Certificate Cothority: Towards Trustworthy Collective CAs

Ewa Syta, Iulia Tamas, Dylan Visher, David Wolinsky, Bryan Ford

Yale University

HotPETs 2015

"Authorities" are Everywhere

- **Conceptually simple** but **security-critical** services
  - Logging and Time-stamping Authorities
  - Naming Authorities
  - Randomness Authorities (e.g., lotteries)
  - Digital Notaries
  - Certificate Authorities (CAs)

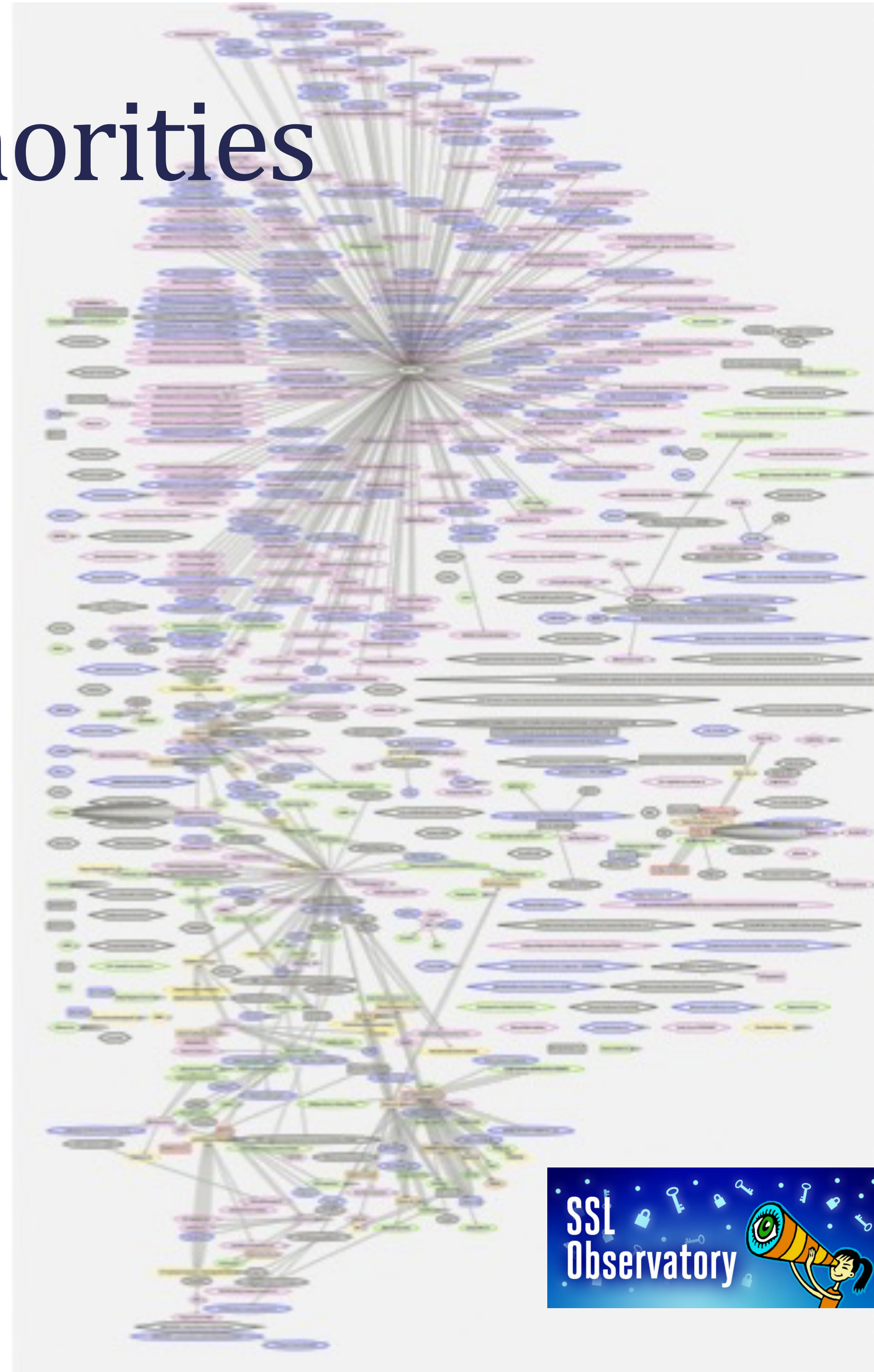
Talk Outline

- **Troubles with Certificate Authorities**
- Designing Certificate Cothorities
  - Scalable Collective Schnorr Log-Signing
  - The Availability Problem
- Prototype and Preliminary Results
- Deployment Scenarios
- Conclusions

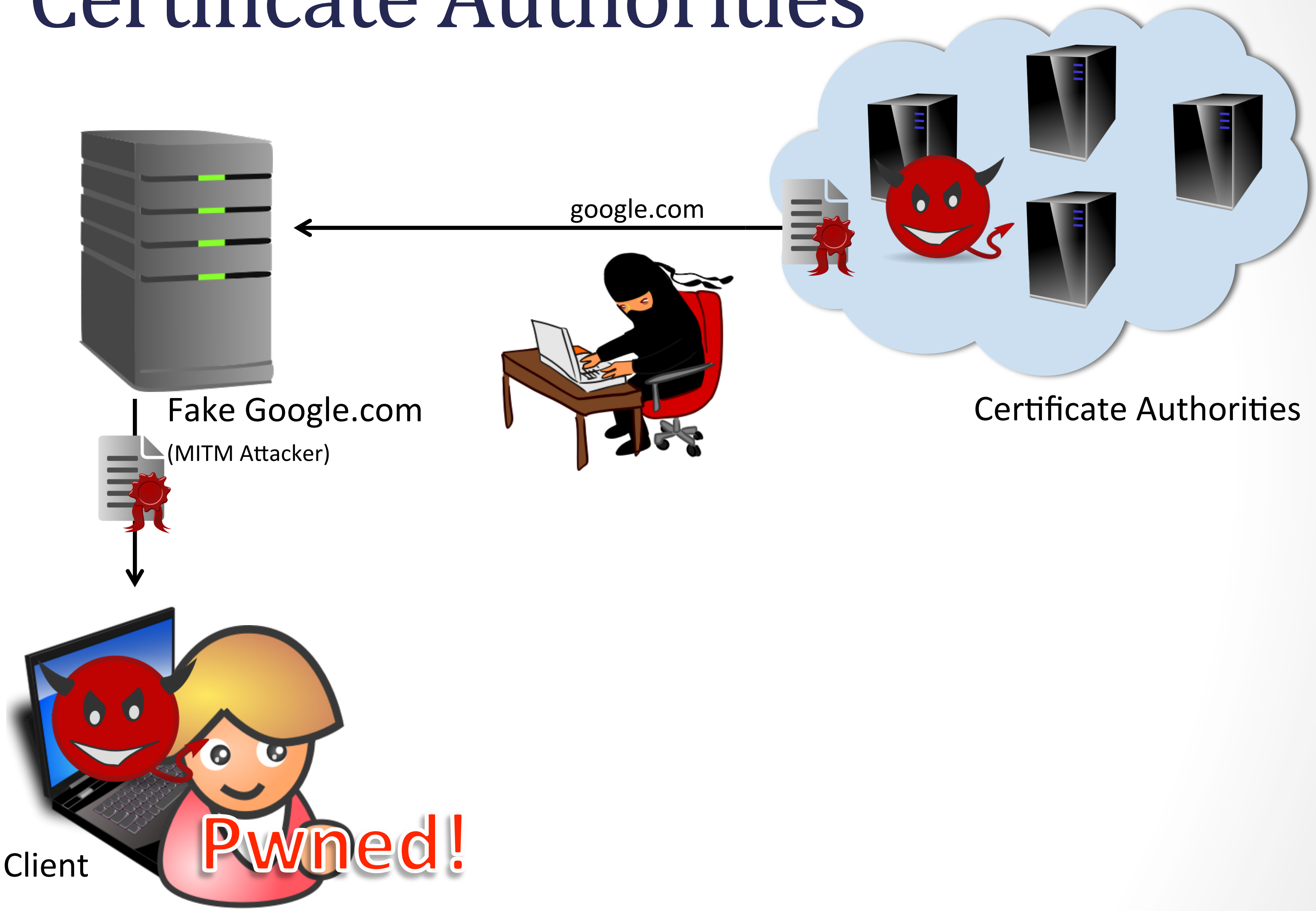
Certificate Authorities

EFF SSL Observatory

- ~650 CAs trusted by Mozilla or Microsoft
- Any CA can issue certs for any domain
- Prime key target
  - MITM attack power
- Breaches do happen
  - DigiNotar'11
  - Comodo'11
  - CNNIC/MCS'15



Certificate Authorities



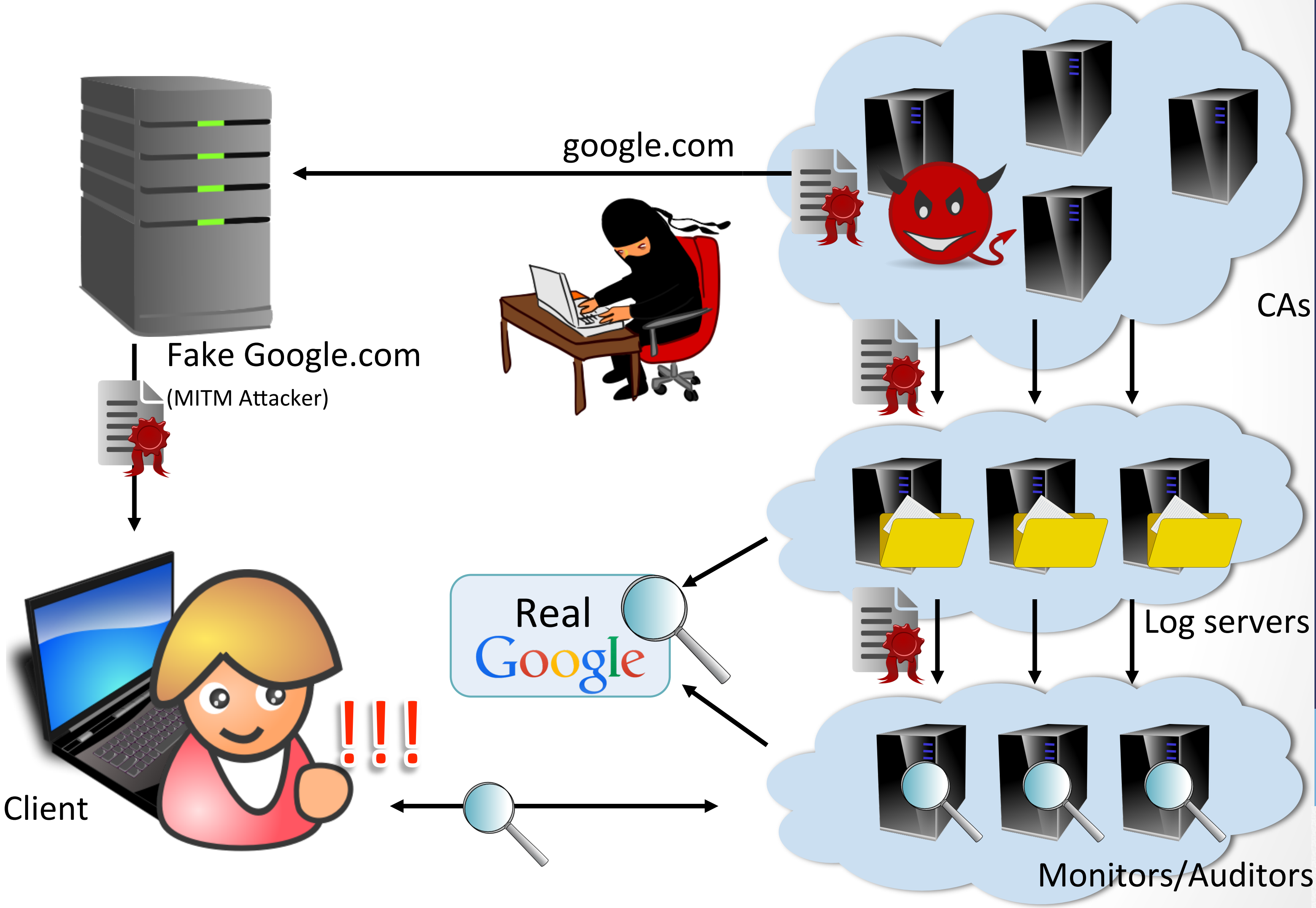
If we trust **many** CAs…

- Attacker gets to choose which one to attack
  → Weakest-link security overall

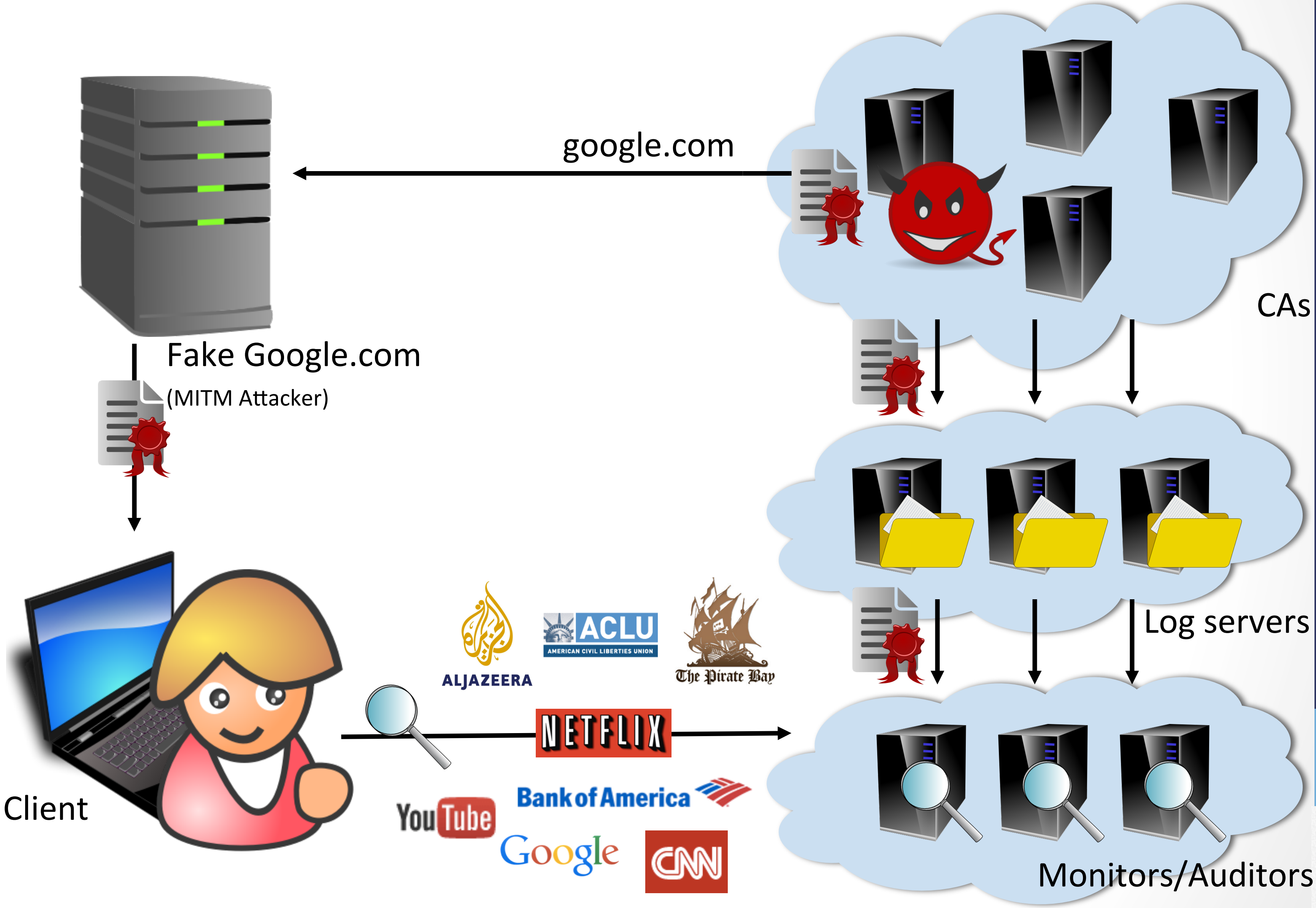
Current Defenses

- Oversight from industry organizations, browser and OS vendors
- Pinning: embed certificates/CAs into the browser
- Logging and monitoring
  - Certificate Transparency (CT) [Laurie'11]
  - Convergence [Marlinspike'11]
  - AKI [Kim'13]
  - ARPKI [Basin'14]
  - PoliCert [Szalachowski'14]

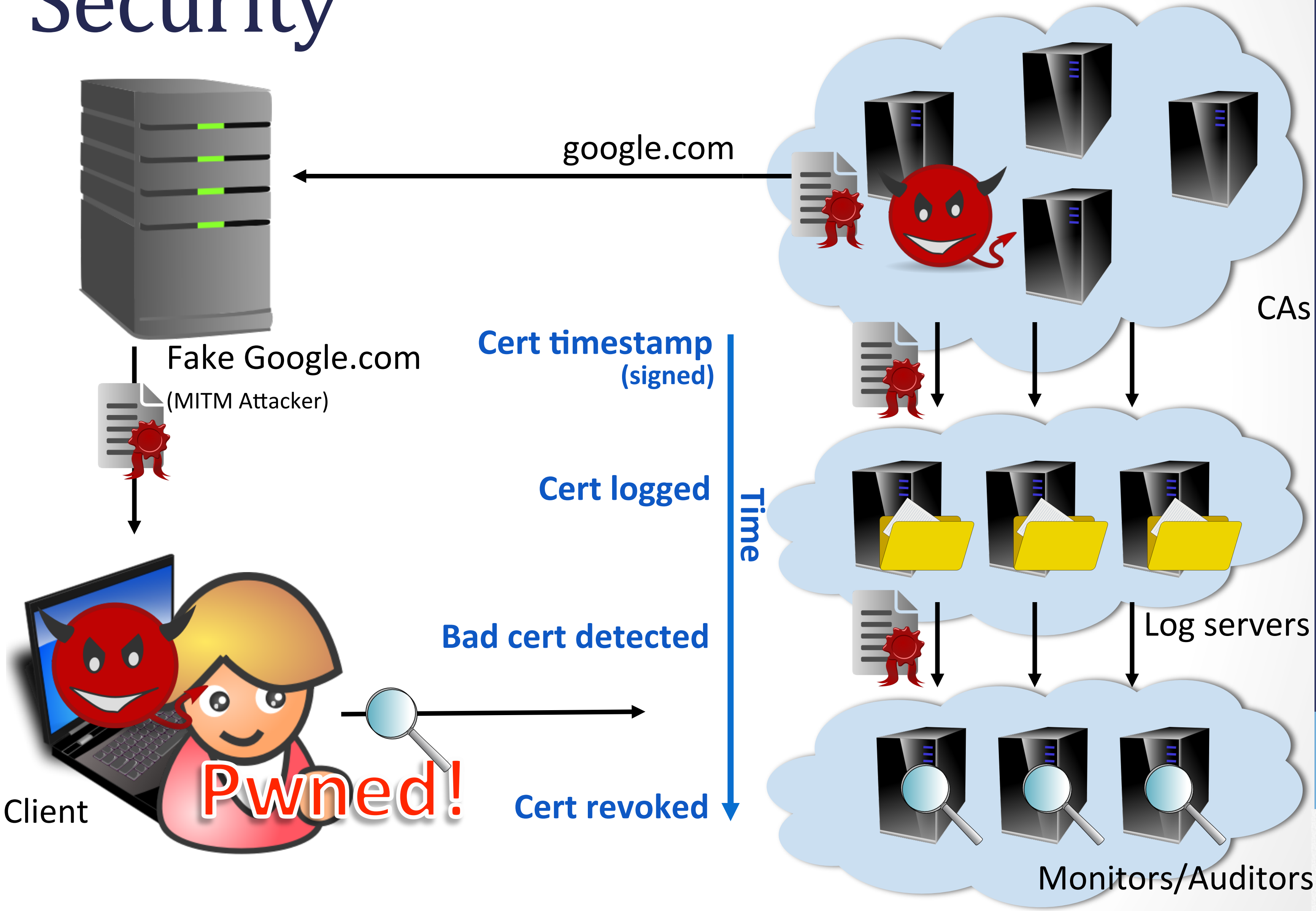
Certificate Transparency



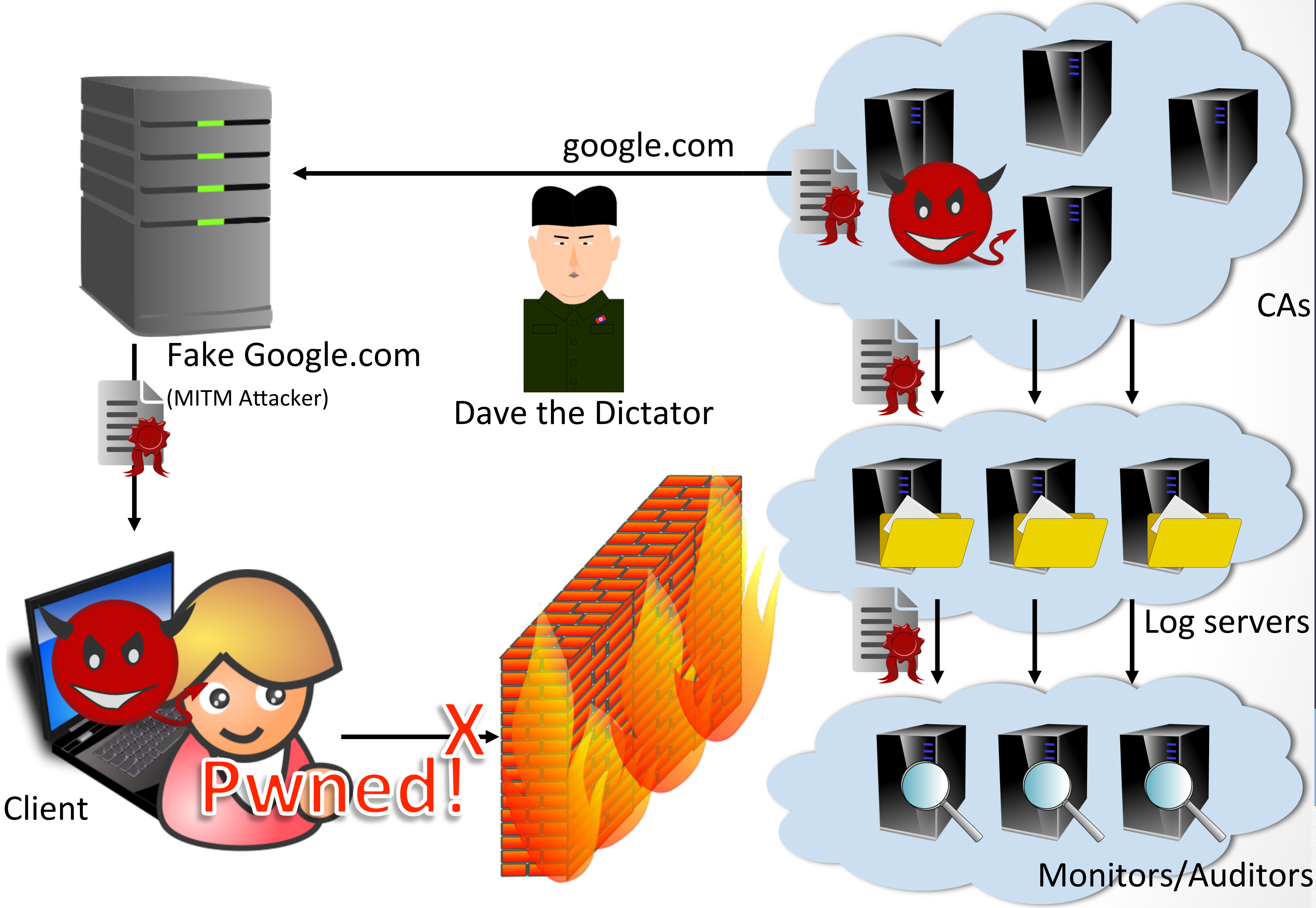
CT's Weakness: Privacy



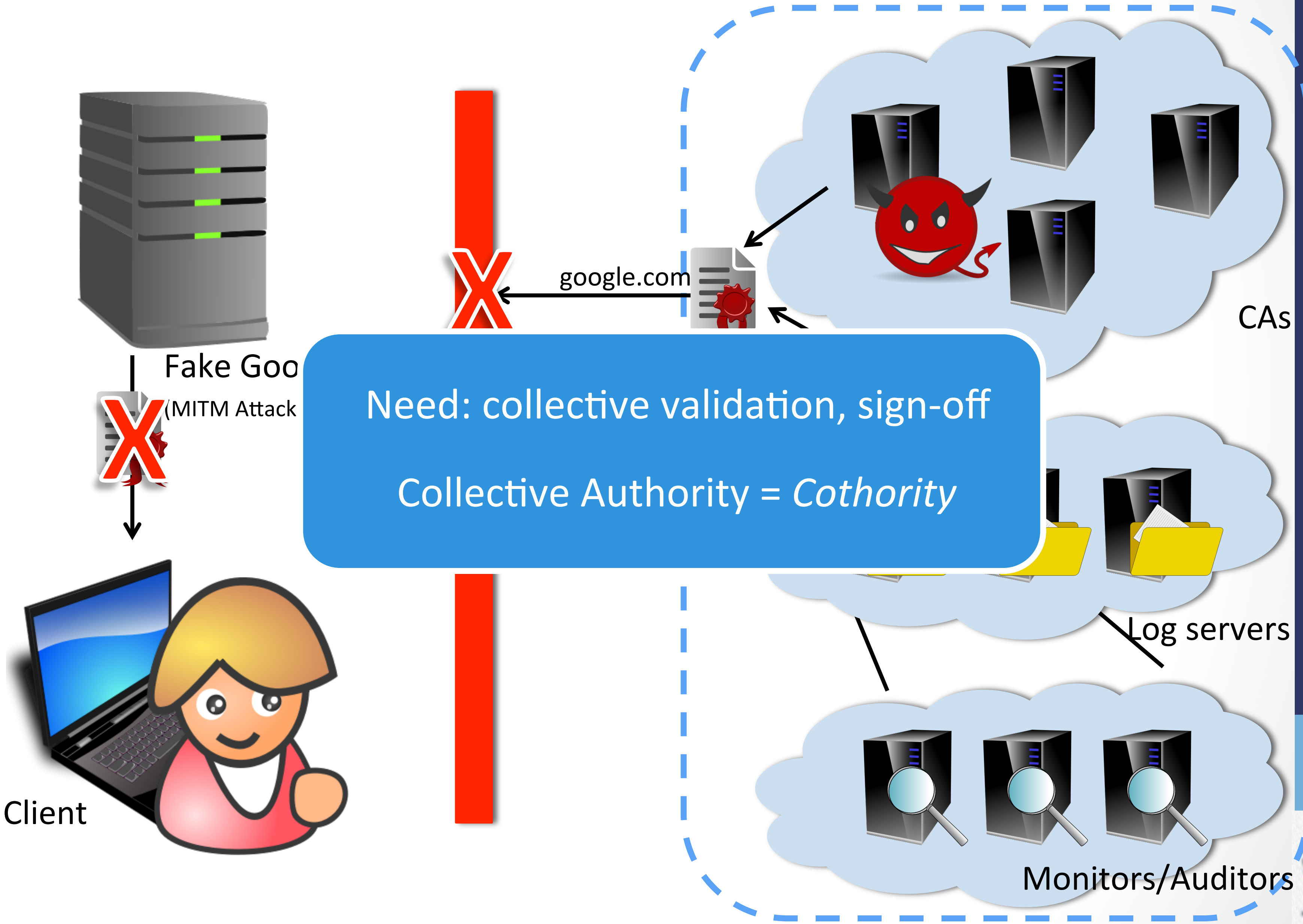
CT's Weakness: Retroactive Security



CT's Weakness: Blocking



We need "Collective Authorities"



Need: collective validation, sign-off

Collective Authority = *Cothority*

Talk Outline

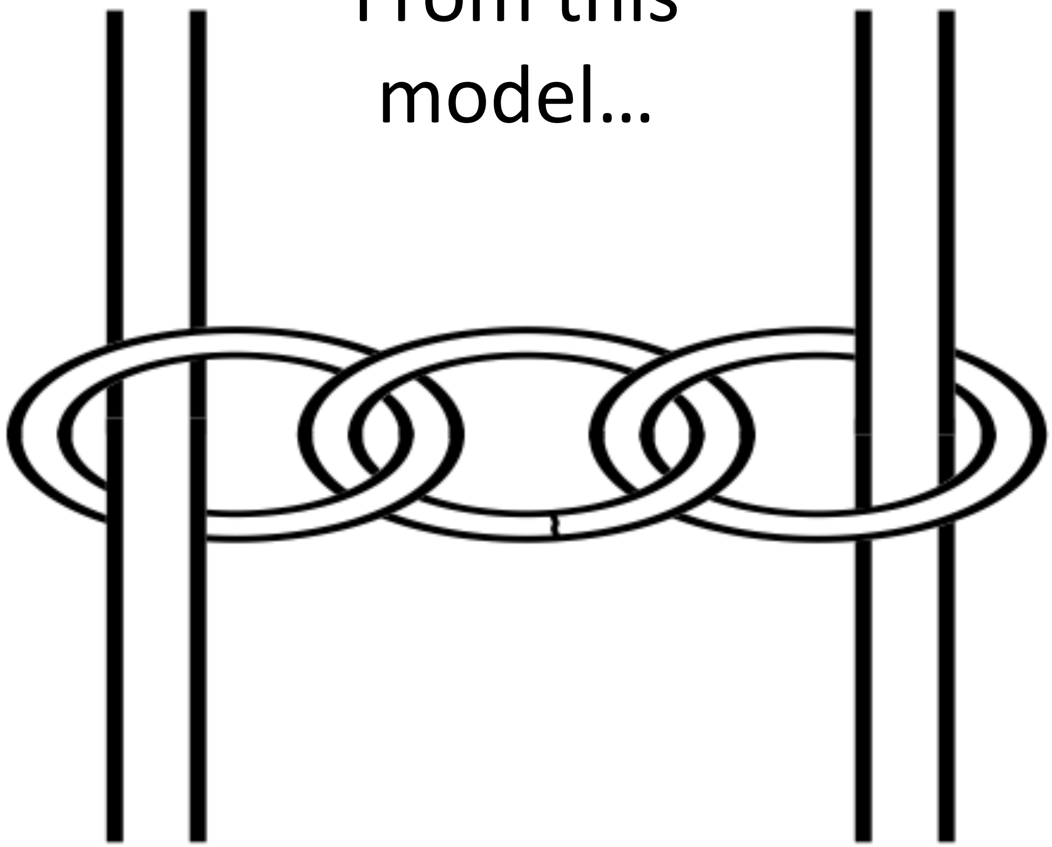
- Troubles with Certificate Authorities
- **Designing Certificate Cothorities**
  - Scalable Collective Schnorr Log-Signing
  - The Availability Problem
- Prototype and Preliminary Results
- Deployment Scenarios
- Conclusions

Certificate Cothority (CC)

- Many parties collectively sign, not just a single CA
  - All participating CAs can propose new certs, all verify
  - Hundreds or thousands of diverse participants
    - CAs, log servers, monitors, auditors
    - Easy to include new participants
- Collective signature = *many* servers sign off
  - Any CA can block signature if cert violates policy
  - Simple verification as if there is one CA
  - Secure unless *many* servers compromised

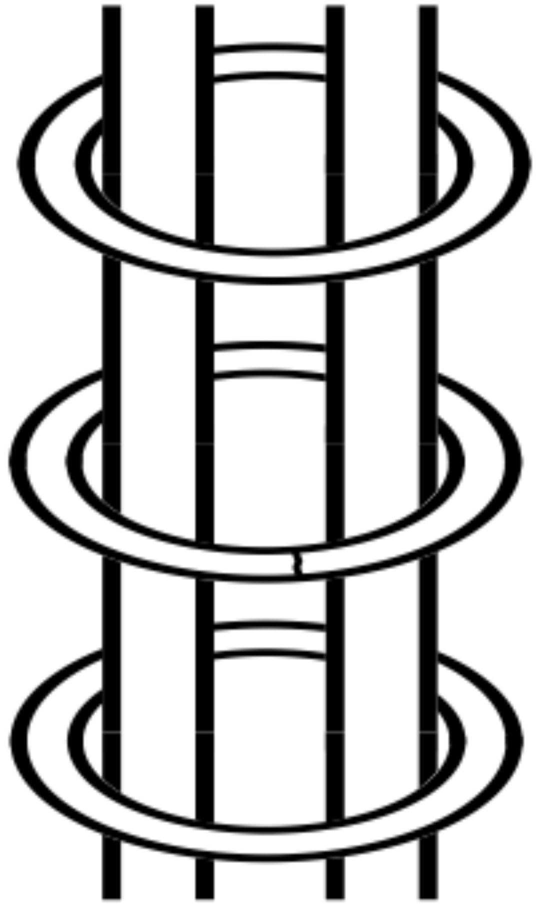
Why Certificate Cothority?

From this
model…



Why Certificate Cothority?

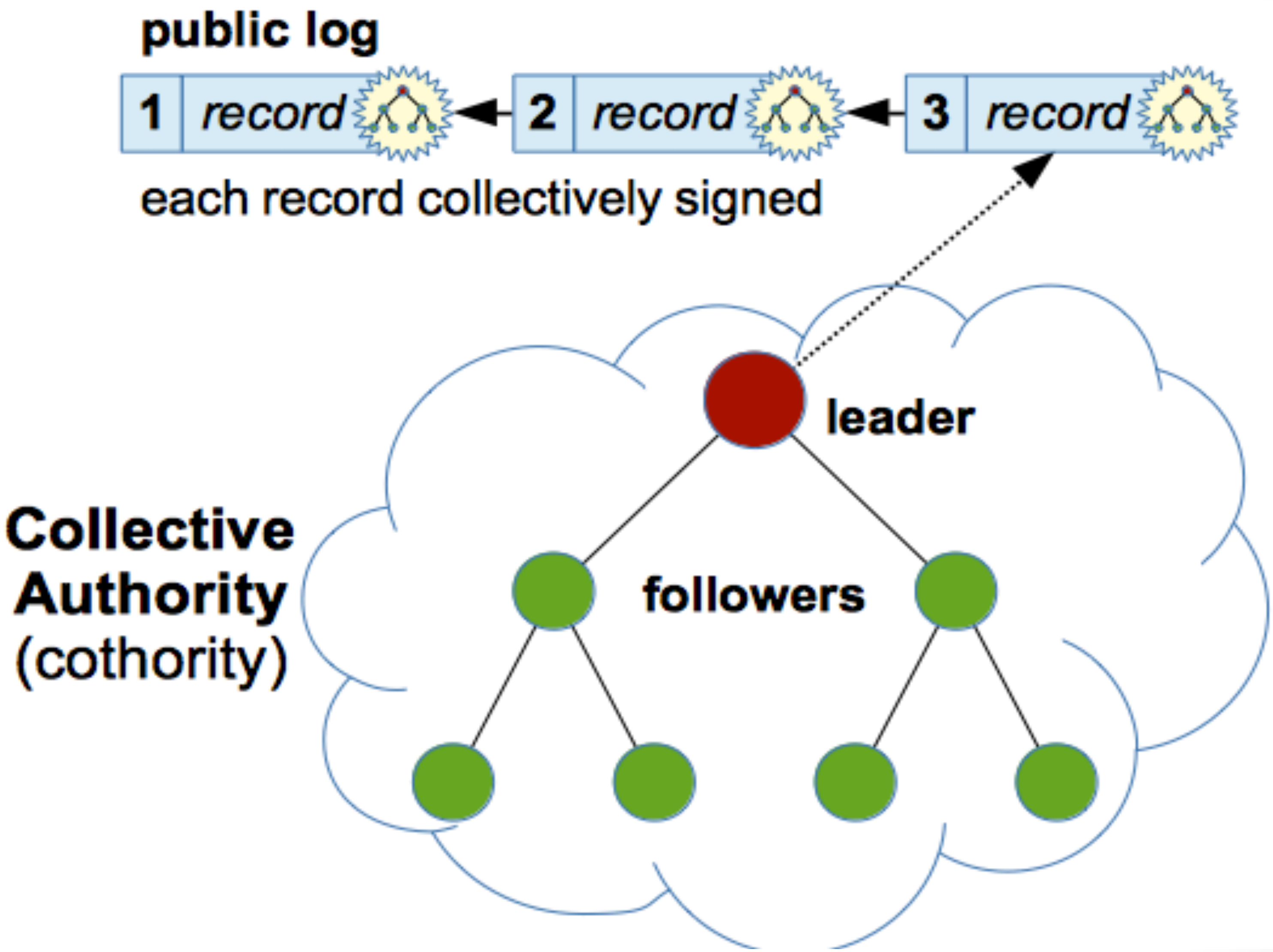
To this model



Talk Outline

- Troubles with Certificate Authorities
- Designing Certificate Cothorities
  - **Scalable Collective Schnorr Log-Signing**
  - The Availability Problem
- Prototype and Preliminary Results
- Deployment Scenarios
- Conclusions

CoSi: Collective Signing



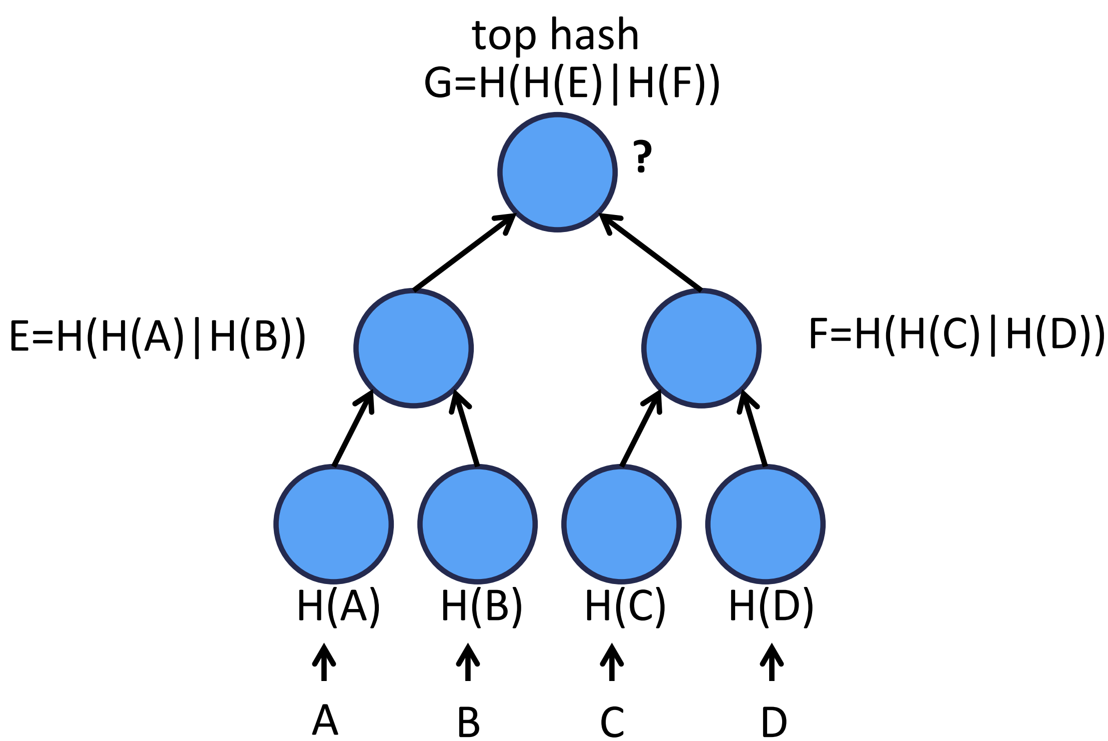
CoSi: Scalable Collective Signing

- CoSi builds upon existing primitives
  - Merkle Trees [Merkle'79]
  - Schnorr Signatures [Schnorr'89] and Multisignatures [Itakura'83],[Ohta'99],[Micali'01],[Bellare'06]

- **Our contribution**
  - Scale multisignatures to thousands of nodes
  - Communication trees and aggregation, as in scalable multicast protocols

Merkle Trees

- Every non-leaf node labeled with the hash of the labels of its children.
- Efficient verification of items added into the tree



Schnorr Signature

- Generator g of prime order q group
- Public/private key pair: $(K=g^k, k)$

	Signer		Verifier
Commitment	$V=g^v$	→	V
Challenge	c	←	$c = H(M\|V)$
Response	$r = (v - kc)$	→	r

Signature on M: (c, r)

Commitment recovery	$V' = g^r K^c = g^{v-kc} g^{kc} = g^v = V$
Challenge recovery	$c' = H(M\|V')$
Decision	$c' = c$?

Collective Signing

- Our goal is collective signing with N signers
  - Everyone produces a signature
  - N signers-> N signatures -> N verifications!
  - Bad for hundreds or thousands of signers!

- Better choice – a multisignature

Schnorr Multisignature

- Key pairs: $(K_1=g^{k_1}, k_1)$ and $(K_2=g^{k_2}, k_2)$

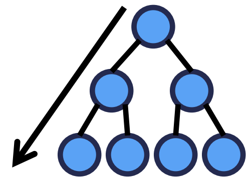
	Signer 1		Signer 2	Verifier		
Commitment	$V_1=g^{v_1}$		~~$V_2=g^{v_2}$~~ →	V_1	V_2	$V=V_1*V_2$
Challenge	c	c	←	$c = H(M\|V_1)$	$c = H(M\|V)$	
Response	$r_1 = (v_1 - k_1c)$		~~$r_2 = (v_2$~~ → $k_2c)$	r_1	r_2	$r=r_1+r_2$

Collective Signature on M: (c, r)   Same signature!

Commitment recovery	Same verification!	$V' = g^rK^c$	$K=K_1*K_2$
Challenge recovery	Done once!	$c' = H(M\|V')$	
Decision		$c' = c$?	

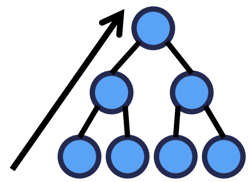
CoSi Protocol

1. Announcement Phase



M

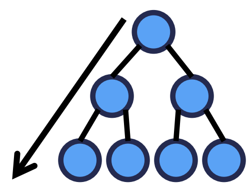
2. Commitment Phase



$\underline{V}_1 = V_1 V_2 \ldots V_N$ (aggregate)

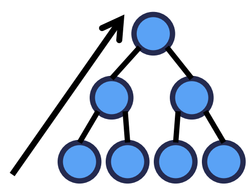
$V_3 = g^{v3}$ (individual)

3. Challenge Phase



$c = H(M|root)$

4. Response Phase



$\underline{r}_1 = r_1 + r_2 + \ldots + r_N$ (aggregate)

$r_3 = v_3 - k_3 c$ (individual)

Collective signature $(c, \underline{r}_1)$

Talk Outline

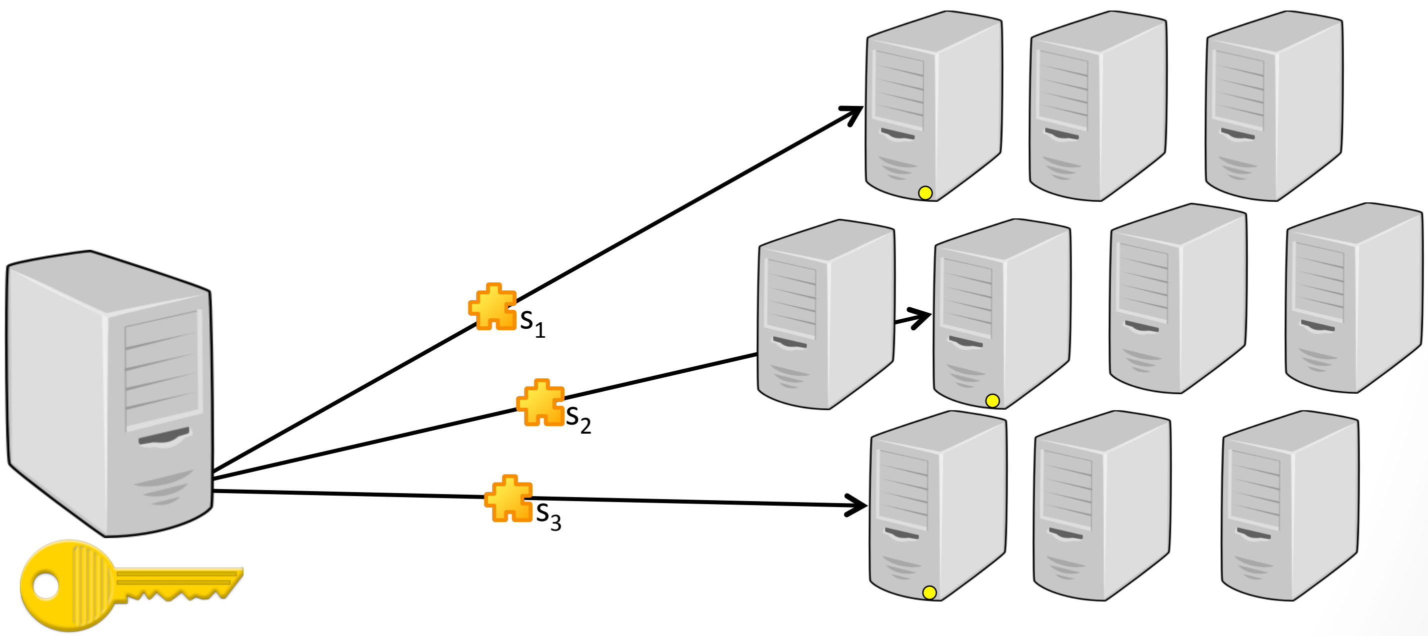
- Troubles with Certificate Authorities (CAs)
- Designing Certificate Cothorities
  - Scalable Collective Schnorr Log-Signing
  - **The Availability Problem**
- Prototype and Preliminary Results
- Deployment Scenarios
- Conclusions

Exceptions

- If node A fails, the remaining nodes can still provide a valid signature but
  - For a modified collective key: $K' = K * K^{-1}_A$
- Client gets a signature under K' and an exception e_A
  - e_A also lists conditions under which it was issued
- Client accepts **only** if a quorum of nodes maintained

Life Insurance Policy

- Node "insures" its private key by depositing the key shares with other servers (insurers)
- If node fails, others recover the key and continue
- Use verifiable secret sharing



Talk Outline

- Troubles with Certificate Authorities
- Designing Certificate Cothorities
  - Scalable Collective Schnorr Log-Signing
  - The Availability Problem
- **Prototype and Preliminary Results**
- Deployment Scenarios
- Conclusions

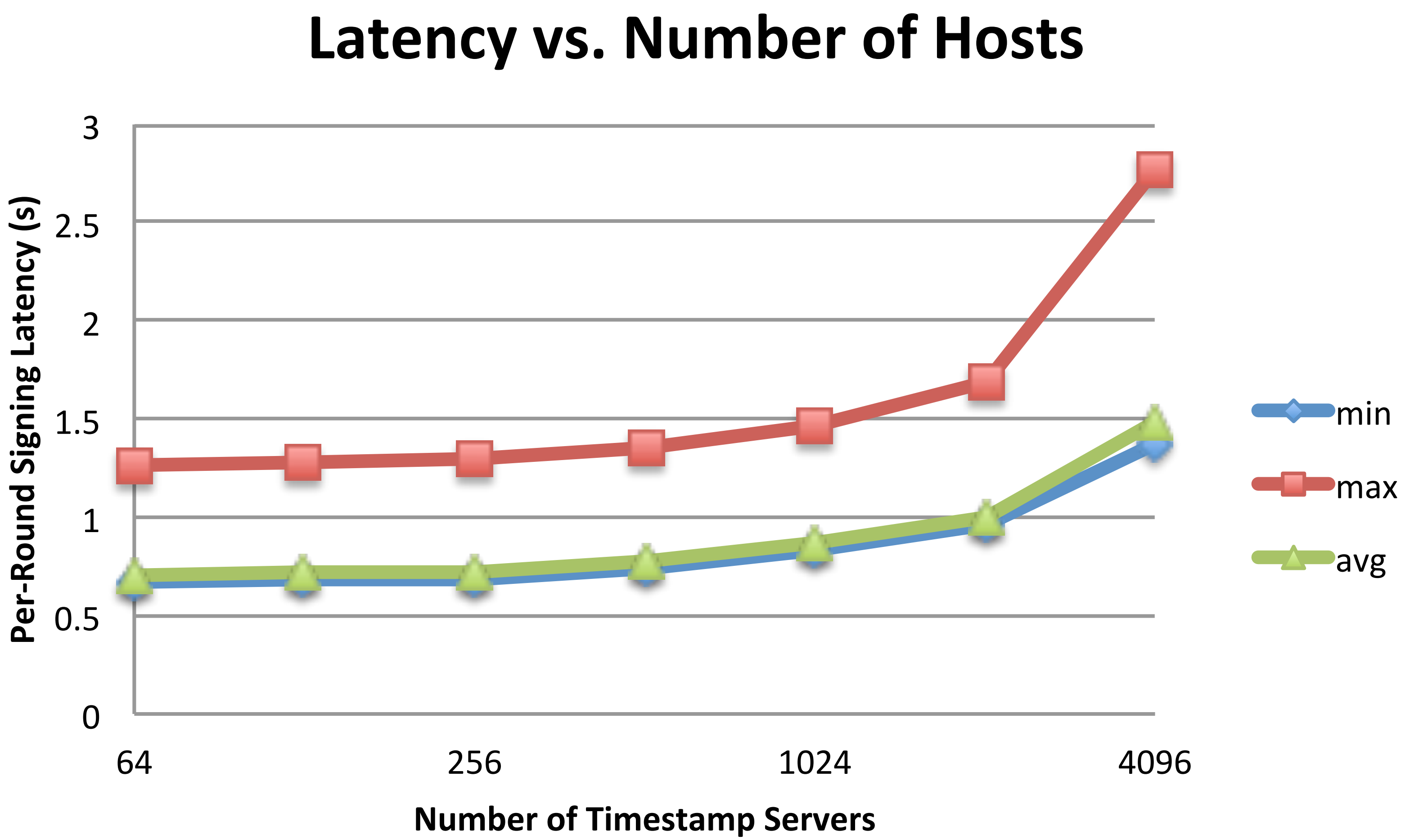
Implementation

- Implemented in Go with DeDis crypto library
  - https://github.com/DeDiS/prifi/tree/master/coco
  - https://github.com/DeDiS/crypto

- Schnorr multisignatures on Ed25519 curve
  - AGL's Go port of DJB's optimized code

- Run experiments on DeterLab
  - Up to 4096 virtual CoSi nodes
  - Multiplexed atop up 32 physical machines
  - Latency: 100ms roundtrip between two servers

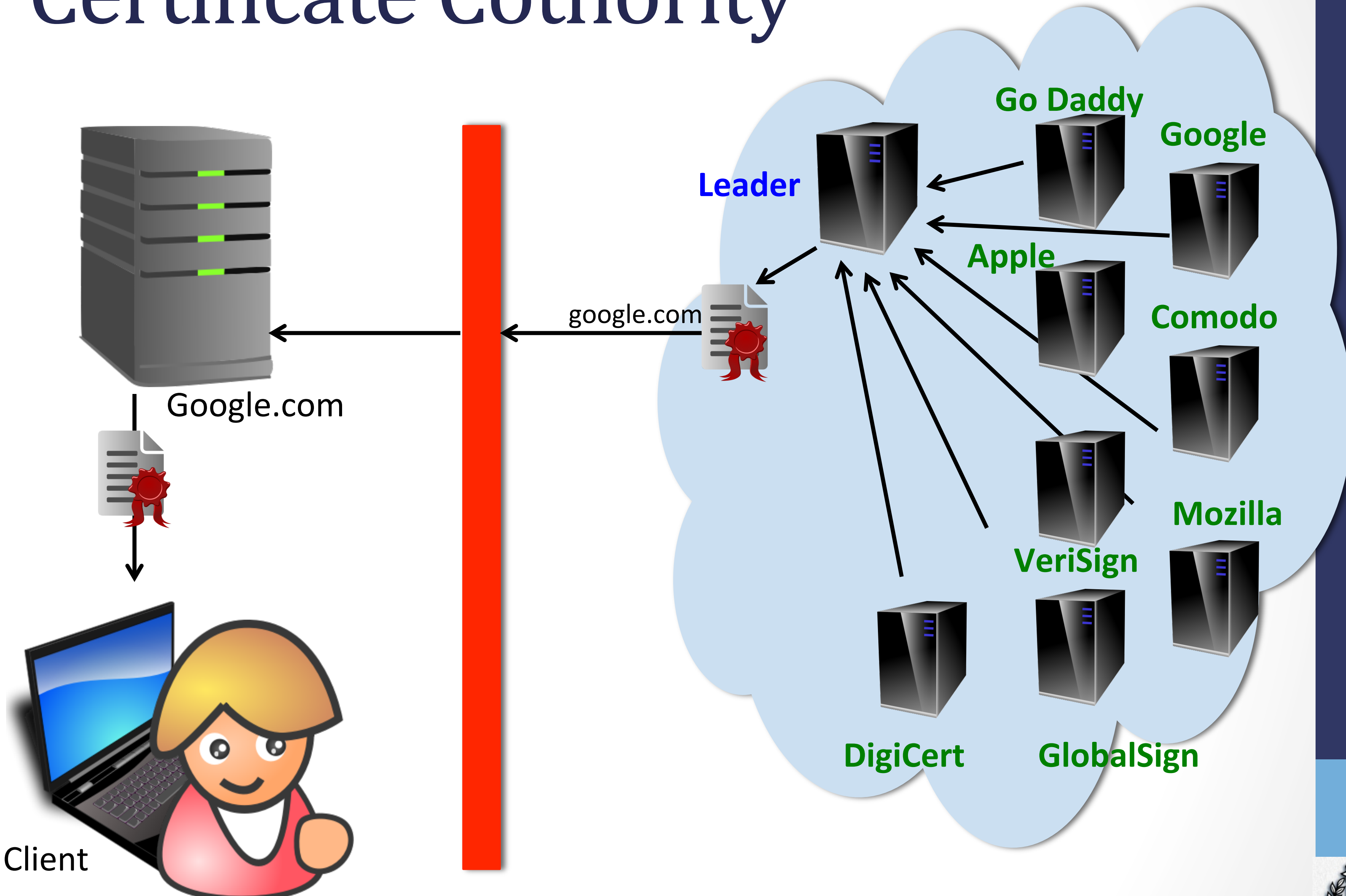
Preliminary Results



Talk Outline

- Troubles with Certificate Authorities
- Designing Certificate Cothorities
  - Scalable Collective Schnorr Log-Signing
  - The Availability Problem
- Prototype and Preliminary Results
- **Deployment Scenarios**
- Conclusions

Certificate Cothority



Deployment Scenarios

Most Ambitious

- **Ideal case: everyone in certificate cothority**
  - Everyone gets to check certs but difficult to deploy
- **Browser-driven certificate cothority**
  - Browser vendor acts as a CC leader and CAs gradually join (eventually must) to remain in the root store
- **Root-CA-centric certificate cothority**
  - Root-CA as a leader and intermediate CAs gradually join (eventually must) to retain their signing power
- **Log server-driven certificate cothority**
  - Backward compatible
  - CT-style: endorse signed certificate timestamps (SCTs)

Most Deployable

YALE

Conclusions

- **We can and should build a better CA system**
  - There seem to be no technical reason not to!
  - Proactively secure: no bad certs endorsed
  - Privacy-friendly: users don't gossip their browsing history
- **Build it using *cothorities***
  - Strongest-link security
  - Built upon well-understood cryptographic primitives
  - Scale to thousands of participants with reasonable delays
- But it will **definitely** take time and effort

Thank you!

Let's chat :)

More details

"Decentralizing Authorities into Scalable Strongest-Link Cothorities"

arXiv:1503.08768

ewa.syta@yale.edu