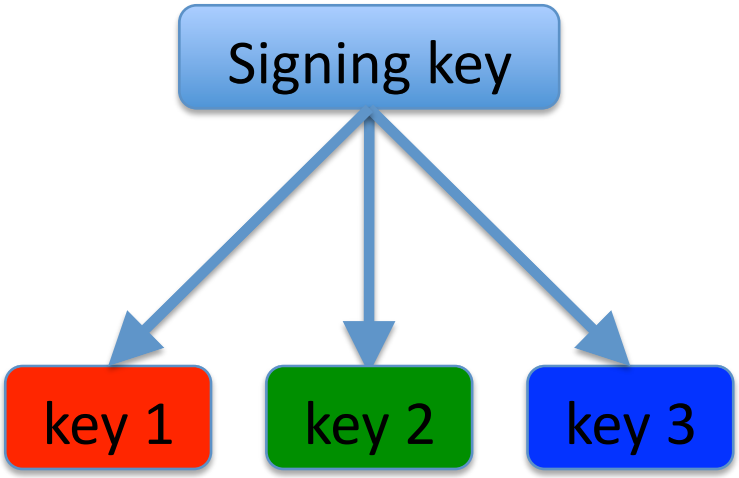


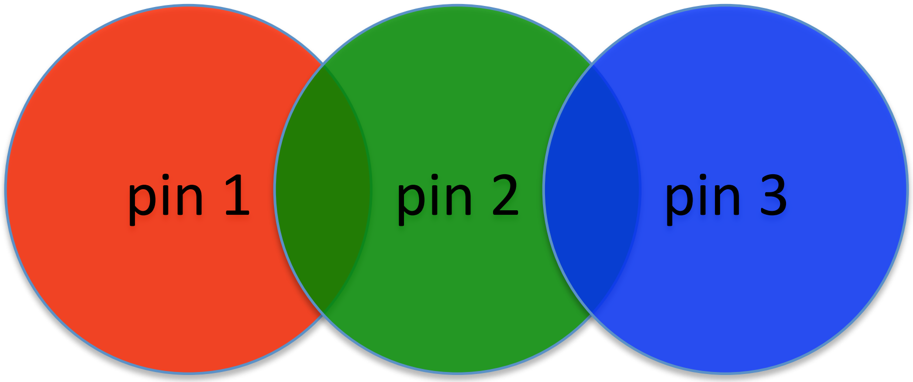
30 days

30 days

Active periods

Pin flexibility





Ex: (**K1**,K2,K3,K4) ➔ (**K3**,K4,K5,K6) ➔ …

Shifting pins could use CA or EE keys

Pin Redundancy

- Pin to multiple public keys (HPKP)
  - E.g. several popular CAs and your TLS key

- Distributed backup / delegation of private key
  - E.g. TACK

Summary

- Lots to think about
- Oh, and we can combine lots of these things!
  - Sovereign Keys ~= transparency + pinning

Thanks!

- http://dnssec-deployment.org
- http://www.certificate-transparency.org
- http://convergence.io
- http://tack.io
- http://tools.ietf.org/html/draft-ietf-websec-key-pinning
- https://www.eff.org/sovereign-keys
- http://www.secure-links.org