Collective Authorities:

Securely Decentralising Trust at Scale

https://github.com/dedis/cothority

32C3

December 27, 2015

Who are we?

Philipp Jovanovic, @Daeinar, EPFL

Ismail Khoffi, EPFL

Ewa Syta, Iulia Tamas, Dylan Visher, David Isaac Wolinsky,
Yale University, USA

Linus Gasser, Nicolas Gailly, Bryan Ford,
EPFL, CH

Code: https://github.com/dedis/cothority
Mailing list: https://groups.google.com/forum/#!forum/cothority

HACK OBTAINS 9 BOGUS CERTIFICATES FOR PROMINENT WEBSITES; TRACED TO IRAN

DigiNotar scandal worsens: 500+ rogue certificates issued, five CAs breached

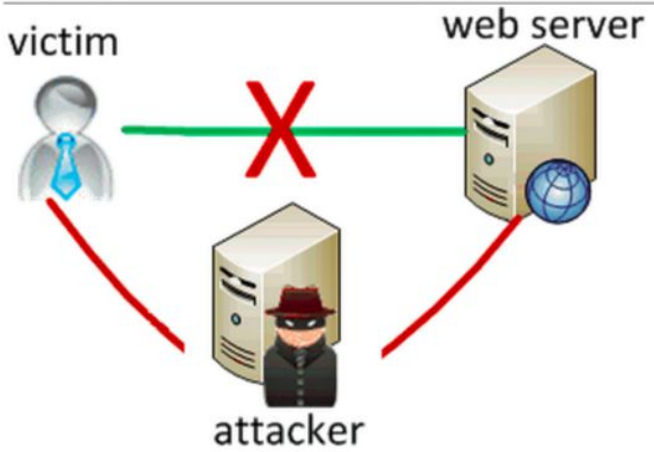
Trustwave Admits It Issued A Certificate To Allow Company To Run Man-In-The-Middle Attacks

Lenovo PCs ship with man-in-the-middle adware that breaks HTTPS connections [Updated]

Superfish may make it trivial for attackers to spoof any HTTPS website.

by **Dan Goodin** - Feb 19, 2015 5:36pm CET

Share Tweet 333



After Lenovo now Dell PCs and Laptops are shipping with rogue root level CA

1

BY VIJAY PRABHU ON NOVEMBER 23, 2015

SECURITY NEWS, TECHNOLOGY

Security

Second Dell backdoor root cert found

36

Blackhats, head straight to the airport lounge.

25 Nov 2015 at 05:00, Darren Pauli

CYBER CRIME SCAMS AND FRAUD

This Dude Hacked Lottery Computers To Win $14.3M Jackpot In U.S.

By Waqas on April 14, 2015 Email @hackread

CATEGORIES FEATURED PODCASTS VIDEOS

10/08/15 5:54

Advanced notice: Security updates for Adobe Acrobat and Reader are due on Patch Tuesday: https://t.co/QLqnpulr0A

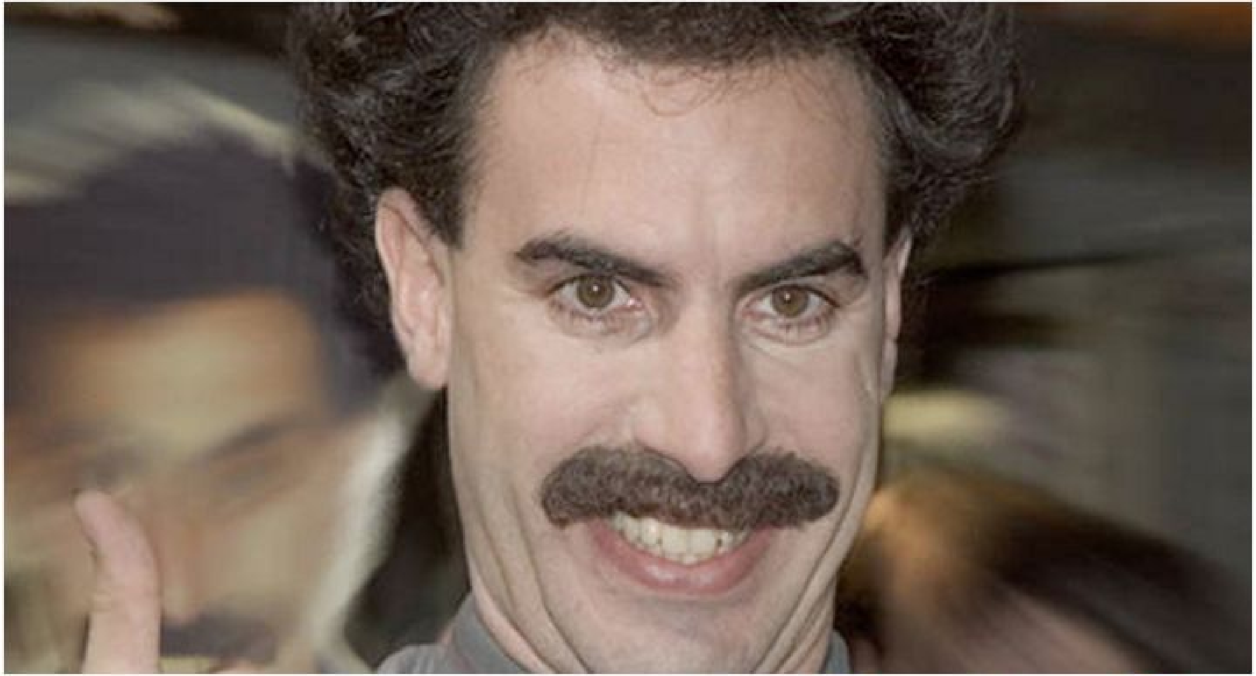
Welcome > Blog Home > Cryptography > D-Link Accidentally Leaks Private Code-Signing Keys

by Michael Mimoso Follow @mike_mimoso September 18, 2015 , 10:21 am

Security

Is Kazakhstan about to man-in-the-middle diddle all of its internet traffic with dodgy root certs?

Come on, guys. Don't go giving the Russians any ideas



3 Dec 2015 at 20:42, Shaun Nichols

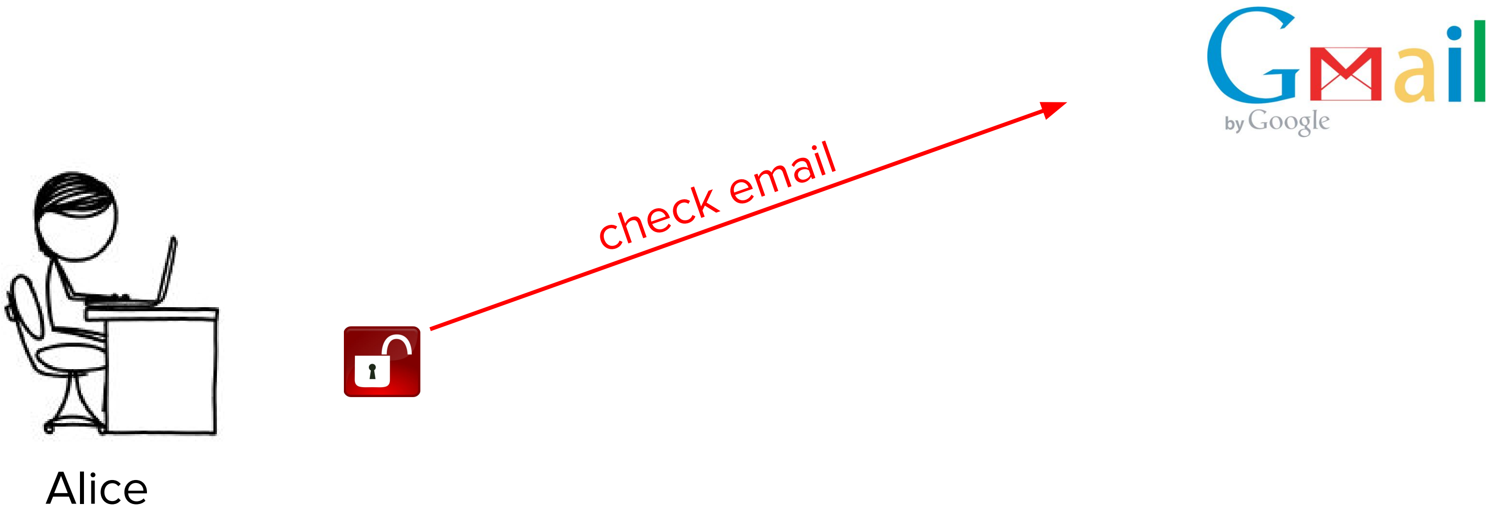
31 39

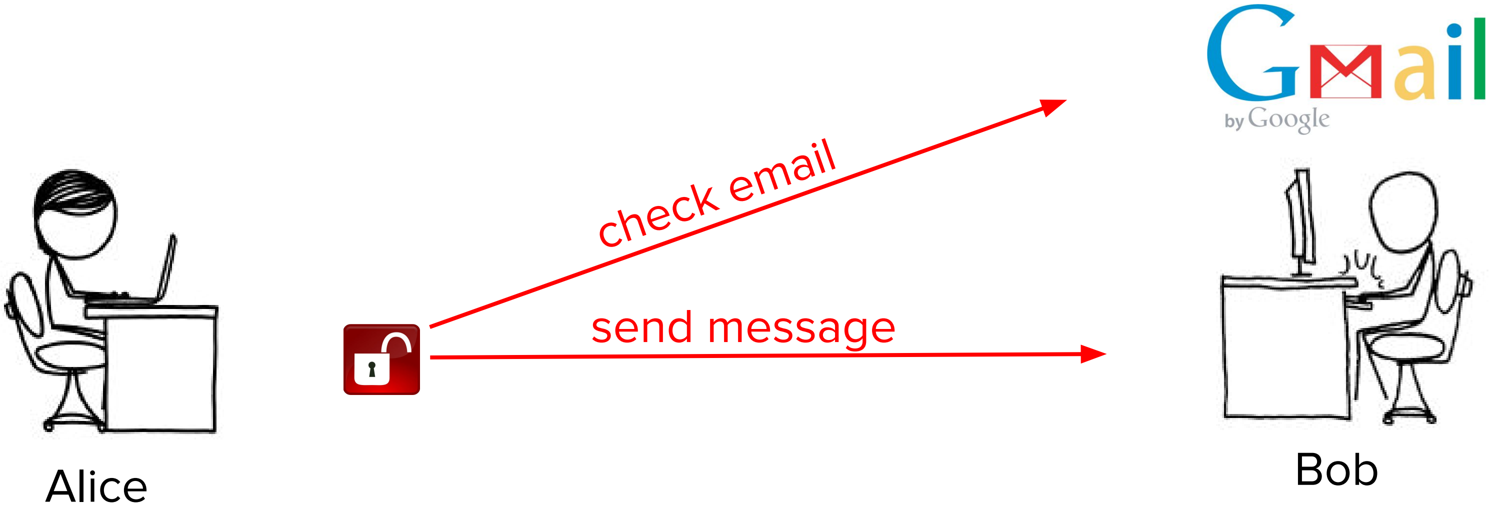
What do all of the previous incidents have in common?

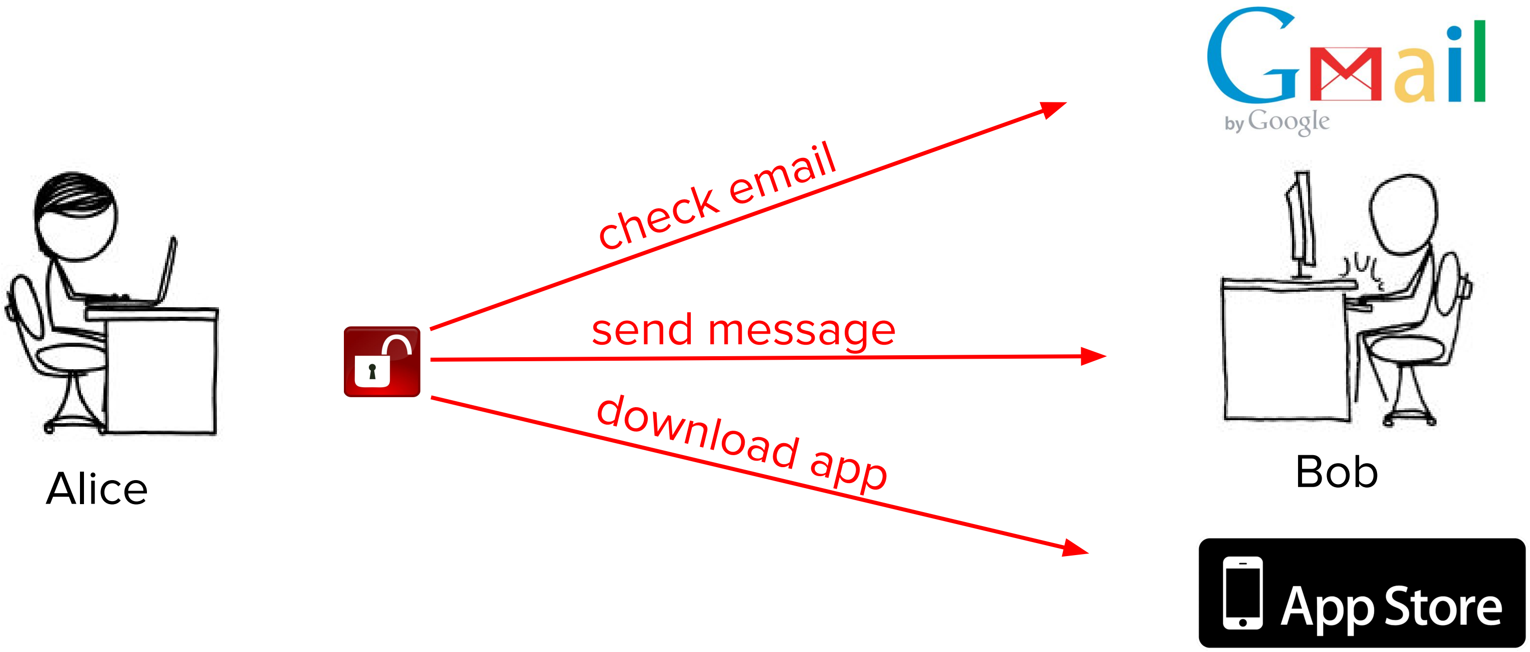
What do all of the previous incidents have in common?

Subverted authorities!

Why do we even have **authorities**?


check email
Gmail
by Google
Alice


Gmail
by Google
check email
send message
Alice
Bob


Gmail
by Google
check email
send message
download app
Alice
Bob
App Store

What is:

- Gmail's TLS public key?
- Bob's IM public key?
- App Store's public key?

Respect my Authoritah!

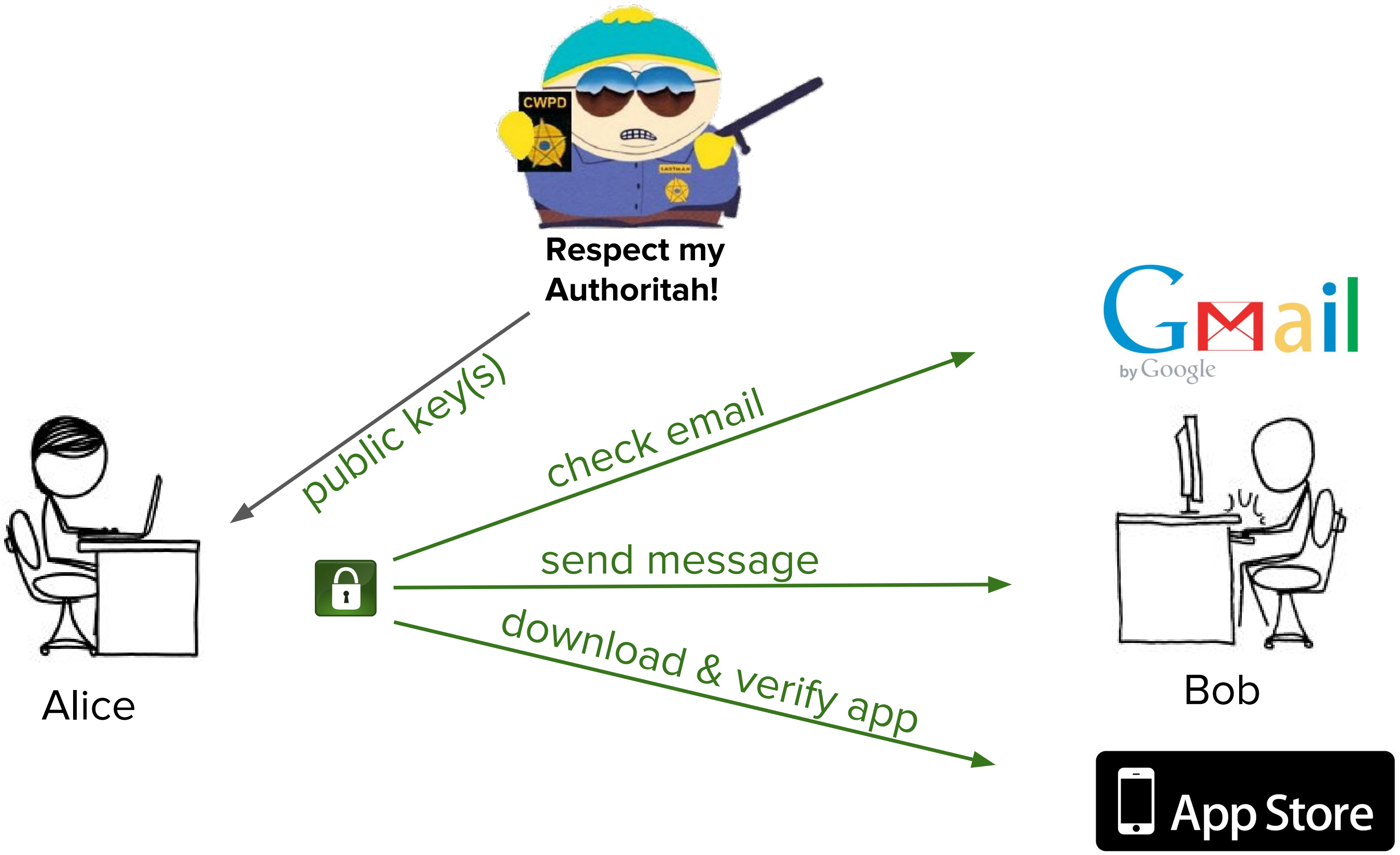
request

Alice

Gmail by Google

Bob

App Store



CWPD
Respect my
Authoritah!
public key(s)
check email
send message
download & verify app
Alice
Bob
Gmail
by Google
App Store

We **often** rely on authorities ...

Logging & Time-stamping Services, Digital Notaries

arXiv.org

eNotary

e-timestamp

SURETY

Logging & Time-stamping Services, Digital Notaries

arXiv.org

eNotary

e-timestamp

SURETY

Certificate Authorities

VeriSign

Go Daddy

COMODO Creating Trust Online

thawte it's a trust thing

Logging & Time-stamping Services, Digital Notaries

arXiv.org

eNotary

e-timestamp

SURETY

Certificate Authorities

VeriSign

Go Daddy

COMODO Creating Trust Online

thawte it's a trust thing

Naming Authorities

ICANN

SECURE64

DNSSEC SECURED by Afilias

Logging & Time-stamping Services, Digital Notaries

Certificate Authorities

Naming Authorities

Software Update Services

Logging & Time-stamping Services, Digital Notaries

Certificate Authorities

Naming Authorities

Software Update Services

Randomness Authorities

Logging & Time-stamping Services, Digital Notaries

arXiv.org
eNotary
e-timestamp
SURETY

Certificate Authorities

VeriSign
Go Daddy
COMODO Creating Trust Online
thawte it's a trust thing

Naming Authorities

ICANN
SECURE64
DNSSEC SECURED by Afilias

Software Update Services

GET IT ON Google play

Randomness Authorities

... but are authorities **trustworthy**?

Authorities going bad

Respect my Authoritah!

Gmail by Google

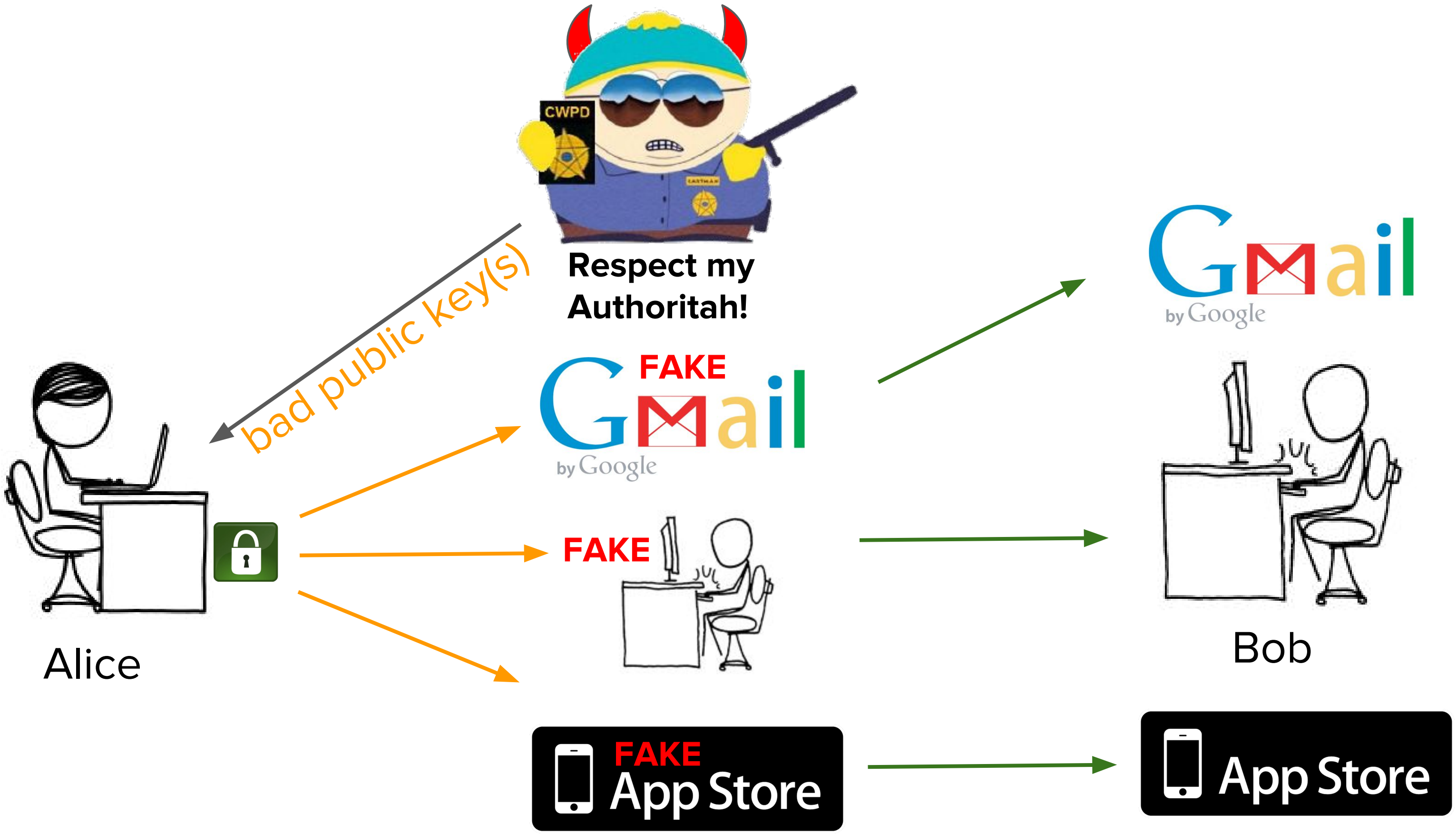
Alice

Bob

App Store

Authorities going bad

Authorities going bad



Problems

1) Authorities are **powerful** and **wide-spread**

Examples:

- Any CA can issue certs for arbitrary domains
- Hundreds of CAs trusted by web browsers

Problems

2) **Things go bad everywhere, all the time**

Examples:

- Insider attacks
- Private key thefts
- Human error
- Hacking
- Compulsory key handover
- Side-channel attacks

Problems

3) **Weakest-link security**: authority systems are very fragile

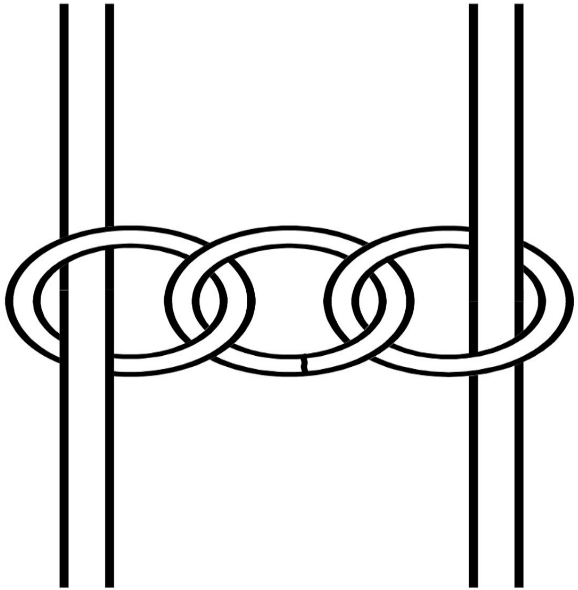
Examples:

- Adversary (e.g. hacker, spy agency) needs only **one** CA key to subvert entire system

What if we could **decentralise** authority services?

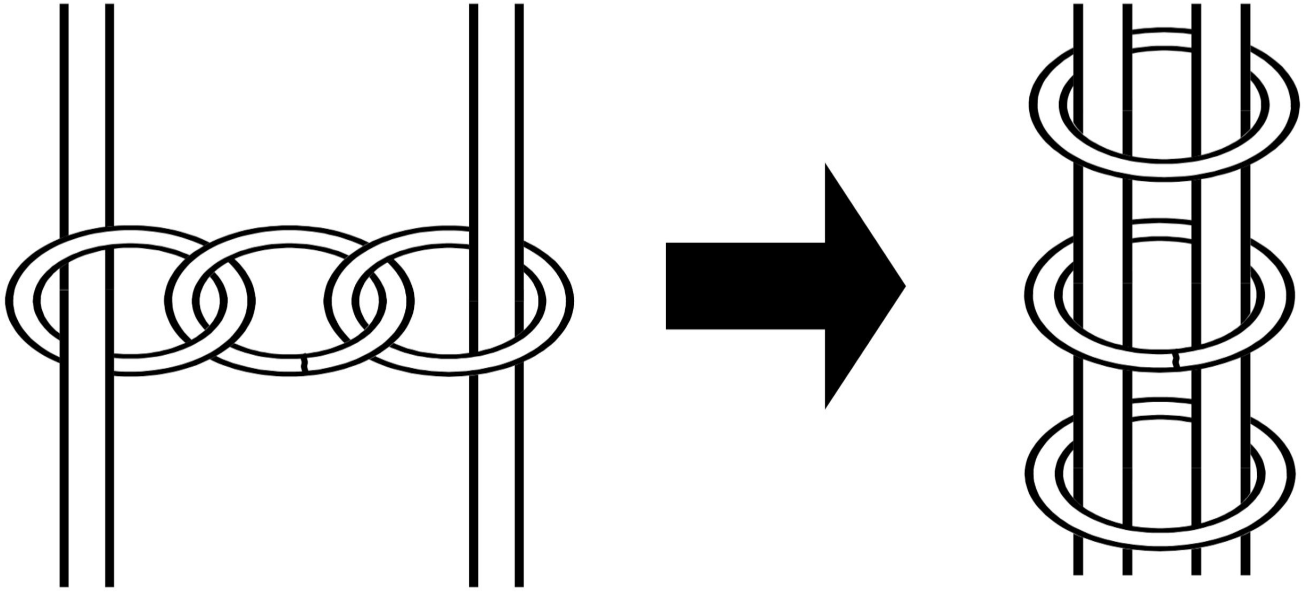
Decentralising Authorities

from **weakest-link**



Decentralising Authorities

from **weakest-link** to **strongest-link** security



Decentralising Authorities

There are already many tools available:

- “Anytrust”: 1-of-k servers honest, all k live
- Byzantine replication: ⅔ honest, ⅔ live
- Threshold cryptography
- Multi-signature schemes

Decentralising Authorities

Trust-splitting (so far):

- **Rare**
- **Challenging** to implement
- Usually **not scalable** to large groups

Decentralising Authorities

Trust-splitting (so far):

- **Rare**
- **Challenging** to implement
- Usually **not scalable** to large groups

But:

- Is splitting across 5-10 servers **enough** (e.g. against state-level adversaries)?
- Are participants truly **independent** and **diverse**?
- **Who** chooses the composition and **how**?

Cothorities

Large-scale collective authorities

Cothorities

Implement trust-splitting that is:

Scalable **Secure** **Robust** **Flexible**

Cothorities

Implement trust-splitting that is:

Scalable **Secure** **Robust** **Flexible**

First-step goal:

Generically improve security of any authority independent of type or semantics.

Witness Cothorities

“Who watches the watchers?”

“Public witnesses!”

Witness Cothorities

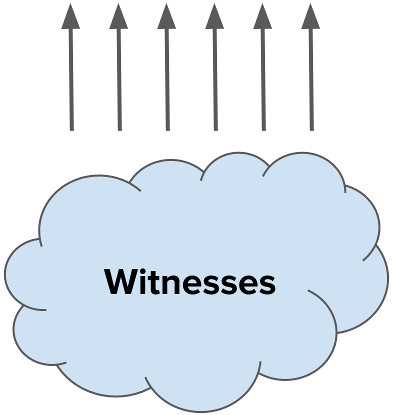
“Who watches the watchers?”

“Public witnesses!”

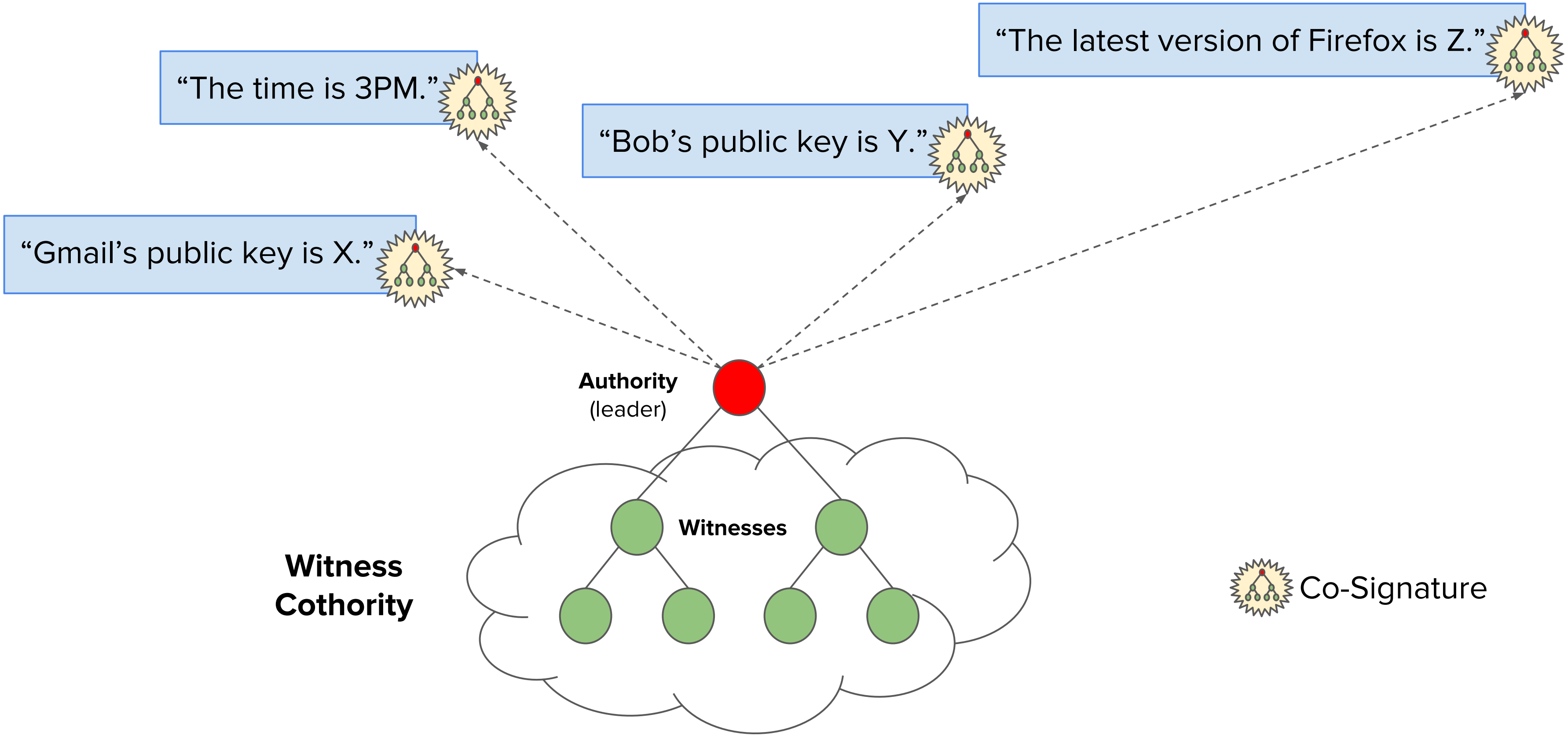
CoSi: Collective Signing Protocol

- **Authority:** generate statements
- **Witnesses:**
    - collective & proactive sanity-check
    - contribute to collective signature

Respect my Authoritah!



CoSi: Collective Signing



CoSi: Design

Builds on well-known crypto primitives:

- Merkle Trees
- Schnorr (Multi-)Signatures

CoSi: Design

Builds on well-known crypto primitives:

- Merkle Trees
- Schnorr (Multi-)Signatures

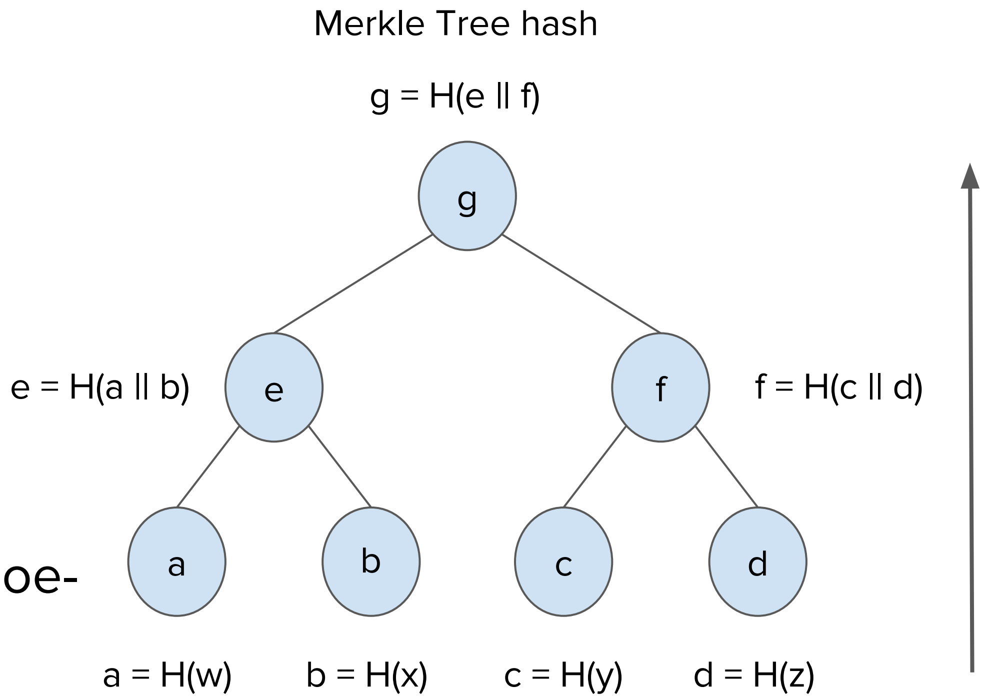
Scalability (to thousands of nodes) through:

- Communication trees
- Aggregation

E.g. as in scalable multicast protocols

Merkle Trees

- hash trees
- verification of large data structures in O(log n)
- signed top hash (STH): efficient authentication
- used in many projects: Git, ZFS, BitTorrent, Bitcoin, Certificate Transparency, Tahoe-LAFS, etc.



Schnorr (Multi-)Signatures

	Signer 1	**Signer 2**	**Verifier**
Private/Public keys	k_1, $K_1 = g^{k1}$	k_2, $K_2 = g^{k2}$	

Schnorr (Multi-)Signatures

	Signer 1	Signer 2		Verifier
Private/Public keys	$k_1, K_1 = g^{k1}$	$k_2, K_2 = g^{k2}$		
1. Commitment	$v_1, \mathbf{V_1} = g^{v1}$	$v_2, \mathbf{V_2} = g^{v2}$	→	$V = \mathbf{V_1} * \mathbf{V_2}$

Signing

Schnorr (Multi-)Signatures

	Signer 1	Signer 2		Verifier
Private/Public keys	$k_1, K_1 = g^{k1}$	$k_2, K_2 = g^{k2}$		
1. Commitment	$v_1, V_1 = g^{v1}$	$v_2, V_2 = g^{v2}$	→	$\mathbf{V} = V_1 * V_2$
2. Challenge	c	c	←	$c = H(M \| \mathbf{V})$

Signing

Schnorr (Multi-)Signatures

		Signer 1	**Signer 2**		**Verifier**
	Private/Public keys	k_1, $K_1 = g^{k1}$	k_2, $K_2 = g^{k2}$		
Signing	1. Commitment	$v_1, V_1 = g^{v1}$	$v_2, V_2 = g^{v2}$	→	$V = V_1 * V_2$
	2. Challenge	c	c	←	$c = H(M \| V)$
	3. Response	$r_1 = v_1 - k_1 c$	$r_2 = v_2 - k_2 c$	→	$r = r_1 + r_2$

Schnorr (Multi-)Signatures

	Signer 1	Signer 2		Verifier
Private/Public keys	$k_1, K_1 = g^{k1}$	$k_2, K_2 = g^{k2}$		
Signing 1. Commitment	$v_1, V_1 = g^{v1}$	$v_2, V_2 = g^{v2}$	→	$V = V_1 * V_2$
2. Challenge	c	c	←	$c = H(M \Vert V)$
3. Response	$r_1 = v_1 - k_1 c$	$r_2 = v_2 - k_2 c$	→	$r = r_1 + r_2$
Signature on M				(c, r)

Schnorr (Multi-)Signatures

	Signer 1	Signer 2	Verifier
Private/Public keys	k_1, $K_1 = g^{k1}$	k_2, $K_2 = g^{k2}$	
Signature on M			(c,r)
1. Commitment recovery	$K = K_1 * K_2$	$V' = g^r K^c$	

Verification

Schnorr (Multi-)Signatures

	Signer 1	Signer 2	Verifier
Private/Public keys	k_1, $K_1 = g^{k1}$	k_2, $K_2 = g^{k2}$	
Signature on M			(c,r)

Verification

1. Commitment recovery	$K = K_1 * K_2$	**V'** $= g^r K^c$
2. Challenge recovery	c' = H(M || **V'**)	

Schnorr (Multi-)Signatures

	Signer 1	Signer 2	Verifier
Private/Public keys	$k_1, K_1 = g^{k1}$	$k_2, K_2 = g^{k2}$	
Signature on M			(c, r)

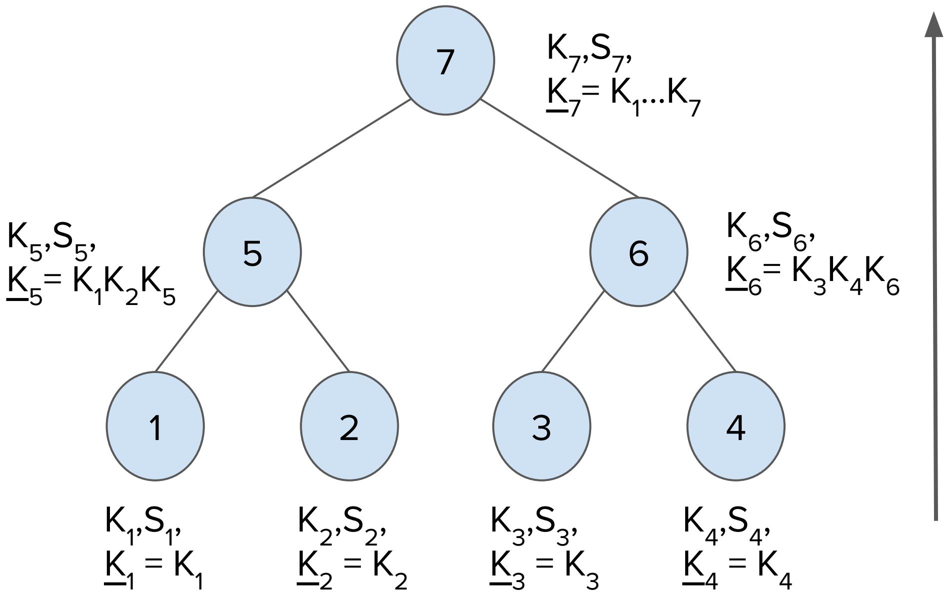
Verification

1. Commitment recovery	$K = K_1 * K_2$	$V' = g^r K^c$
2. Challenge recovery	$c' = H(M \parallel V')$	
3. Decision	$c \stackrel{?}{=} c'$	

CoSi: Setup

Merkle Tree containing:

- Public key K_i
- Self-signed certificate S_i (using secret key k_i)
- Aggregate public keys $\underline{K}_i$

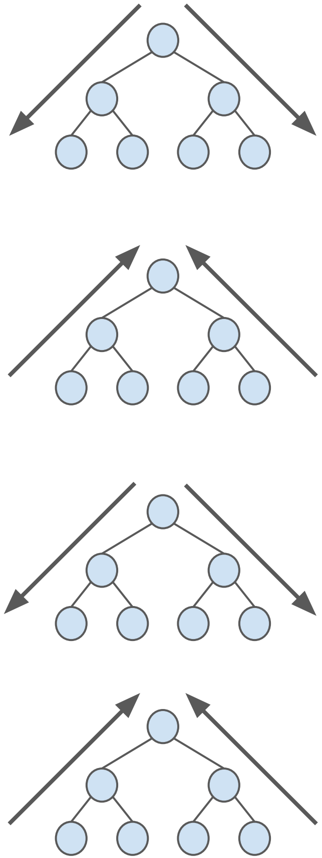


One-time verification costs: O(n)

On group change: O(|m-n|)

CoSi: Round

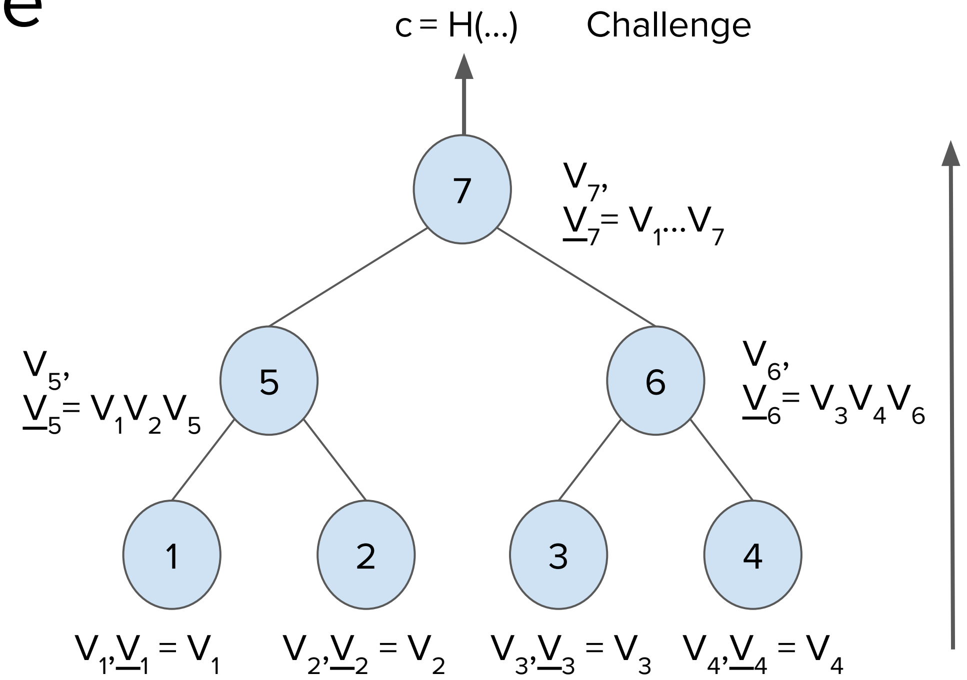
1. Announcement Phase
2. Commitment Phase
3. Challenge Phase
4. Response Phase



CoSi: Commitment Phase

Merkle Tree containing:

- Commits $V_i = g^{vi}$
- Aggregate commits $\underline{V}_i$



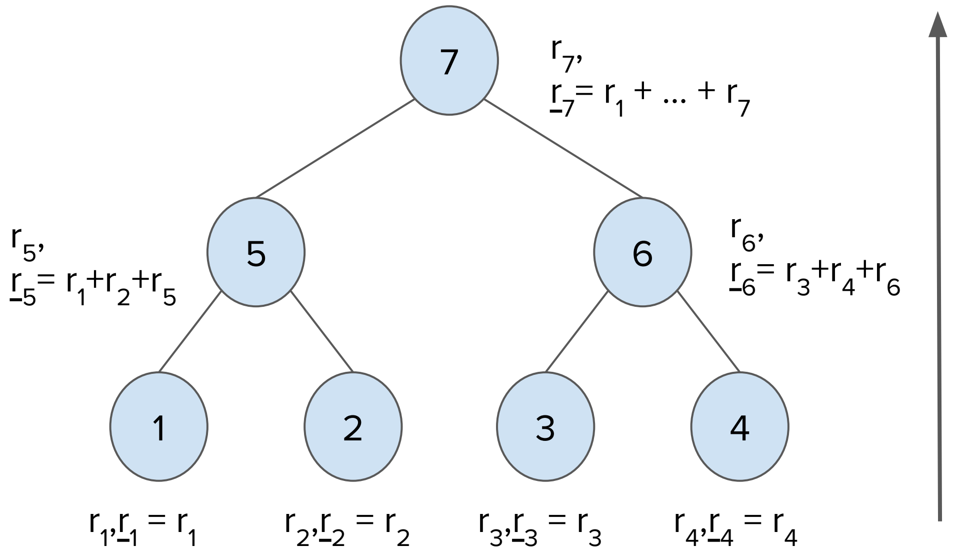
Output:

- root hash = collective challenge c

CoSi: Response Phase

Compute:

- Response $r_i = v_i - k_i c$
- Aggregate response $\underline{r}_i$



Outputs:

- Valid partial signatures (c, r_i)
- Complete signature (c, r_7)

The Availability Problem

- Assumption: server failures **rare** but **non-negligible**
- Availability loss
- DoS vulnerability if not addressed
- Persistently bad servers administratively handled

The Availability Problem

- Assumption: server failures **rare** but **non-negligible**
- Availability loss
- DoS vulnerability if not addressed
- Persistently bad servers administratively handled

Solutions: (work-in-progress)

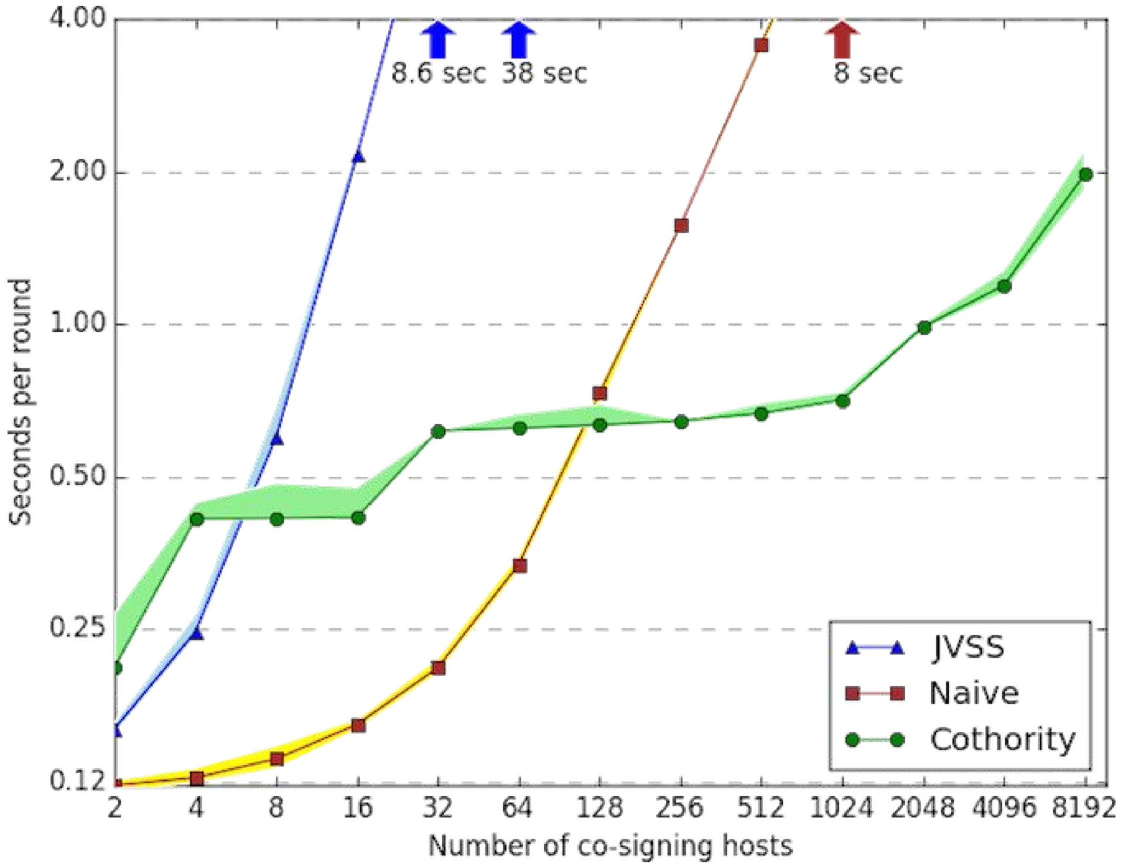
- Exceptions (remove failing node from co-signing, notify client)
- Life insurance (based on VSS)

Cothority Implementation

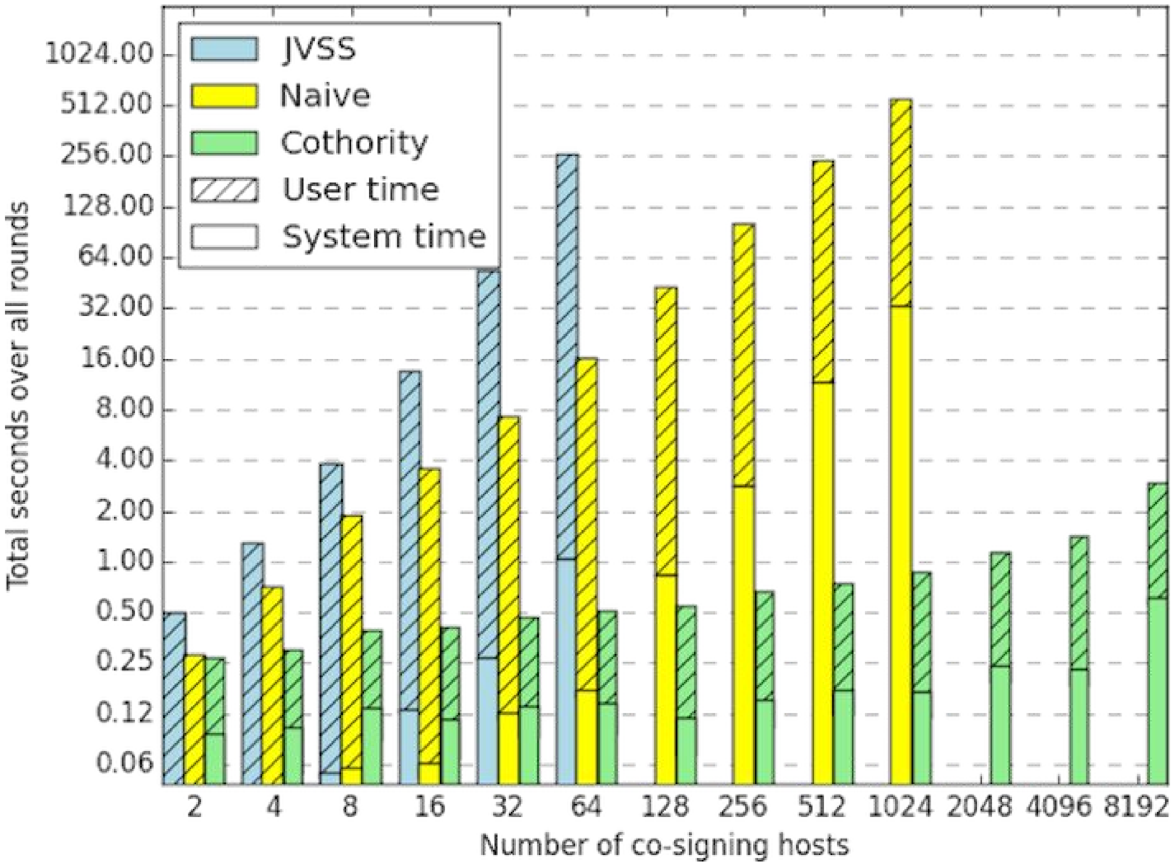
Implementation

- Implemented in Go:
  - Cothority prototype: https://github.com/dedis/cothority
  - Crypto library: https://github.com/dedis/crypto
- Schnorr multi-signatures based on Ed25519:
  - AGL's Go port of DJB's optimised code
- Experiments on DeterLab
  - Up to 8192 virtual CoSi nodes
  - Multiplexed on top of up to 32 physical machines
  - Latency: 100ms round-trip between two servers

Experimental Results: Collective Signing Time



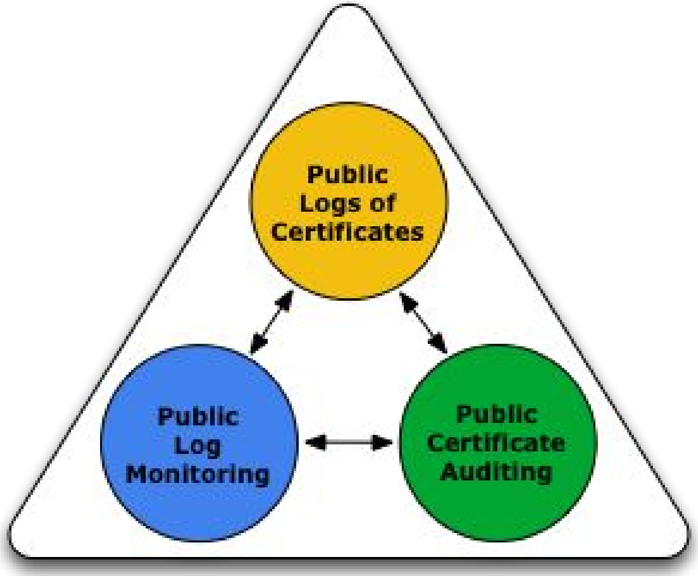
Experimental Results: Computation Costs

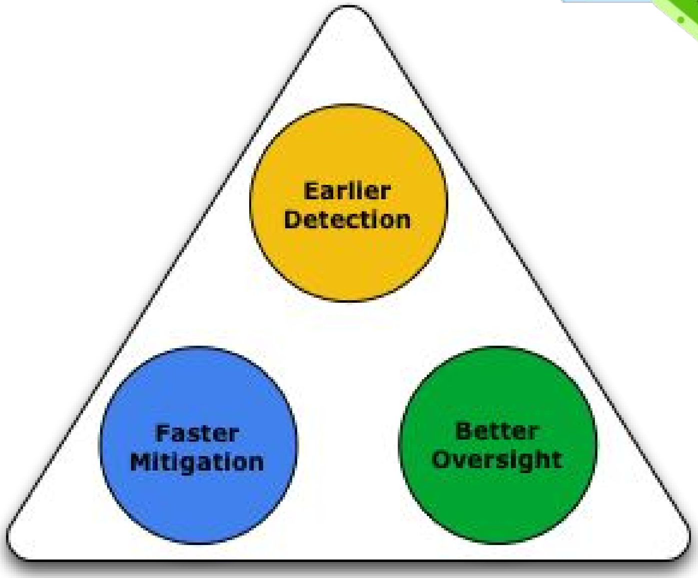


Cothority Applications

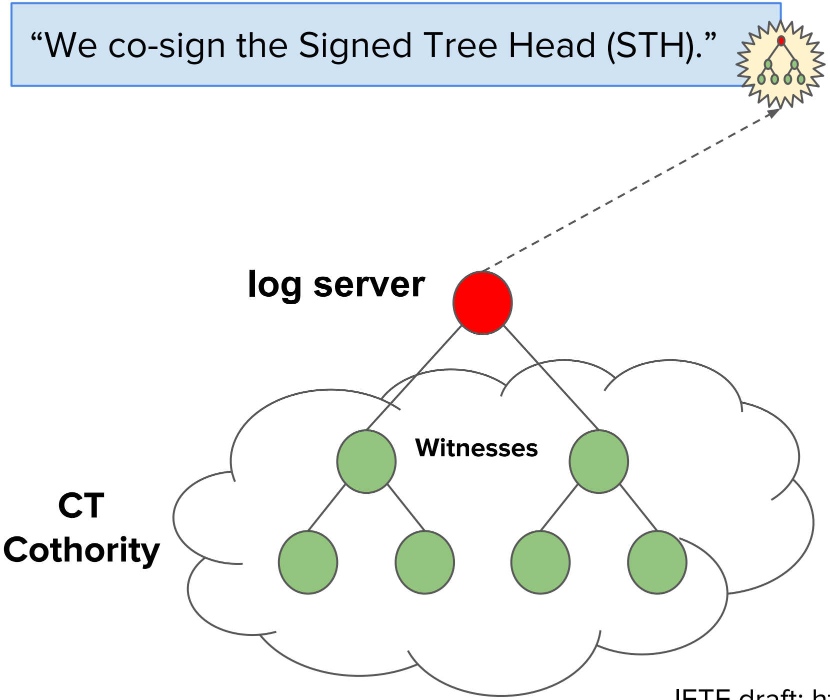
Let’s fix the Internet! :-)

Certificate Transparency



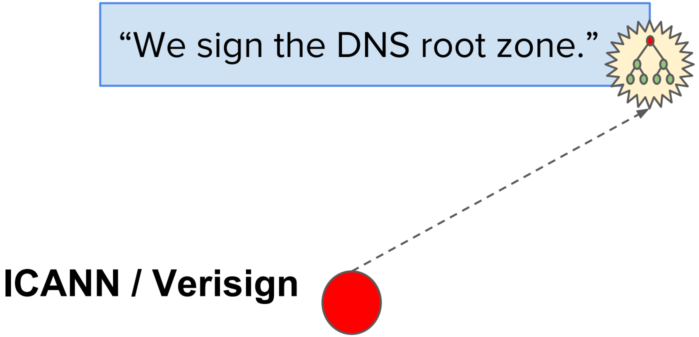


Certificate Transparency

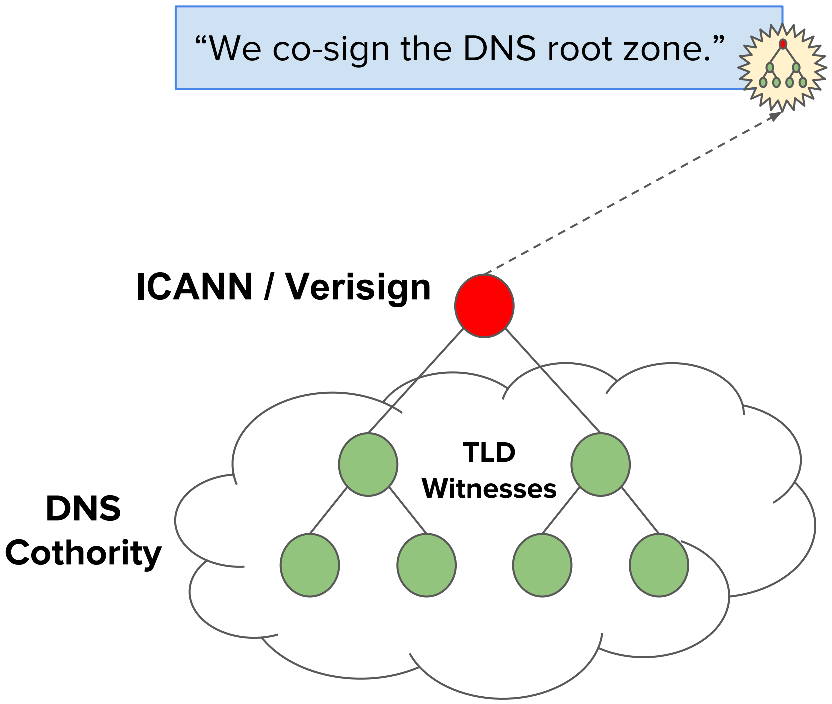


IETF draft: https://datatracker.ietf.org/doc/draft-ford-trans-witness/

DNSSEC



DNSSEC



“We co-sign the DNS root zone.”
ICANN / Verisign
TLD
Witnesses
DNS
Cothority

Software Distribution

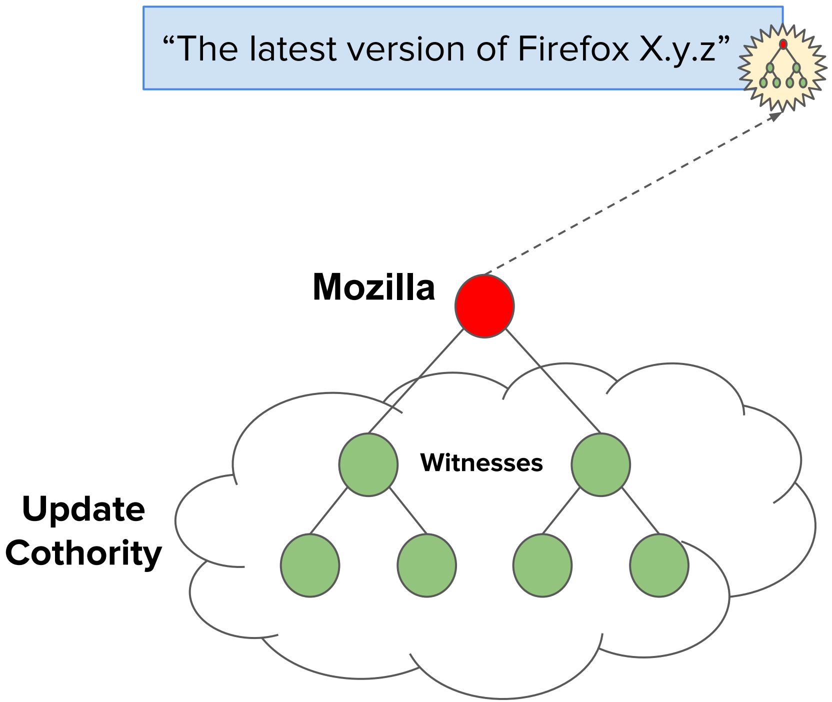
App Store

GET IT ON Google play

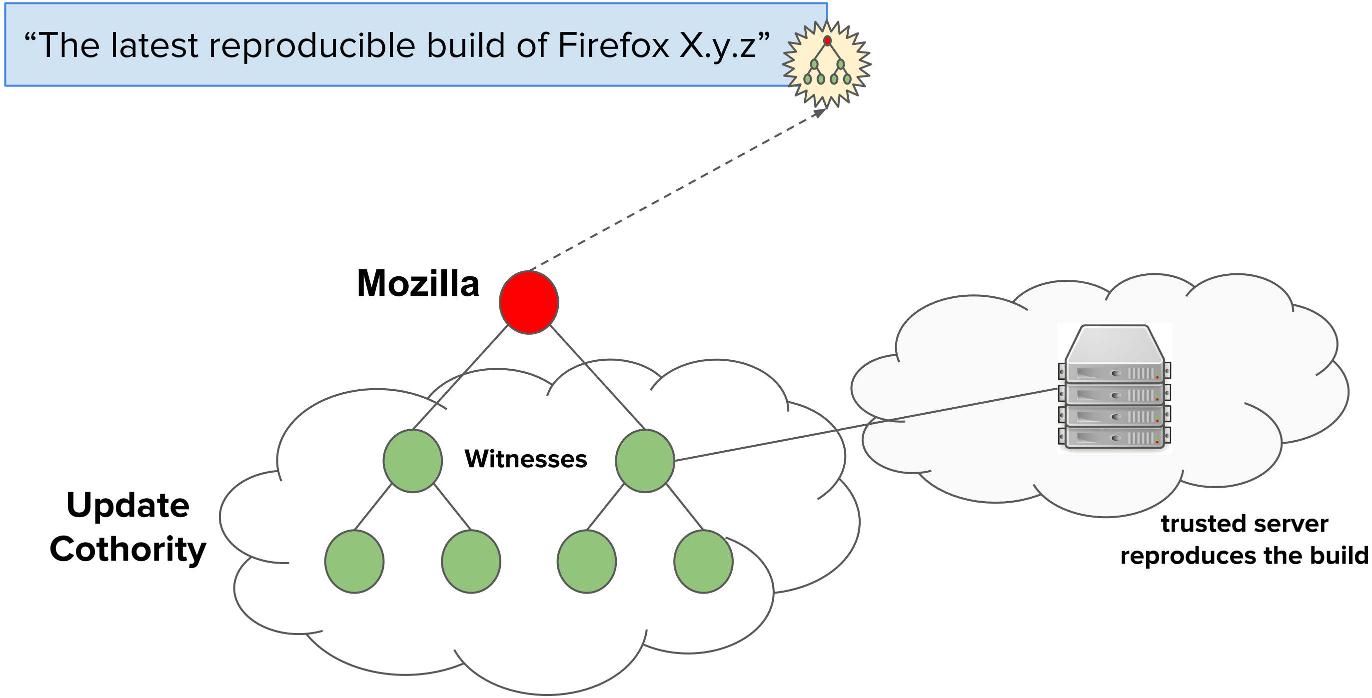
reproducible-builds.org

archlinux

Software Distribution



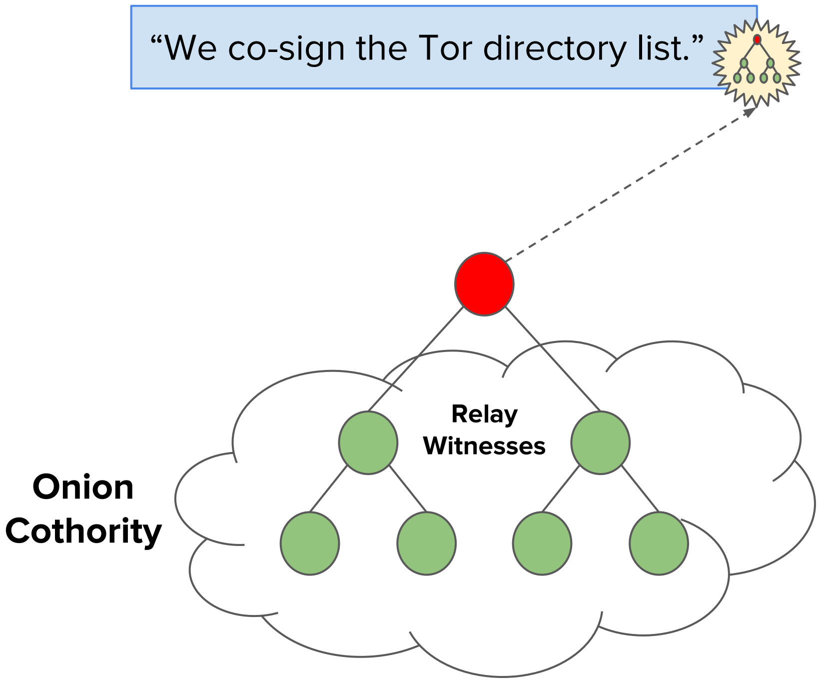
Reproducible Builds



Tor

picture credit: http://jordan-wright.com/

Tor



"We co-sign the Tor directory list."
Relay
Witnesses
Onion
Cothority

Cryptocurrencies

allcoinsnews

Bitcoin, Altcoin & Blockchain News

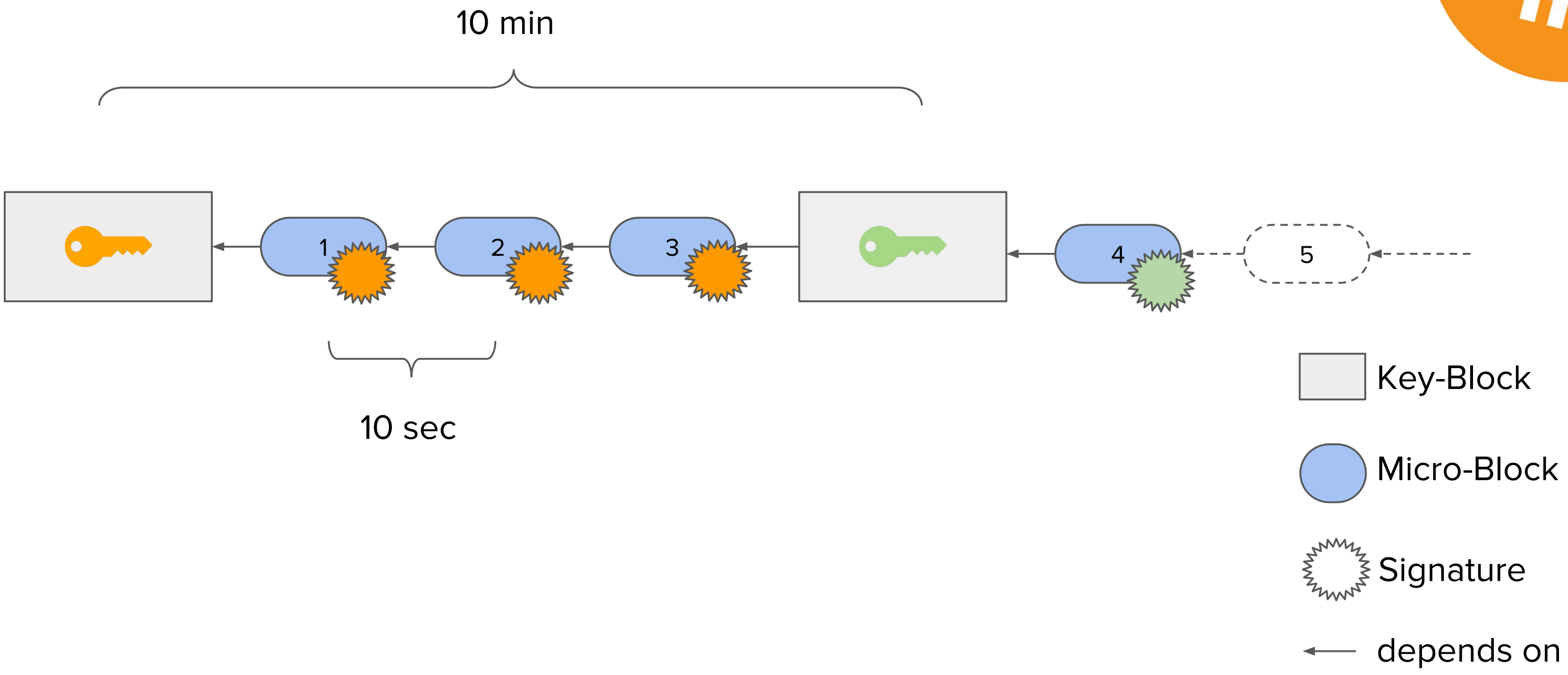
Home » Bitcoin & Blog »

Blocksize Debate Rages while Bitcoin-NG Addresses Bitcoin Scalability Issues

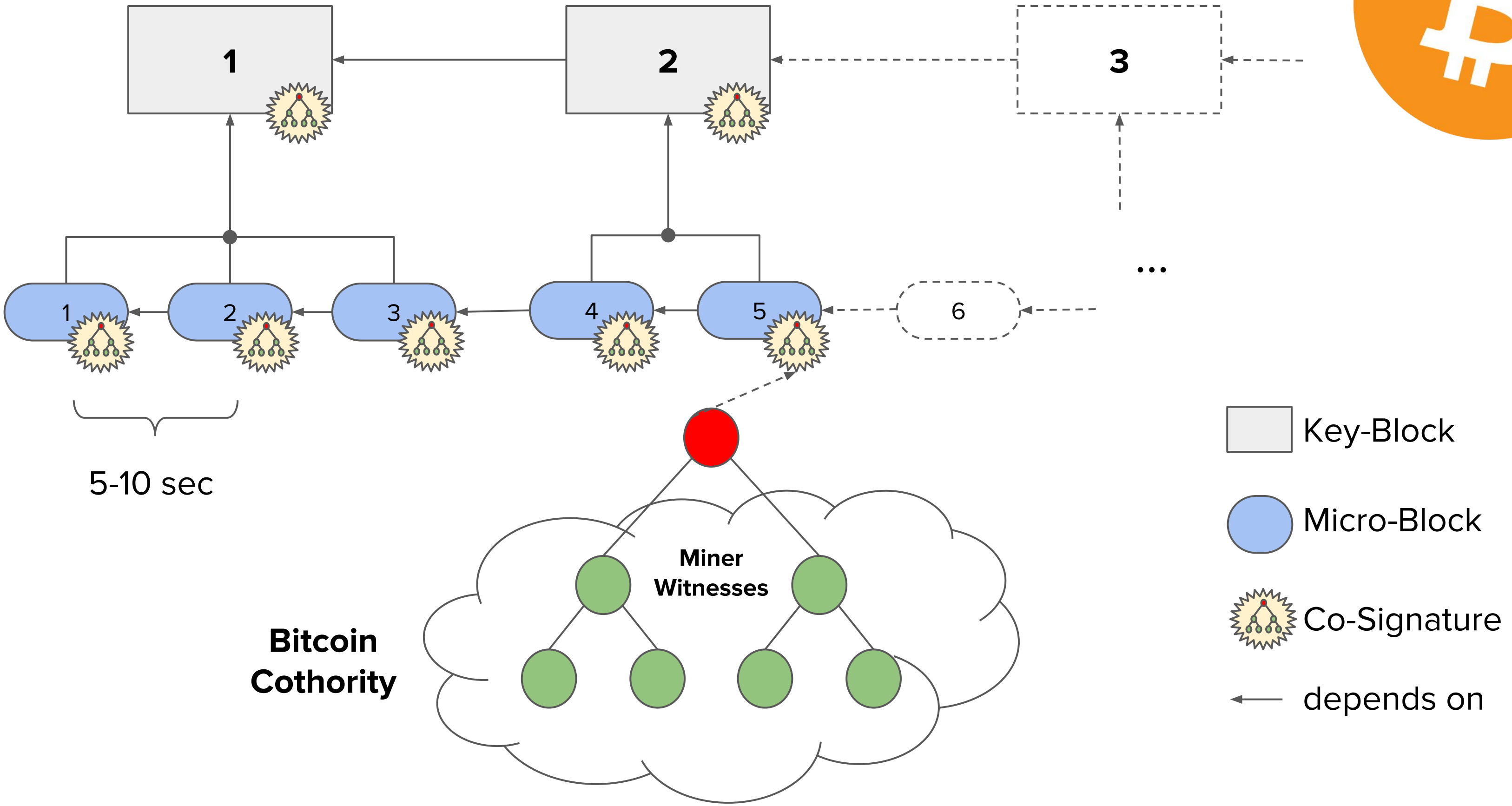
Blocksize Debate Rages while Bitcoin-NG Addresses Bitcoin Scalability Issues

Hans Lombardo November 11, 2015

Cryptocurrencies – Bitcoin-NG



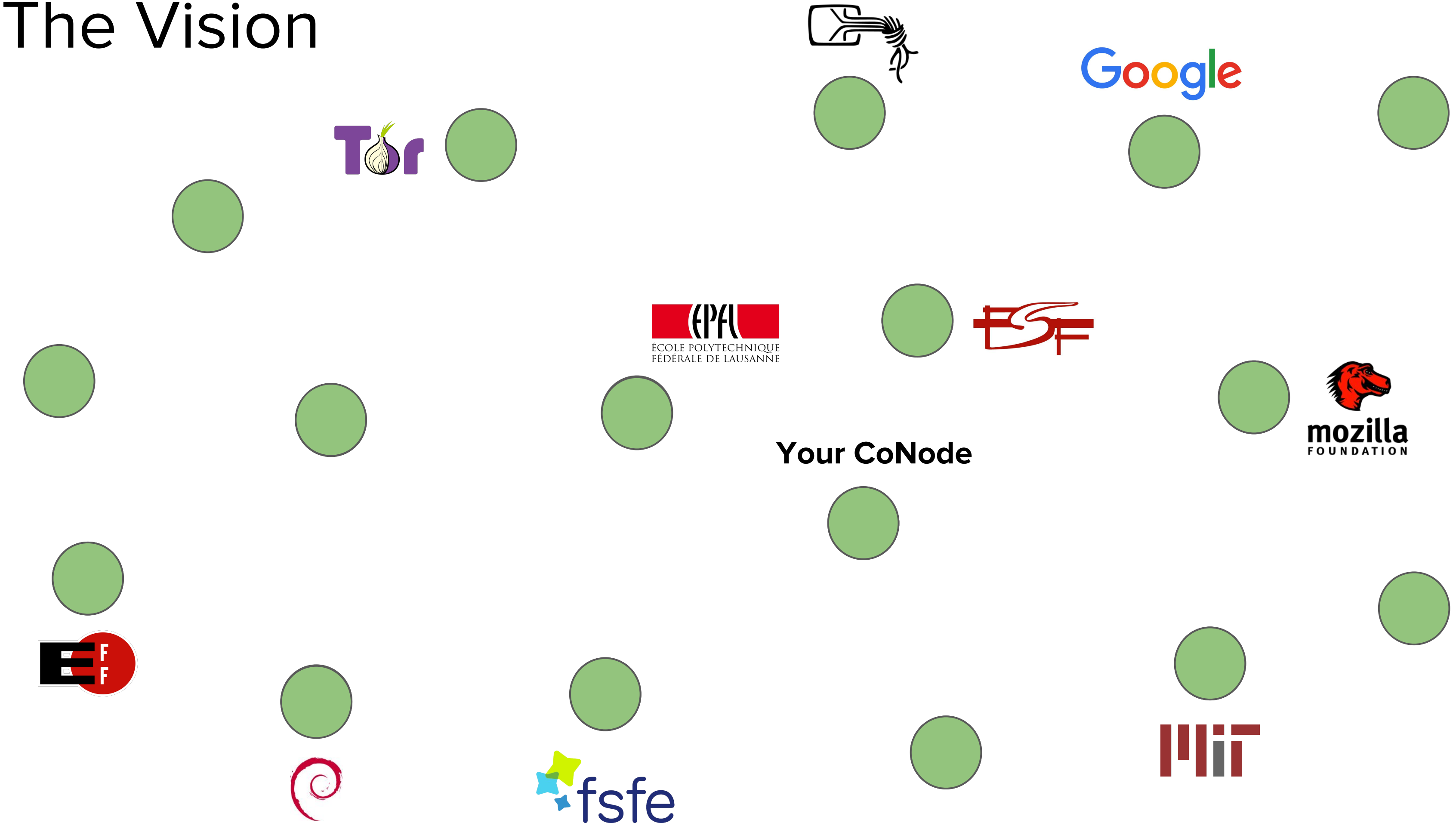
Cryptocurrencies – BitCoSi



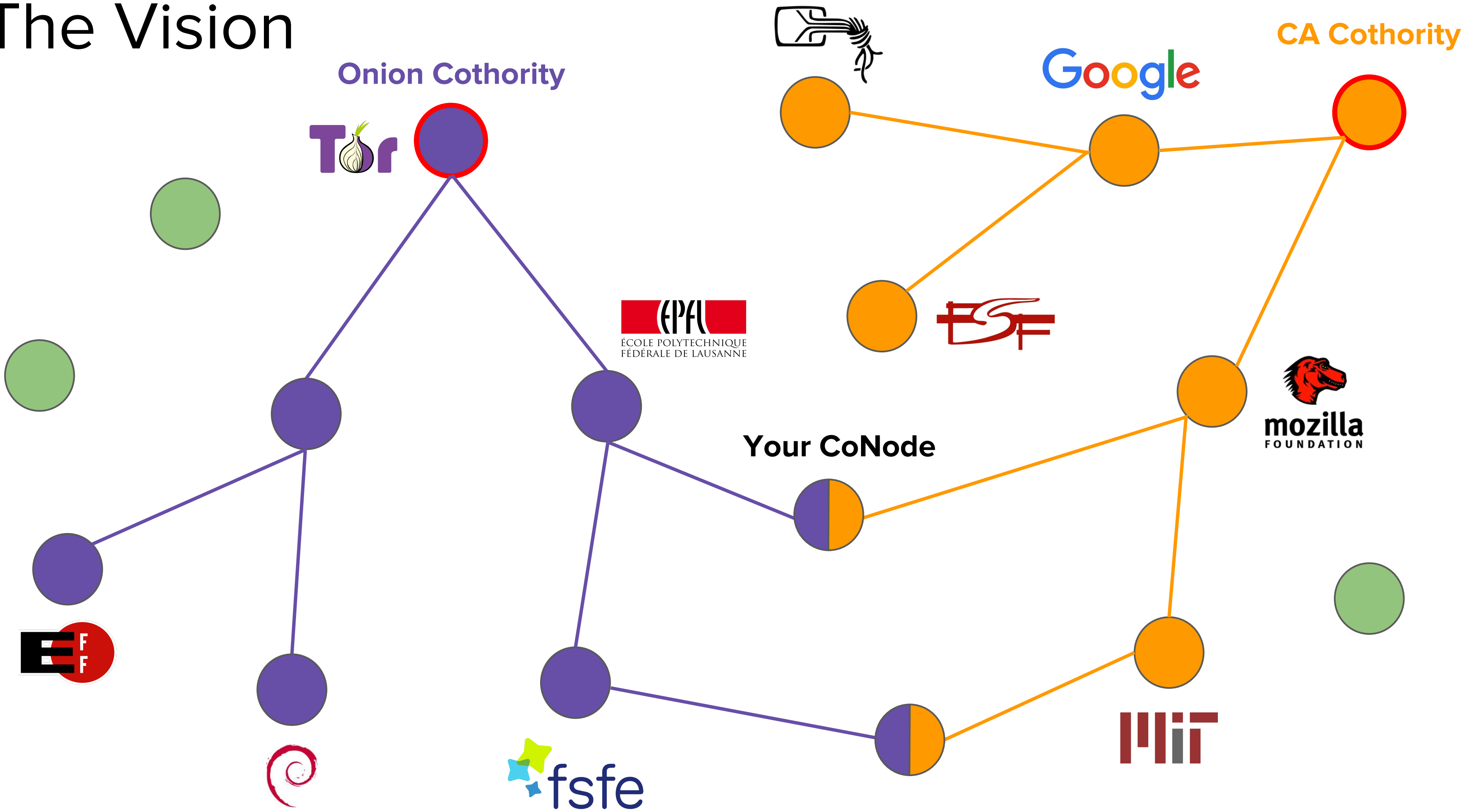
… and many more applications …

(public randomness, git, … stay tuned!)

The Vision


Tor
Google
EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Your CoNode
mozilla
FOUNDATION
EFF
fsfe

The Vision


Onion Cothority
CA Cothority
Google
Tor
EPFL
ÉCOLE POLYTECHNIQUE
FÉDÉRALE DE LAUSANNE
Your CoNode
mozilla
FOUNDATION
EFF
fsfe

Setup Your CoNode, Join the EPFL-Cothority!

```
$ curl https://api.github.com/repos/dedis/cothority/releases/latest \
| grep '"browser_download_url":' | awk -F\" '{ system("curl -L " $4) }' > conode-latest.tar.gz
$ tar -xvf conode-latest.tar.gz
$ ./start-conode.sh setup <ip-address>:<port>
```

Send the generated public key `key.pub` to

https://groups.google.com/forum/#!forum/cothority

and wait until we have verified your CoNode.

```
$ ./stamp sign <file>     # co-sign <file> through the EPFL-cothority
$ ./stamp check <file>    # verify the signature of <file>
```

Conclusion

Cothorities build on well-known ideas:

- Distributed/Byzantine consensus
- Merkle Trees
- Threshold crypto
- Multi-signature schemes

But demonstrate how to do trust-splitting at scale:

- Strongest-link security
- Practical: demonstrated for 8000+ participants
- Efficient: < 2 seconds signing latency at scale

Thank you!

Don’t forget to check out:

http://arxiv.org/abs/1503.08768 (paper)

https://github.com/dedis/cothority (code)

https://groups.google.com/forum/#!forum/cothority (mailing list)