MIT 6.566

FOKS - The Federated Open Key Service

Max Krohn / https://foks.pub / 2026.04.23

Migration from “On-Prem” to Cloud
2006-2026

- 2003-2008: MIT Athena, MIT LCS and later MIT CSAIL had dedicated “server rooms”
- Today: Mostly Cloud-based (AWS, GCP, Azure, etc)
- Applications: have stayed largely the same
  - {IRC,Zephyr} vs {Slack,Signal}, Email vs Gmail, AFS vs Dropbox, RCS vs Git
  - Scale has ~1000x’ed

Consequences of Cloud Migration

- Simplified management for all but the professionals
- Changed the cost-structure (own vs rent)
- Concurrent with proliferation of WiFi+NATs, better access to shared resources
- Totally upended the security picture

The Cloud: A Data Privacy “Challenge”

- Most cloud services store data in plaintext or encrypted with the key essentially sitting next to the data
- Susceptible to attacks at all layers of the stack
- Compelled to comply with lawful discovery
- All data is essentially **undeletable**

Quick Sidebar: The 3rd Party Doctrine

- The 4th Amendment says probable cause and warrants required for search and seizure of homes and property
- Founding fathers did not anticipate cloud computing
- Until 2018, users of cloud had “no reasonable expectation of privacy”
- Recent challenges and court cases this year; state-level legislation, etc.
- My uninformed view: this stuff is by no means settled
- But, “the cloud is forever”
- ∴ even deleted cloud data is at risk with further legal and political volatility

How To Safely Use the Cloud?

The challenge

- Clearly the cloud has some advantages
- Can we engineer a system to give us the best of both worlds:
  - Cloud-style convenience, pricing, uptime, reliability
  - Homelab-style data privacy

Secure Cloud Systems

A brief history (Research + Commercial)

- Messaging and Email:
  - PGP - Pretty Good Privacy - E2E-Encrypted Email (1990s)
  - “Off The Record” (2004) →Signal / WhatsApp
- Files:
  - 1990s-2000s: Cephus, Plutus, Oceanstore, SFS, SUNDR
  - 2010-present: Tahoe-LAFS, Tarsnap, Keybase
- Show of hands: who uses…..

Why is this still unsolved?

The threat was anticipated >20 years ago

- Key management is hard
- Security decisions affect the UI/UX
- For a system to work and win, it has to be as usable and feature-ful as the things people currently use that aren’t encrypted
  - WhatsApp / Signal have crossed the threshold
  - Maybe Keybase did, debatable….

Hi max!

People
Chat 211
Files
Crypto
Teams
Wallet
Git
Devices
Settings

Invite friends
48

Search (⌘K)
New chat

mikem 9:45 AM
You: https://www.nyti...

keybase.vine 9:42 AM
chris: :-)

pzduniak 9:22 AM
pzduniak: ok then

sckrohn 8:56 AM
Max Krohn Engagem...

samyagan, chris 8:32 AM
samyagan: One minute

chris 8:09 AM
You: i think this one is...

+4074 more

agorise
general
chia_network.public
general
eos_ghostbusters
general
facultybase
general
fauxbar
foo
general
fosssecurityteam

211 unread messages

Mike Maxim

mikem • Software Engineer at Keybase 🔐, former CTO at OkCupid ❤️

3d
```
[max@MacBook-Pro ~/Library/Application Support ]
ls -lsa Google
ls: Google: No such file or directory
[max@MacBook-Pro ~/Library/Application Support ]
mkdir Google
mkdir: .: No such file or directory
[max@MacBook-Pro ~/Library/Application Support ]
mkdir Schmoogle
[max@MacBook-Pro ~/Library/Application Support ]
```

mikem 9:00 AM
hah weird

max 9:01 AM
it all started chrome would start up and rev up to 200% cpu

so then i tried to delete all chrome files and reinstall

and then got stuck in this amazing pickle

max 9:45 AM
https://www.nytimes.com/interactive/2020/05/06/travel/coronavirus-travel-questions.html?action=click&module=Top%20Stories&pgtype=Homepage

nytimes.com • Published Today 5:00 AM

The Future of Travel

Perhaps no industry has been as hard hit by the pandemic as tourism. As restrictions on companies and travelers ease, what will the new world look like?

1 1 1

3d Write an exploding message

bold, _italics_, `code`, >quote, @user, @team, #channel

Turn into a team

Add and delete members as you wish.

Members Attachments Bots Settings

Add a bot

In this conversation

Google Meet • by keybase
Start Google Meet video calls from...

Urban Dictionary • by haukened
Lookup phrases on Urban Dictionary

Polling Service • by keybase
Public and anonymous polls in chat

Bartender • by mikem
Have fun with drink recipes, and...

Reminder Bot • by jessk
Never forget again.

Featured

Zoom Meetings • by keybase
Zoom video calls in chat

Google Calendar • by keybase
Google Calendar notifications...

Jitsi Meet • by haukened
Jitsi Meet video calls in chat

GitHub • by keybase
Bring your code to the...

GitLab • by keybase

Keybase

Related work

- Goal: End-to-End Encrypted Files, Git hosting, Slack-like chat
- User experience: multiple devices (phone, laptop, backup)
- Admin experience: tamper-proof teams of users
- Contributions: Apply Certificate-Transparency-like (lecture 14) techniques to the above
- Shortcomings: VC-backed business model pushed toward centralized server
  - Not a general protocol like TLS, HTTP, SMTP

FOKS - Federated Open Key Service

A new open-source protocol / Keybase successor

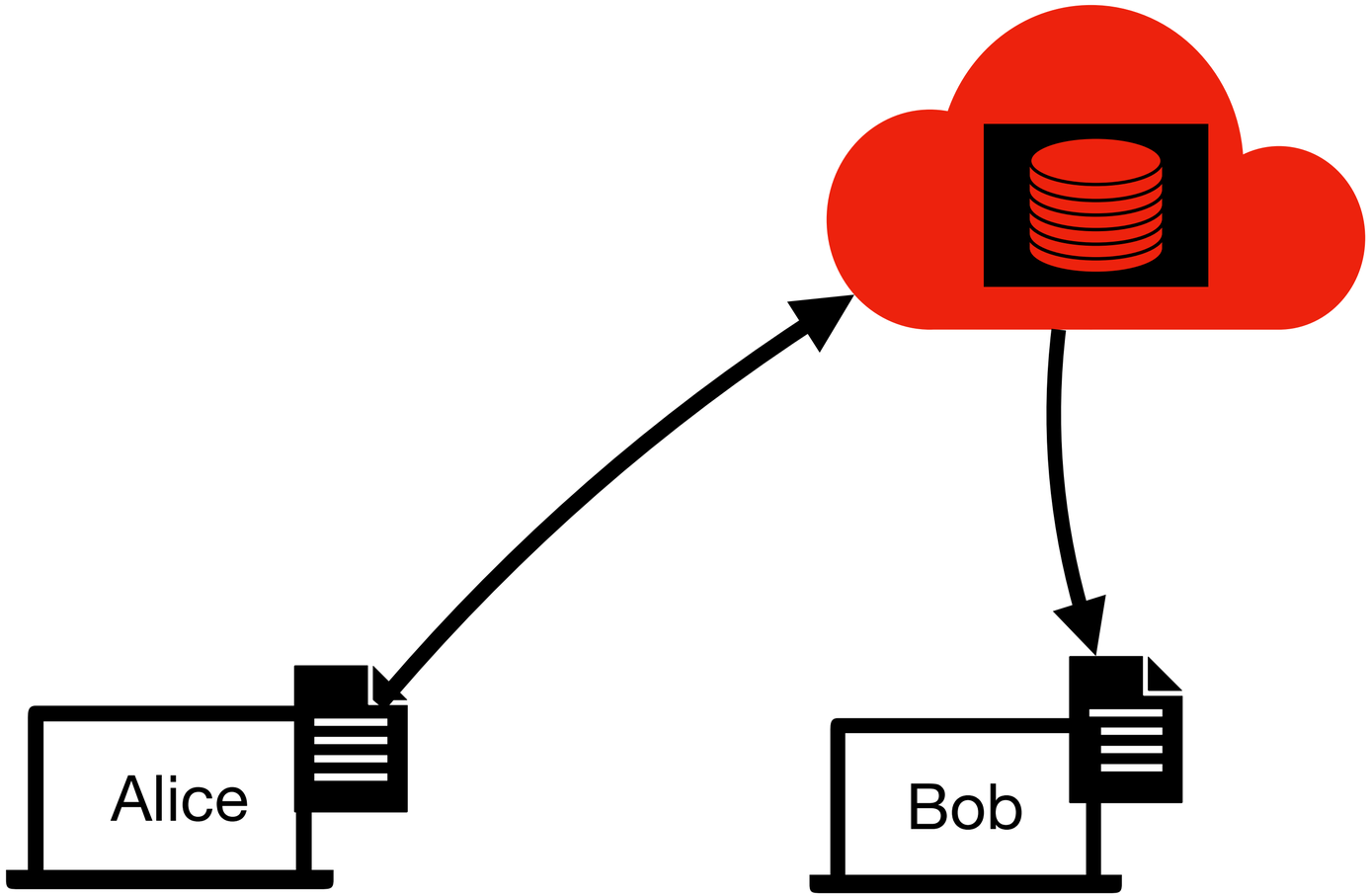
- Goal: E2EE Files, chat in a commercially-useful system
- In scope:
  - multiple devices per user: get new device, all data available immediately
  - mutable teams of users, teams of teams, etc.
  - proper key rotation after device revoke or team member removal
- Two supported applications: KV-store and Git server
- Primary improvement: federated servers (a protocol, not a business)
- Various manifestations of 2nd (3rd?) system syndrome
  - PQ-security, U2F support (lecture 15), improved transparency tree construction, various privacy improvements, more flexible teams, etc.

Outline

✓ ~~Motivation~~

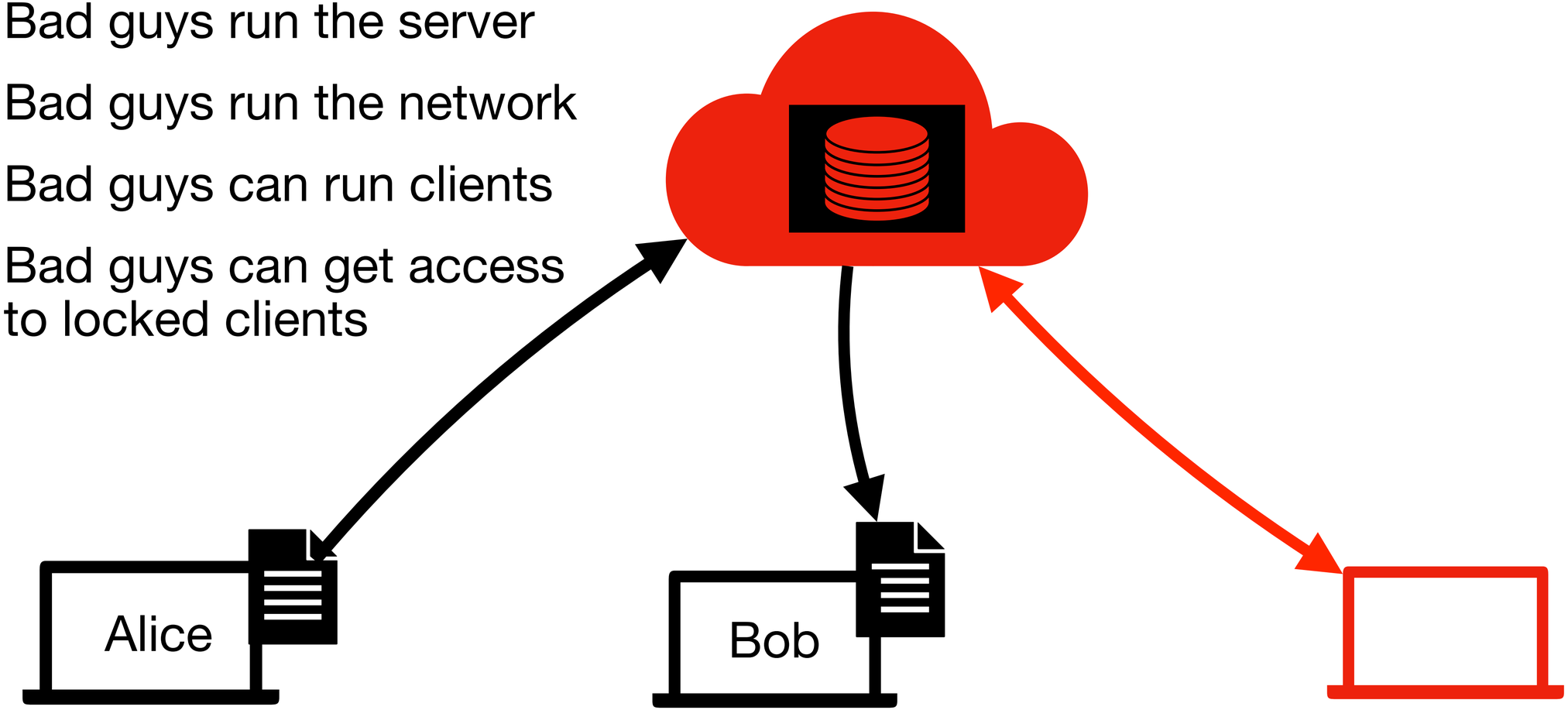
- Threat model
- Devices & Users
- Teams
- Transparency Tree
- Federation

Threat Model



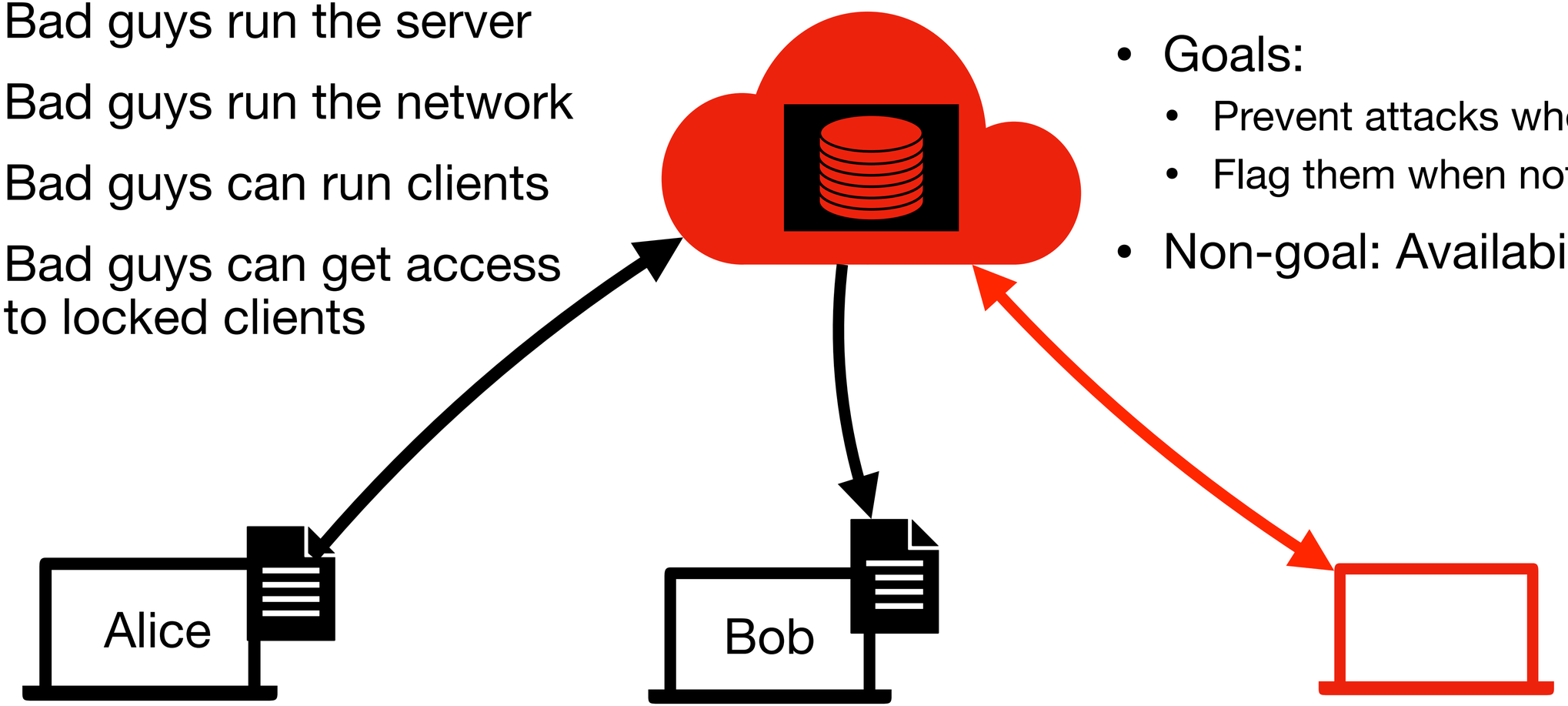
Threat Model

- Bad guys run the server
- Bad guys run the network
- Bad guys can run clients
- Bad guys can get access to locked clients



Threat Model

- Bad guys run the server
- Bad guys run the network
- Bad guys can run clients
- Bad guys can get access to locked clients



- Goals:
  - Prevent attacks when possible
  - Flag them when not
- Non-goal: Availability


GENE
HACKMAN
DANNY
DeVITO
DELROY
LINDO
HEIST


FROM PRODUCERS OF BREAKING BAD
HALT AND CATCH FIRE
THE BATTLE FOR CTRL BEGINS.

Outline

✓ ~~Motivation~~

✓ ~~Threat model~~

- Devices & Users
- Teams
- Transparency Tree
- Federation

Devices & Users

Desired behavior

- Step 1: Signup on laptop
- Step 2: Install app on phone, and get instant access to account.

Devices & Users

Bad idea 1: Master Password

- When user signs up, enter a password
- Derive all necessary crypto keys from “stretched” password
- Adding a new device just requires input of password on new device
- Why is this a bad idea?

Devices & Users

Bad idea 2: U2F Master Key


11 161 408
VL12

Devices & Users

Idea 2b: Two U2Fs

Devices & Users

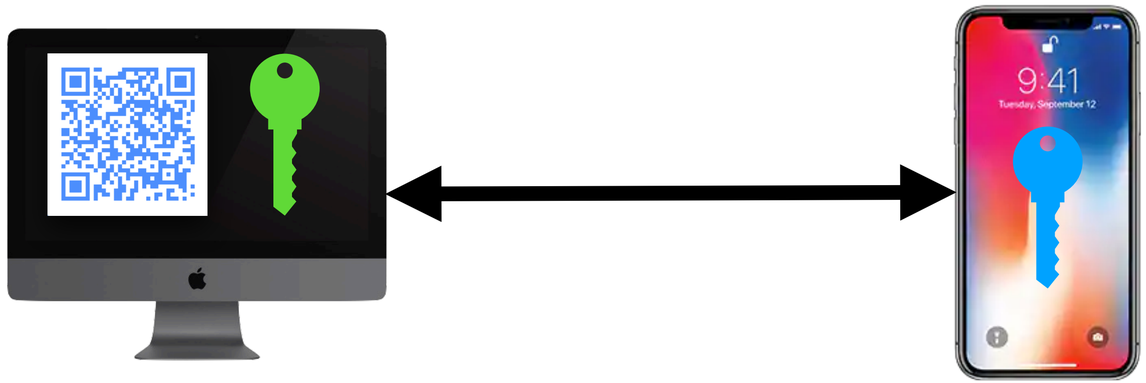
Bad Idea 3: One Device to Rule them All


Pavel Durov
@durov
WhatsApp's "E2E encryption by default" claim is a giant consumer fraud: ~95% of private messages on WhatsApp end up in plain-text backups on Apple/Google servers — not E2E-encrypted. Backup encryption is optional, and few people enable it — let alone use strong passwords.
10:40 AM · Apr 12, 2026 · 17.9M Views
852
2.5K
12K
2.3K


Pavel Durov @durov · Apr 12
Even if you encrypt your WhatsApp backups with a strong password, your messages still end up in unencrypted cloud backups — because 90%+ of the people you message haven't done the same. Add the fact that WhatsApp stores and discloses who you chat with, and the picture is dire.
86
400
3.7K
337K

Devices & Users

The right idea



- No passwords; secret keys never leave devices
- “Seeing Is Believing” Device Pairing (McCune & Perrig 2008)
- Keybase, current WhatsApp, current Signal
- Kind-of: Apple iCloud
- FOKS

Devices & Users

Approach

- Users think about “devices” not “keys”
- Private key never leaves device (not true of iCloud)
- No passwords
- Each device in a user’s cloud is equally powerful. Why?
  - We’ve all lost phones, laptops, slips of paper
  - The more devices, the less likely you are to lose your data
  - And you’re most likely to discard your **oldest** device

Devices & Users

Key Hierarchy

- Each device gets its own key pair, private key never leaves
- Each user gets their own *Per-User Key* (PUK)
  - Secret = 32-byte random seed
  - Derived: Signing Keypair + DH Keypair + ML-KEM keypair
- The secret seed of the PUK is encrypted for all the device public keys; encryptions sent to server
- Adding a device: encrypt the PUK seed for the new public device key (Fast!)
- Revoking a device: generate a new PUK; encrypt secret seed for all remaining public device keys (Slower, but can happen lazily)
- PUK = “how to encrypt for Alice; or verify a statement by Alice”

Devices & Users

First Device

- Alice generates:
  - new device secret *s*, stores in local hardware enclave
  - first PUK
  - user ID
- Derives signing key pair, DH secret key pair, and ML-KEM key pair from *s*
- Encrypts the PUK secret for the device public keys derived from *s*
- Generates commitments for username and device name
- Signs public keys, user ID, commitments with device signing key
- Posts to server as self-signed user creation statement

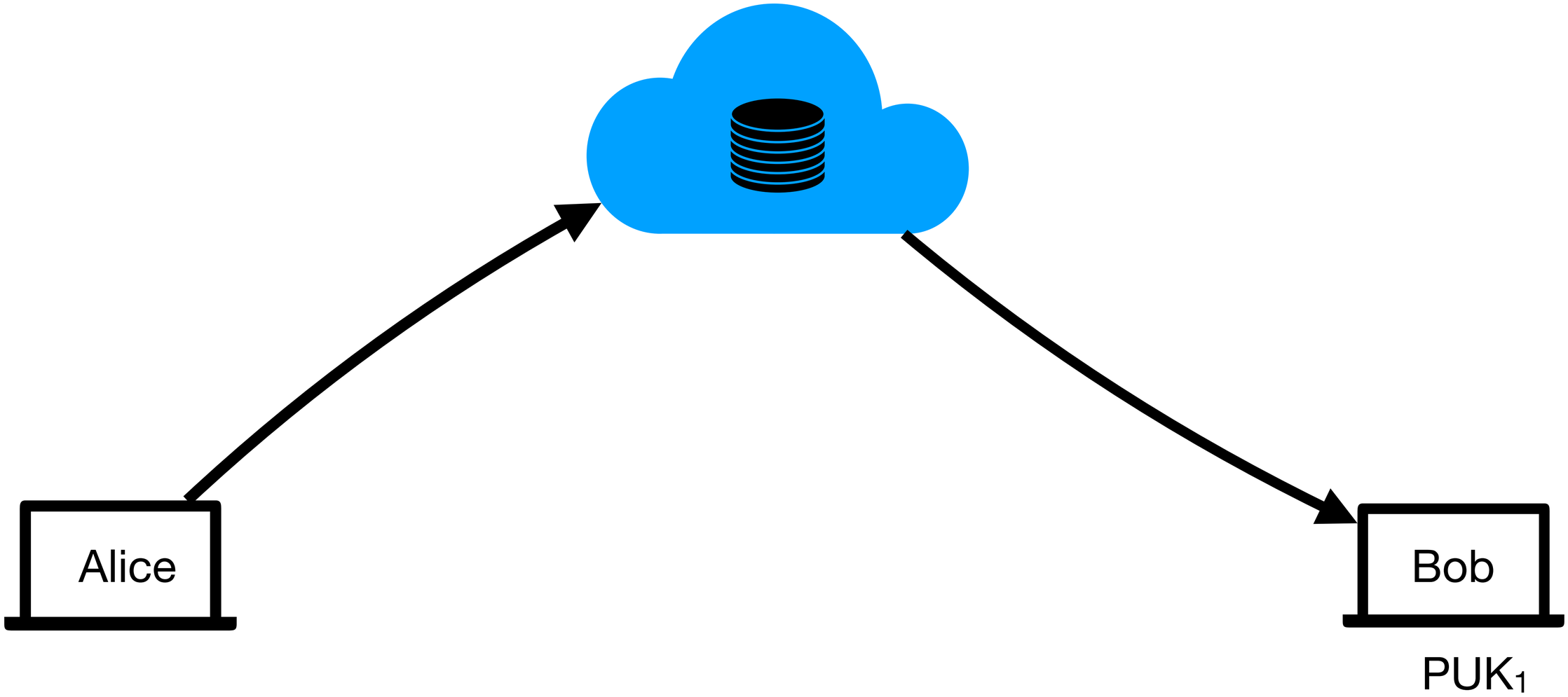
Devices & Users

Cryptographic Commitments

- Given a public string *n*, something like a username or device name
- Generate a random string *r*
- Commitment = *H*(*r*,*n*)
- Can sign permanent, public statements about the commitment
- Without *r*, everyone sees you committed to *something;* you can’t change what it is
- Knowing *r*, one can verify you committed to the string *n*
- Server can later delete *n* and *r*, if you want to erase all traces of *n*

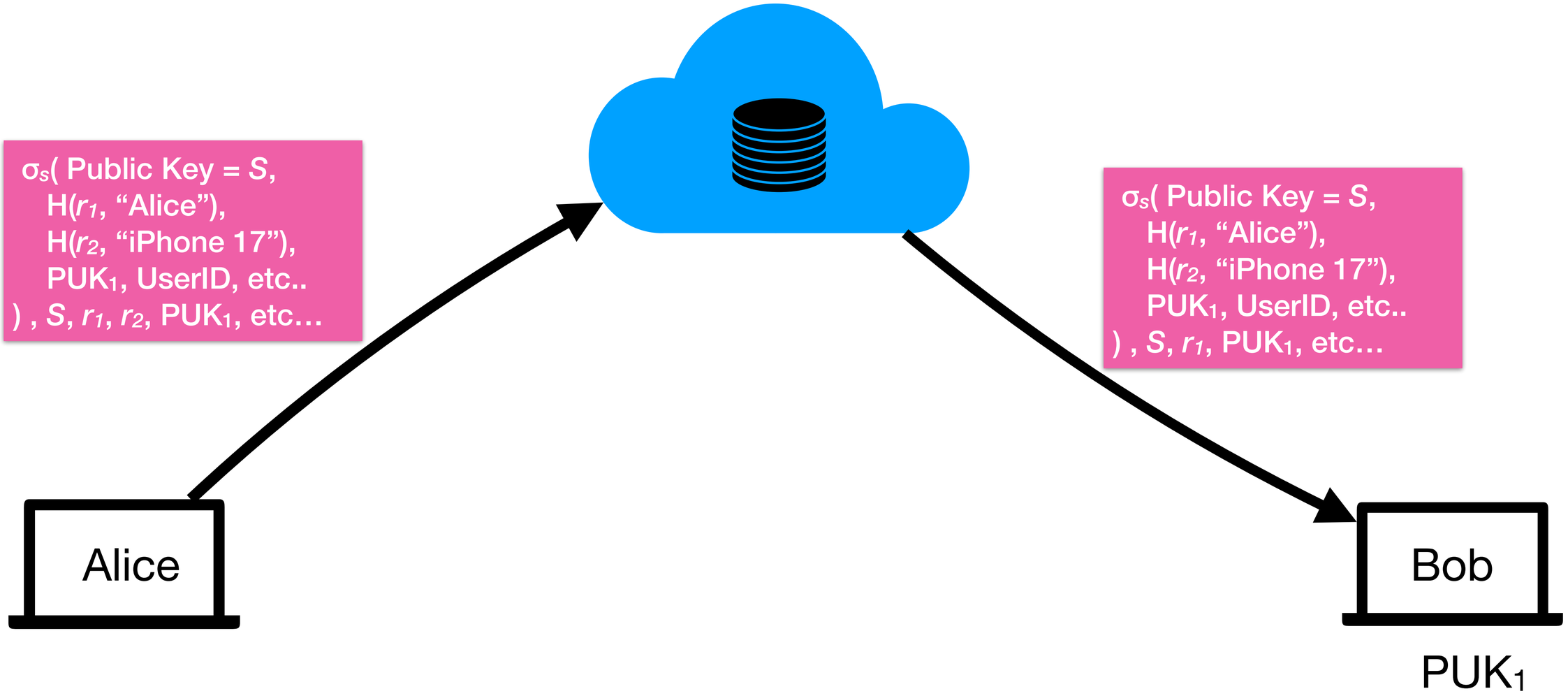
Devices & Users

First Device: Put/Get



Devices & Users

First Device: Put/Get



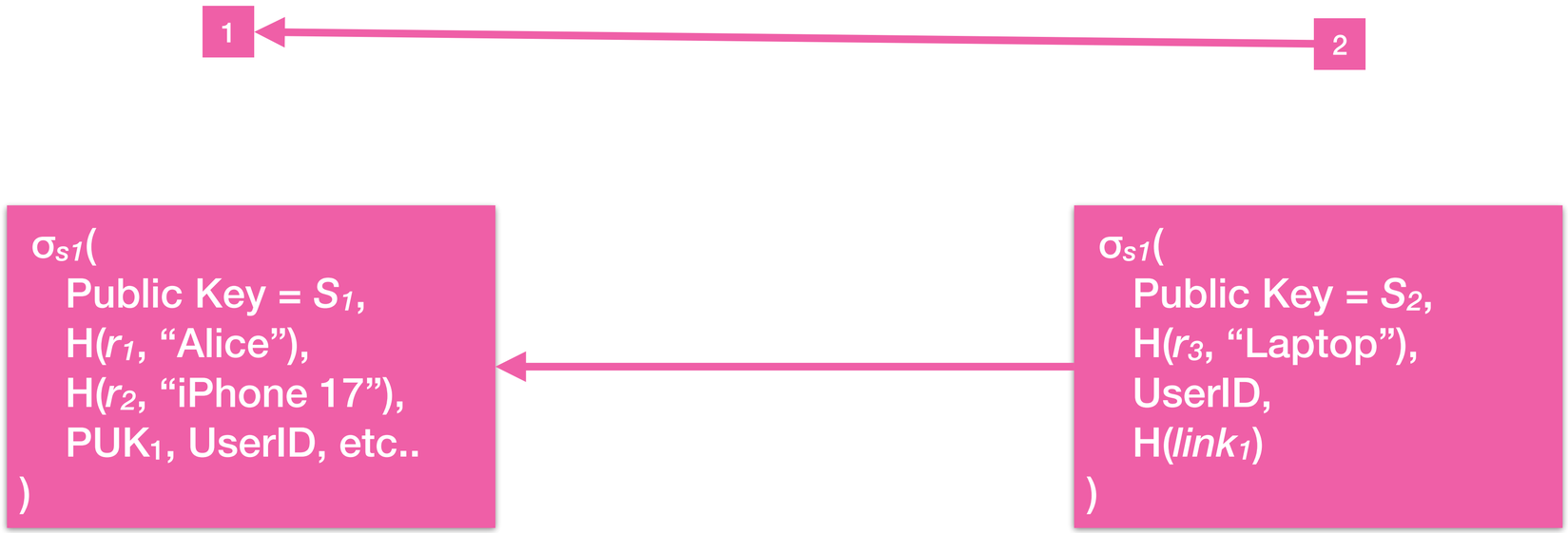
Devices & Users

Addition

- On the new device, Alice:
  - Generates a new secret seed, s_2, stored in local hardware enclave
  - Derives public keys from s_2
  - Makes a new commitment for the new device name
- On the existing device
  - Generates a hash of the previous statement
  - Signs all the above
  - Encrypts the PUK secrets for the new device
  - Posts all to the server

Devices & Users

User “Sigchain”



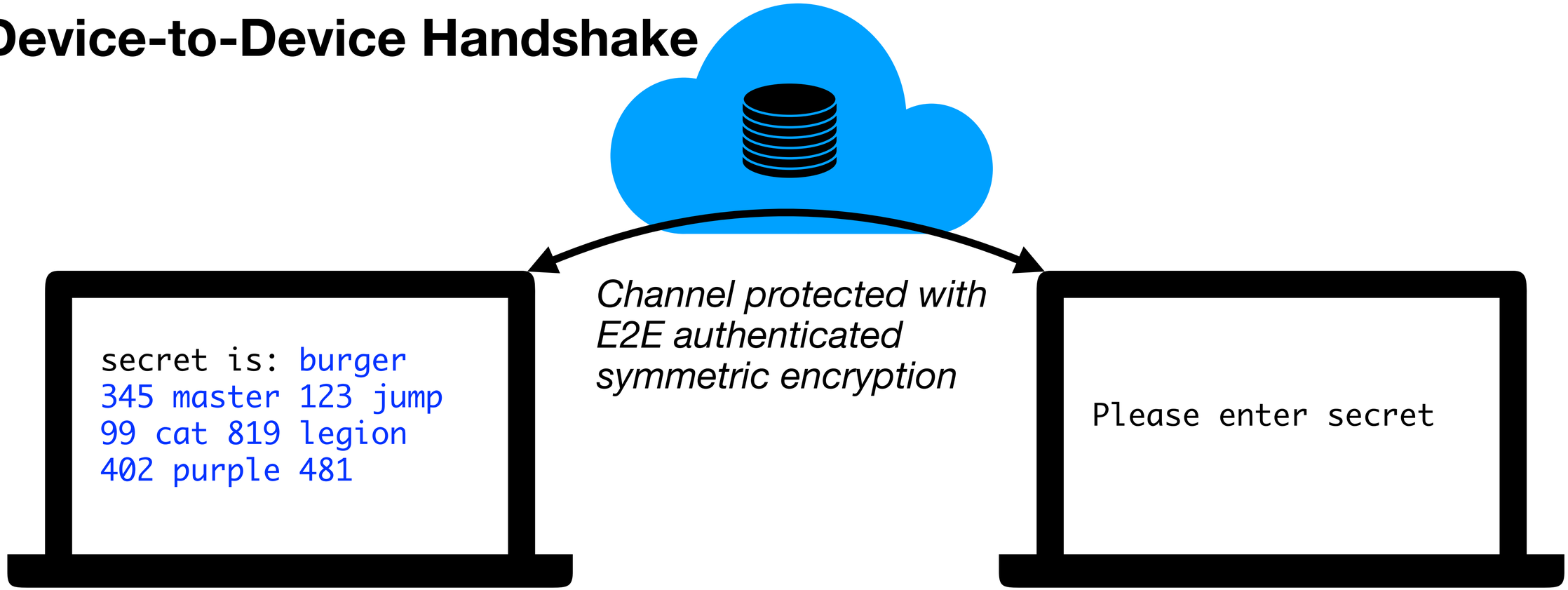
Devices & Users

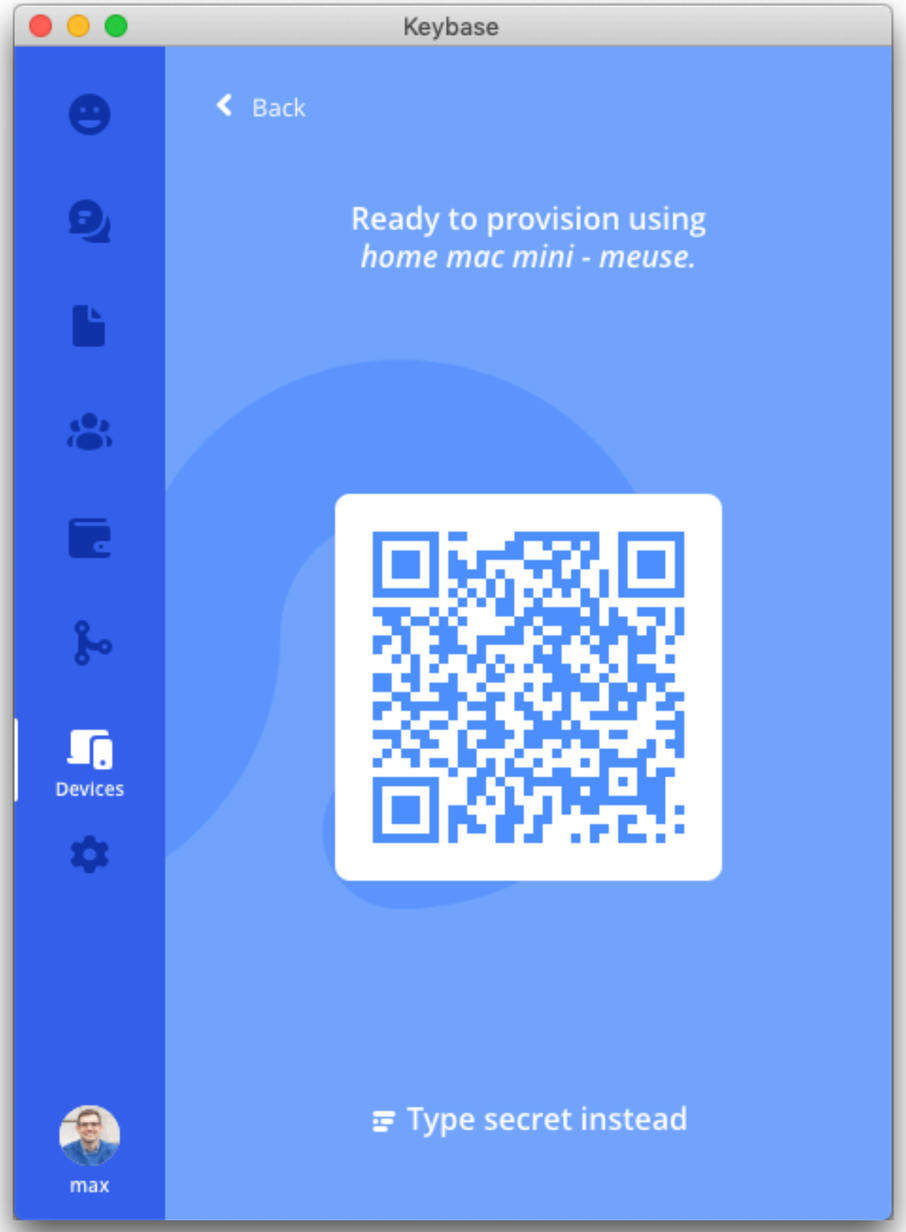
Addition

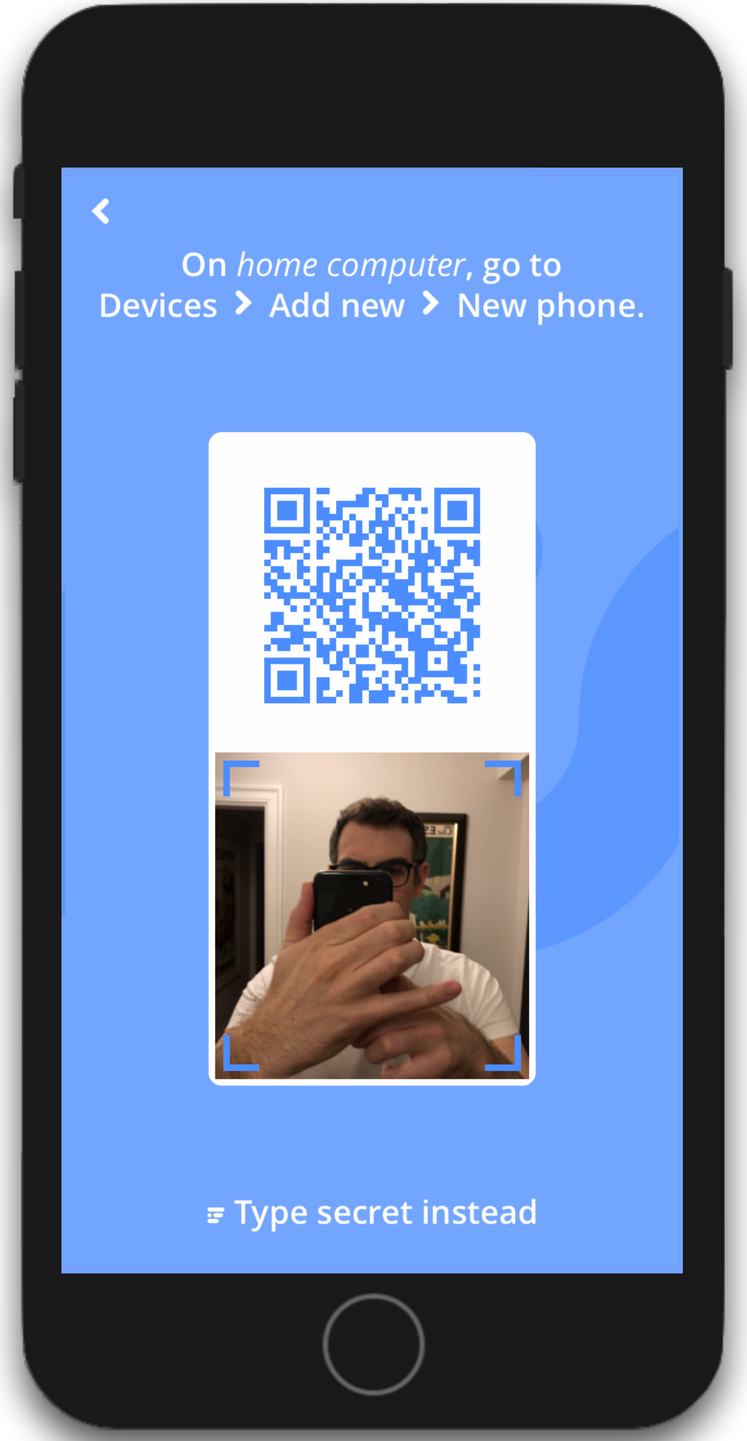
- On the new device, Alice:
  - Generates a new secret seed, s_2, stored in local hardware enclave
  - **Derives public keys from s_2**
  - Makes a new commitment for the new device name
- On the existing device
  - Generates a hash of the previous statement
  - **Signs all the above**
  - Encrypts the PUK secrets for the new device
  - Posts all to the server

Devices

Device-to-Device Handshake




Keybase
Back
Ready to provision using
home mac mini - meuse.
Devices
max
Type secret instead


On home computer, go to
Devices › Add new › New phone.
Type secret instead

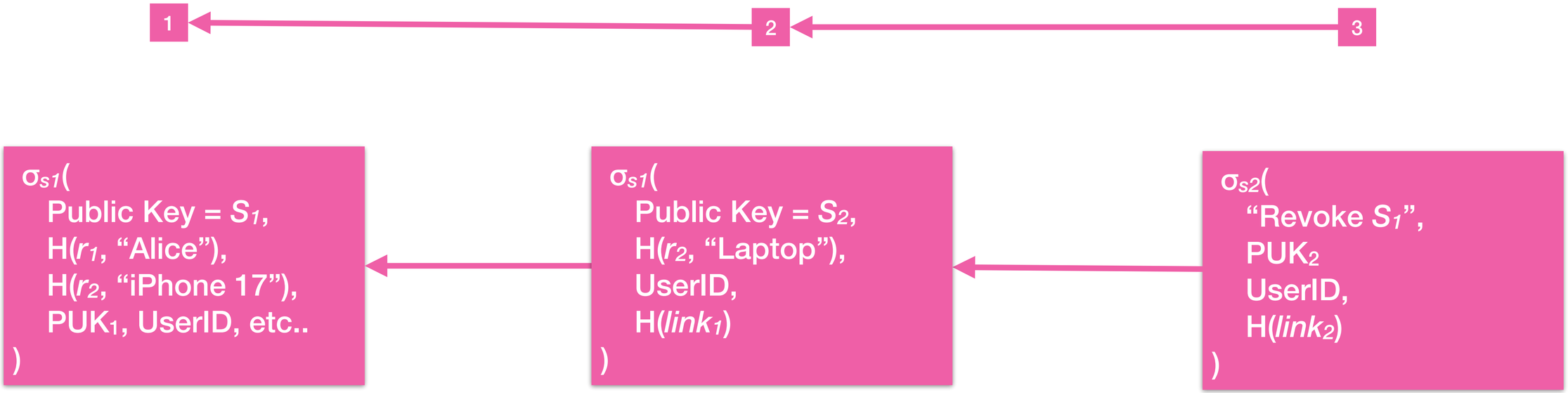
Devices & Users

Deletion

- On a remaining device, Alice:
  - Generates a new PUK
  - Generates a hash of the previous statement
  - Signs a revoke statement of the above
  - Encrypts the PUK secrets for the new device
  - Posts all to the server

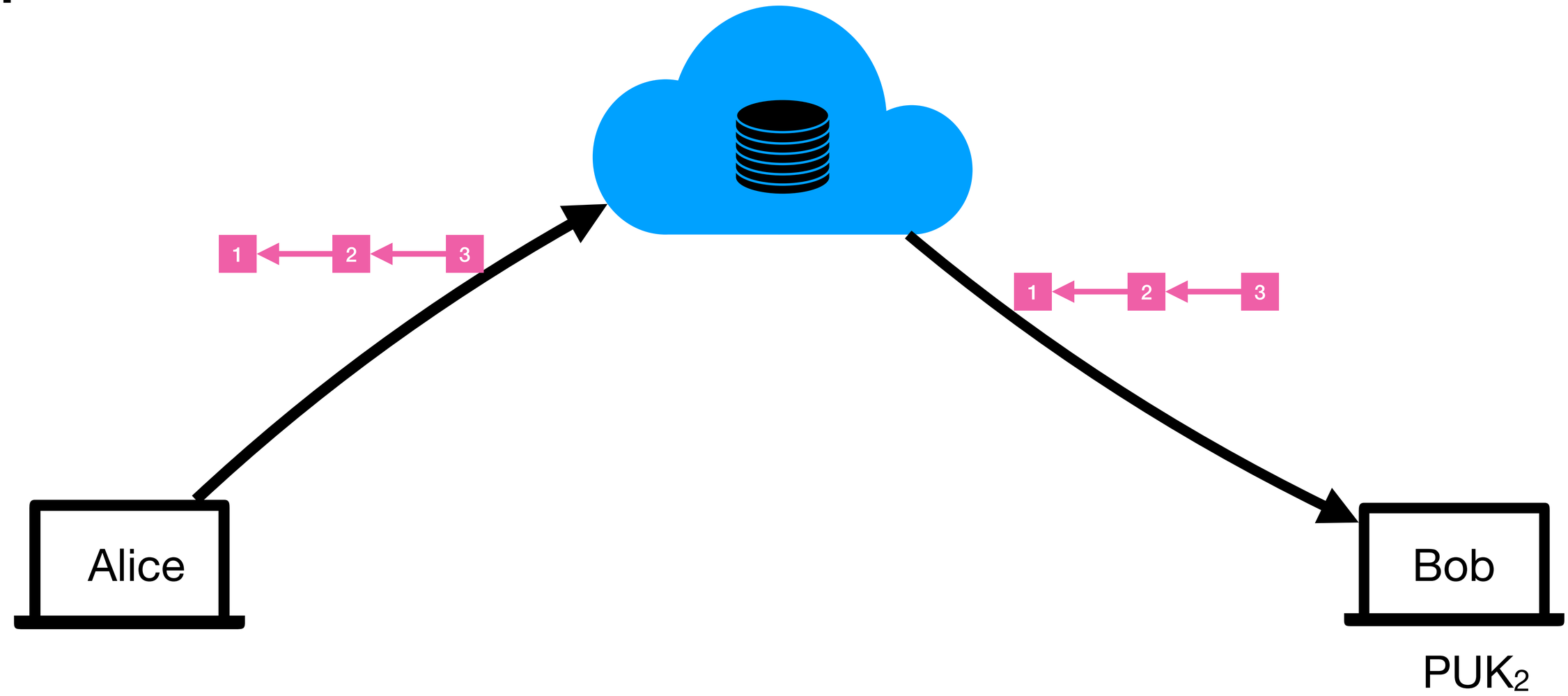
Devices & Users

User “Sigchain”



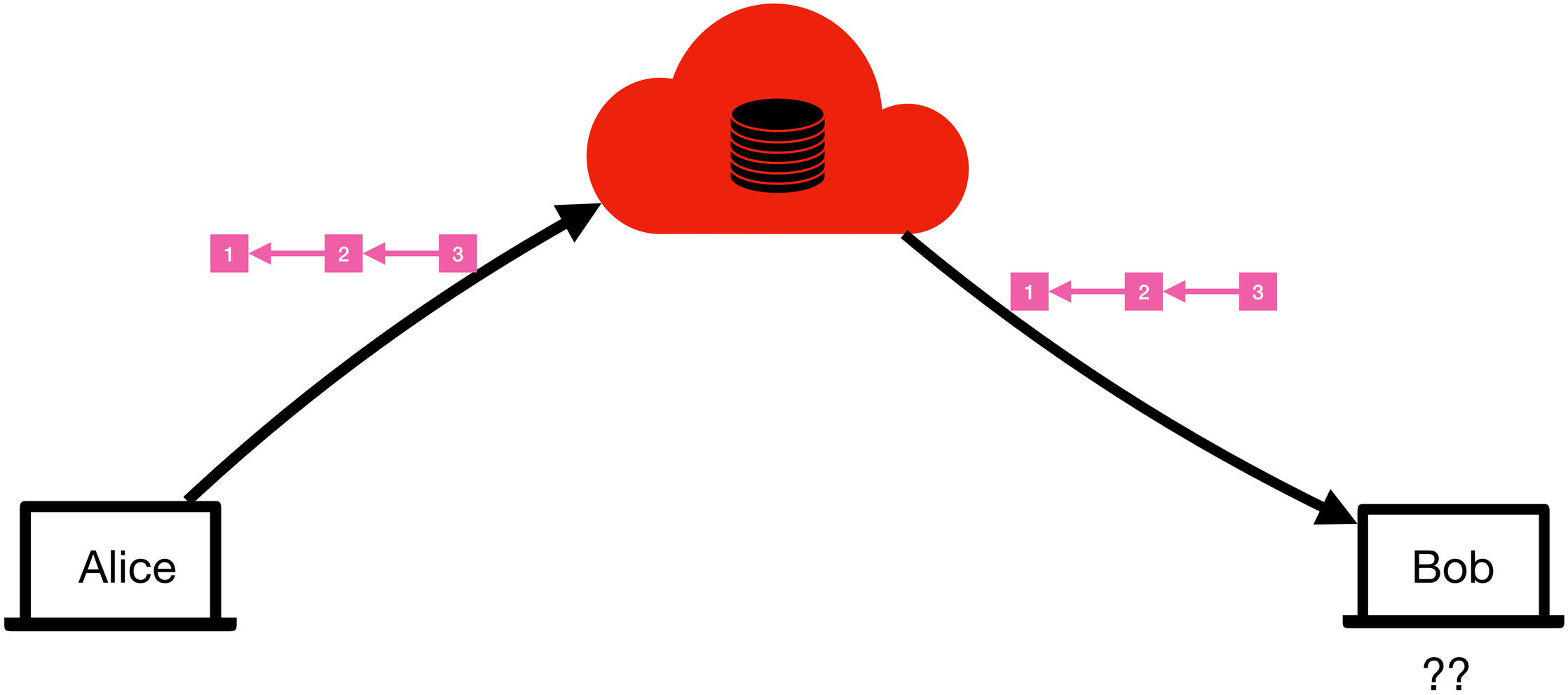
Devices & Users

Updated Put/Get



Devices & Users

Question: What if the server is evil?



Devices & Users

Question: What if the server is evil?

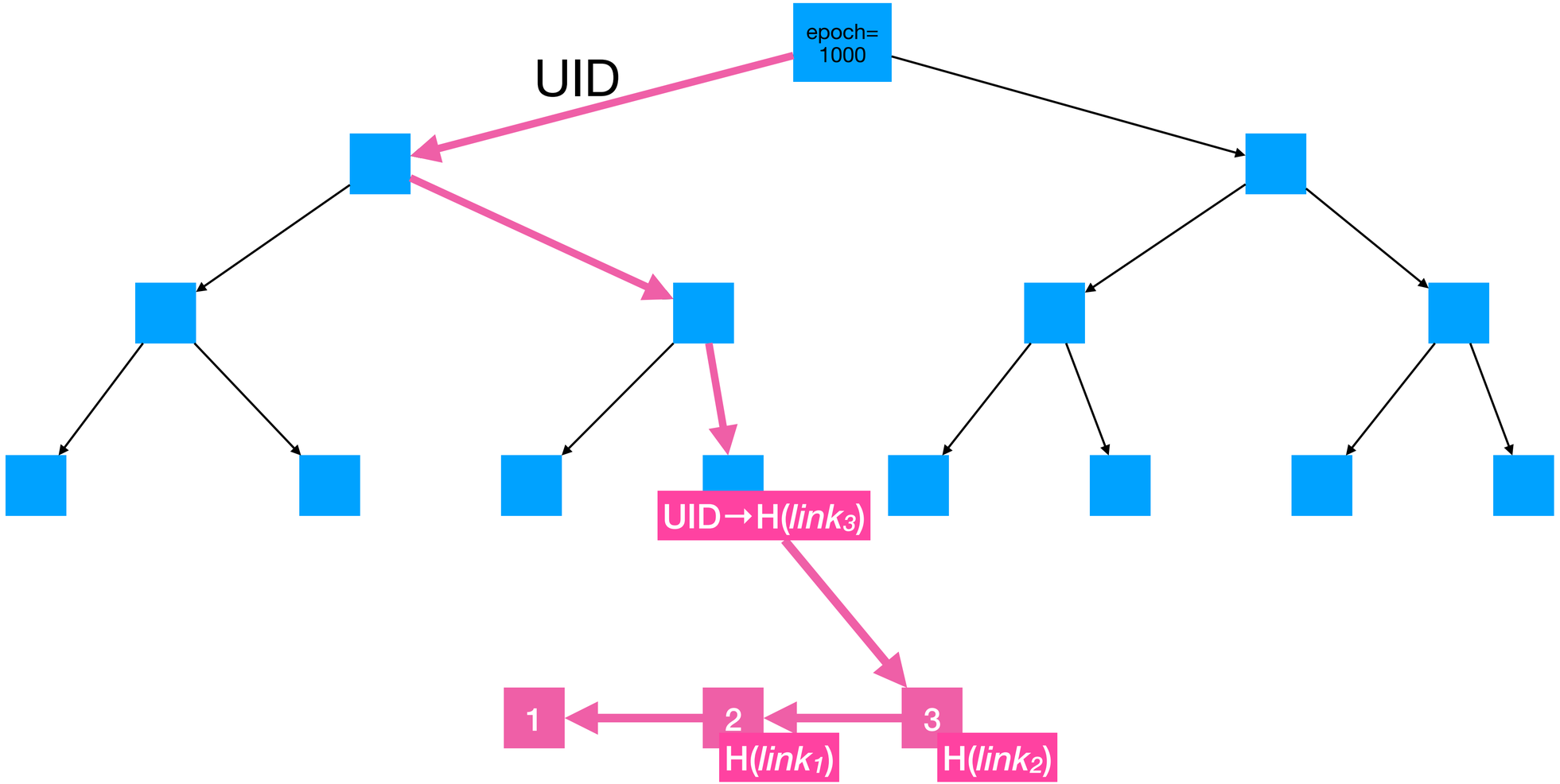
Devices & Users

Question: What if the server is evil?

- Wholesale substitution attack
- Forking attack
- Withholding attack

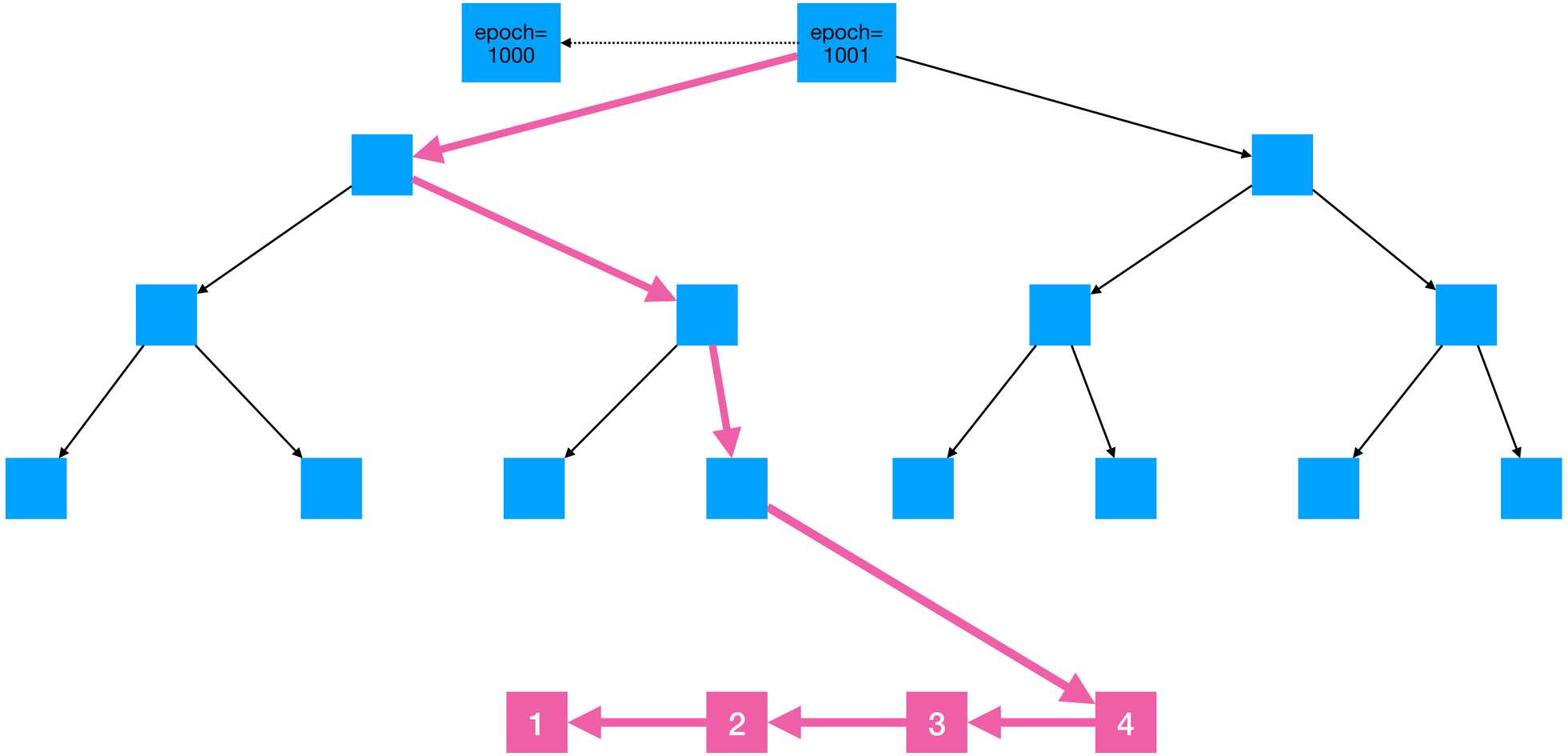
Transparency Tree

One hash encapsulates the whole system



Transparency Tree

The root updates whenever any leaf changes



Devices & Users

Bob looks up Alice with Merkle Tree

- Download Merkle root from server, and verify explicit signature (i.e., don't just trust TLS). (Why?)
- Descend the Merkle tree to Alice's leaf
- Fetch tail of her "sigchain" and confirm the returned sigchain ends in the advertised tail
- As before

Devices & Users

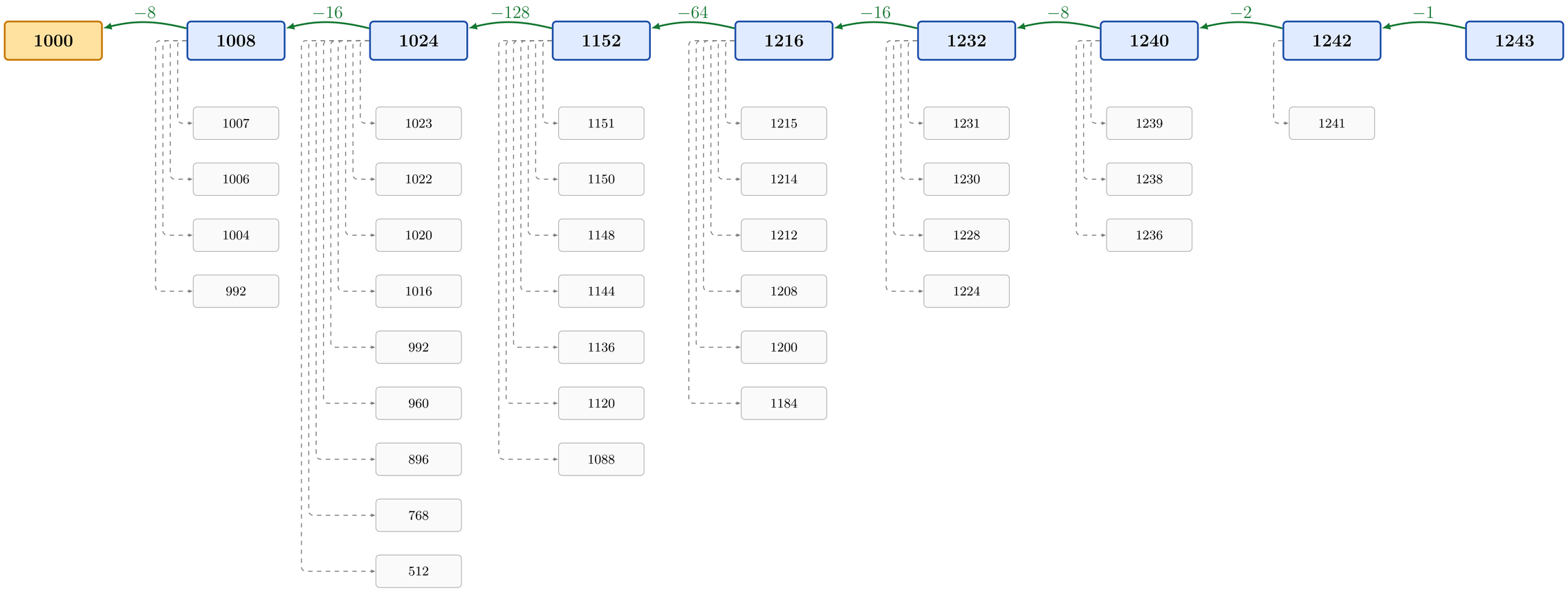
Additional Bookkeeping

- Whenever Bob looks up Alice at time t_1 and t_2, he asserts the new links fit at the end of the chain
- Whenever Bob looks up at time $t_2 > t_1$, ensures:
  - The global Merkle epoch # has increased
  - And that the global Merkle root points back to the earlier root via logarithmic “skip pointers”

Transparency Tree

Skip Pointers

MerkleSkipPointers$(1243 \to 1000)$



Path: $1243 \to 1242 \to 1240 \to 1232 \to 1216 \to 1152 \to 1024 \to 1008 \to 1000$
Path nodes: 8 **Siblings:** 30 **Total skip pointers:** 38
Bounds: $|\text{path}| < 2\log_2 1243 \approx 22$, $|\text{sib.}| < (\log_2 1243)^2 \approx 121$

Transparency Tree
Fork Consistency

- See SUNDR (Li & Mazierès 2004)
- Server can still fork the Merkle Tree and show different versions to Bob and Alice
- Must maintain the fork indefinitely and has no viable strategy for merging
- Alice and Bob write down all signed roots; can later present conflicting roots to an auditor as definitive proof of malfeasance
- Keybase historically pinned the root of the tree to Bitcoin or Stellar blockchains, to mitigate this attack.

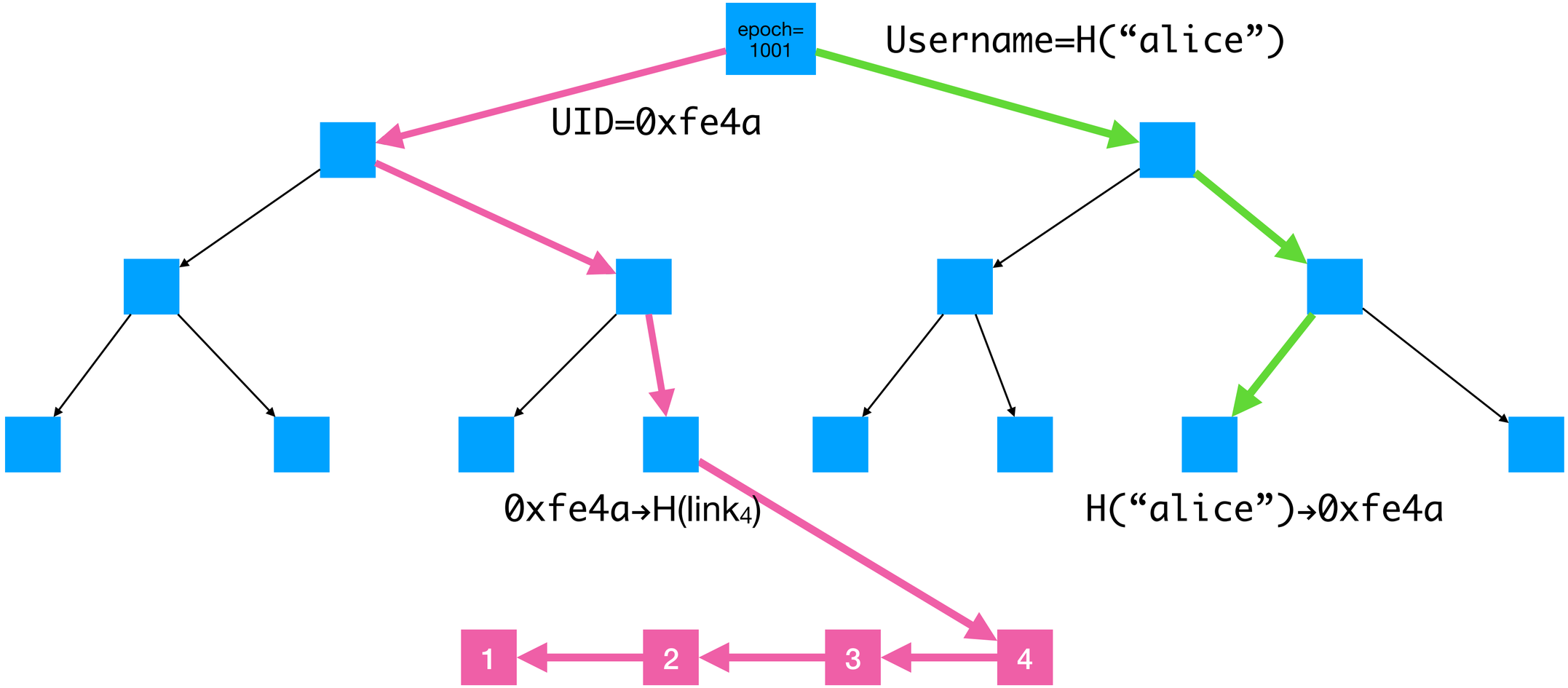
Devices & Users

Username mutability

- Lookup users by UID in Merkle Tree
- If usernames are immutable, then we can pick UID = H(username)
- What about if not?

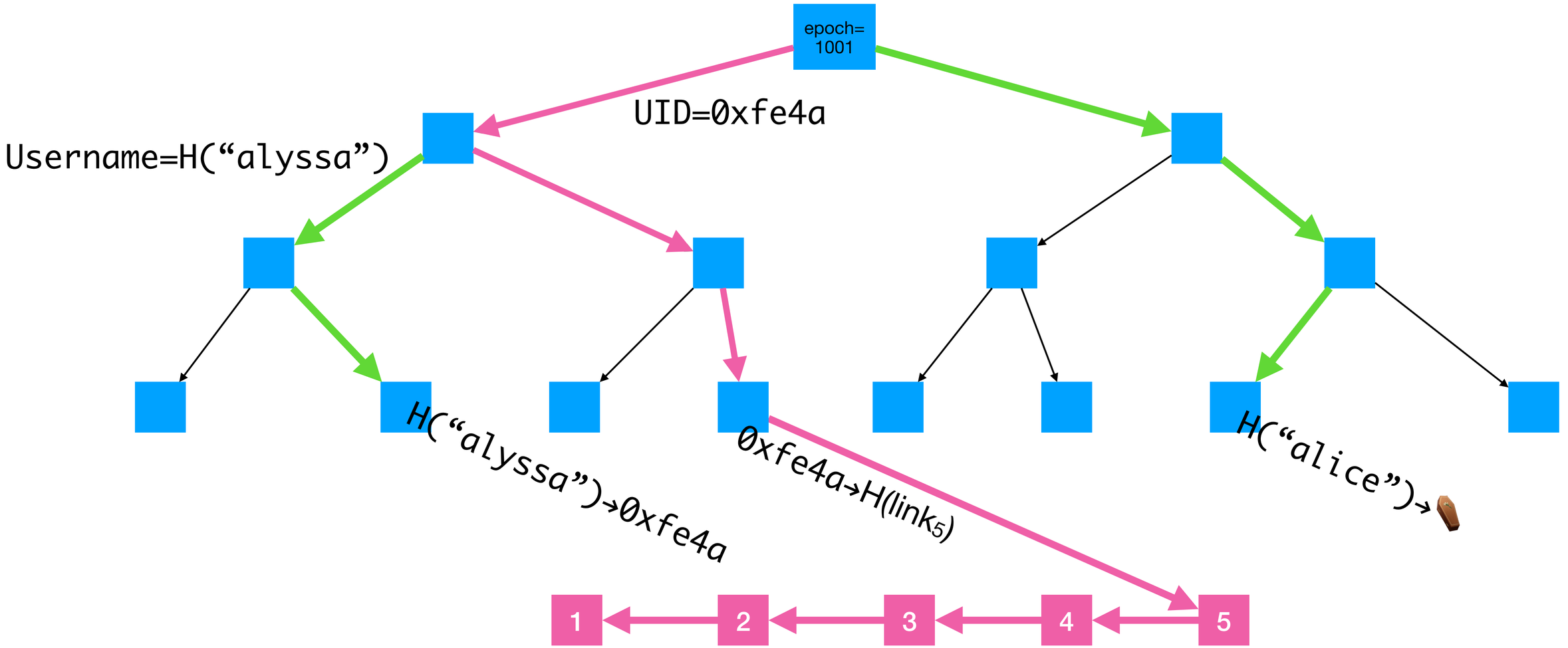
Transparency Tree

Usernames & UIDs



Transparency Tree

Username change



Outline

✓ ~~Motivation~~

✓ ~~Threat model~~

✓ ~~Devices & Users~~

- Teams
- Transparency Tree (redux)
- Federation

Teams

What is a team?

- A named group, consisting of users and other teams
- Mutable membership
- Owners: can create, delete; can add/remove owners, admins, members
- Admins: can add/remove members, admins
- Members: can read and write data
- All share symmetric keys that can be used to encrypt data like chats and files

Teams

Initialization

- Alyssa (owner) generates:
  - first PTK (per-team key)
  - team ID
- Encrypts the PTK secret seed for her PUK and Bob’s PUK
- Generates commitments for the team name
- Signs public keys, team ID, commitments with her PUK
- Posts to server as a signed team creation statement
- Sound familiar?

Teams

Addition of New Members

- Alice or Bob (team owners):
  - Signs statement with PUK that Charlie is now an owner of team
    - Includes Charlie’s PUK and a hash to the previous link in the team’s chain
  - Encrypts the team’s PTK secret seed for Charlie’s PUK
  - Posts to server

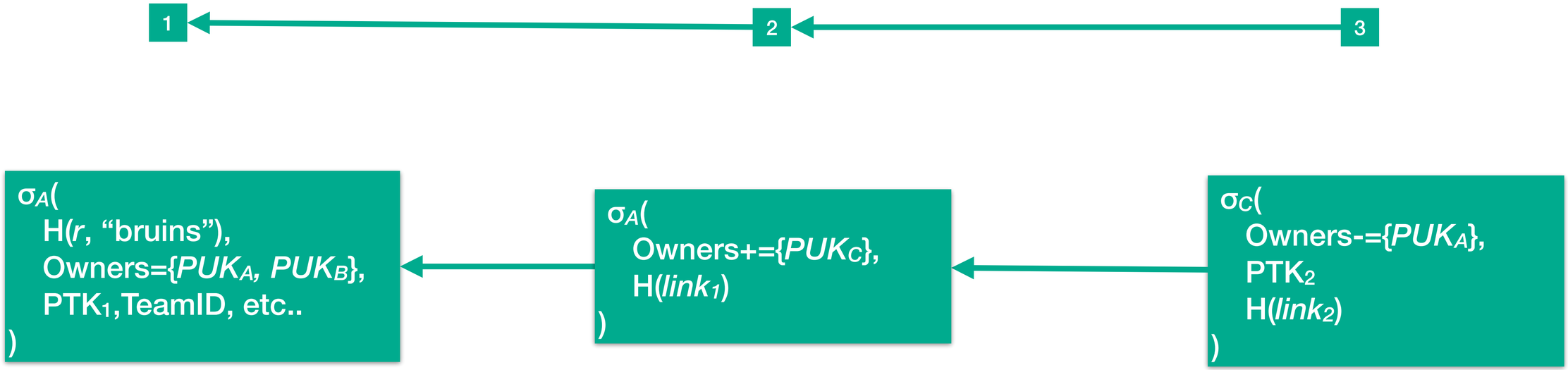
Teams

Removal of Members

- Charlie (team admin):
  - Signs statement with PUK that Alyssa is out of the team
  - Rolls a new PTK
  - Encrypts the new PTK secret seed for all remaining members
  - Posts to server

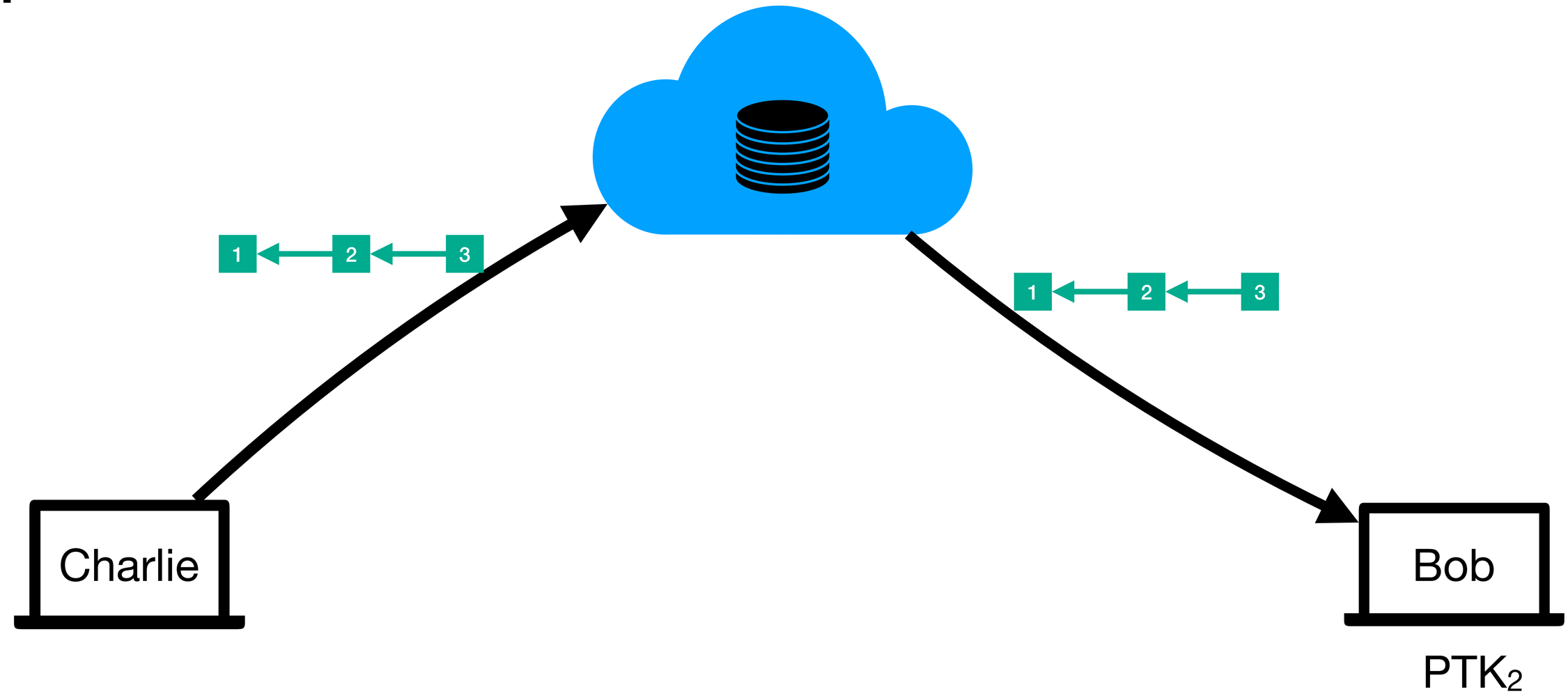
Teams

Team “Sigchain”



Teams

Updated Put/Get

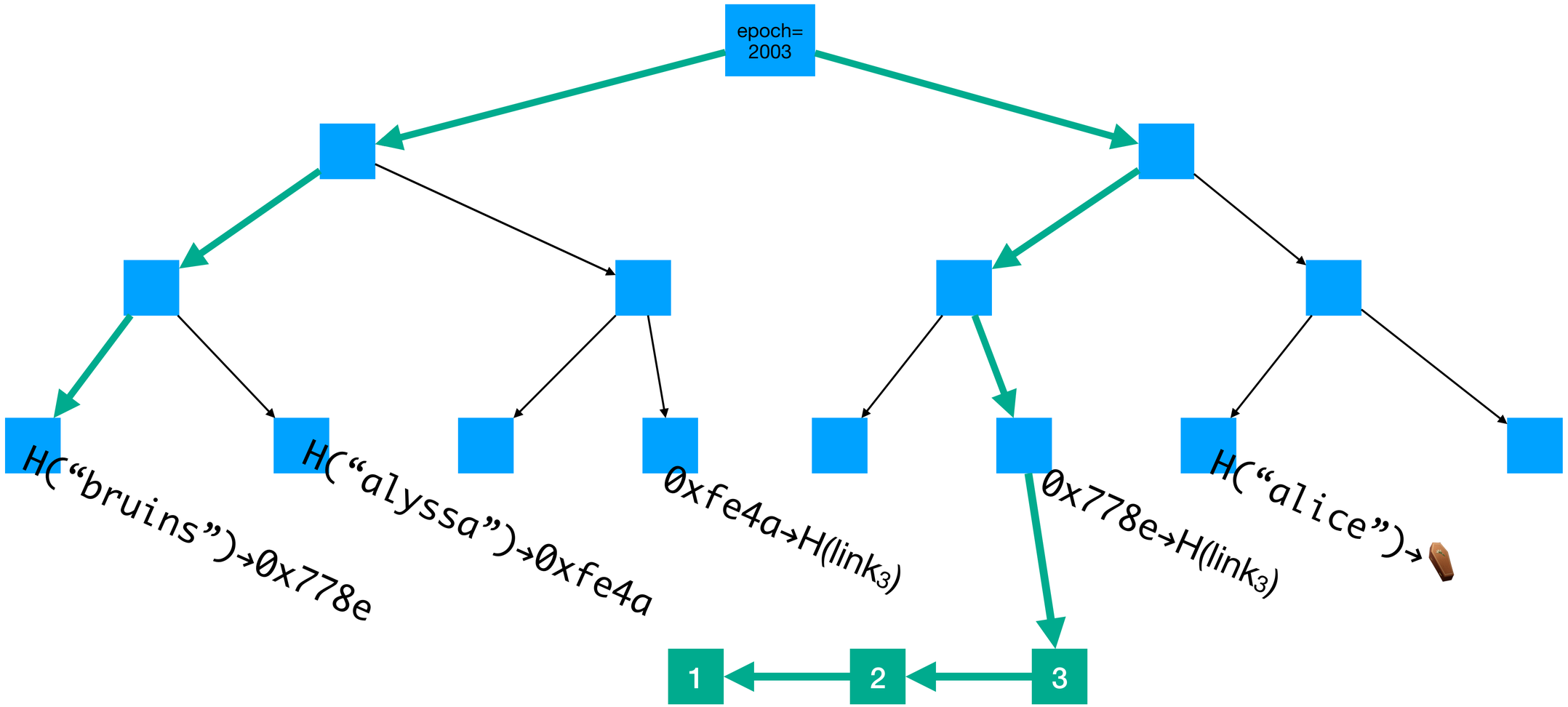


Teams

All done, right?

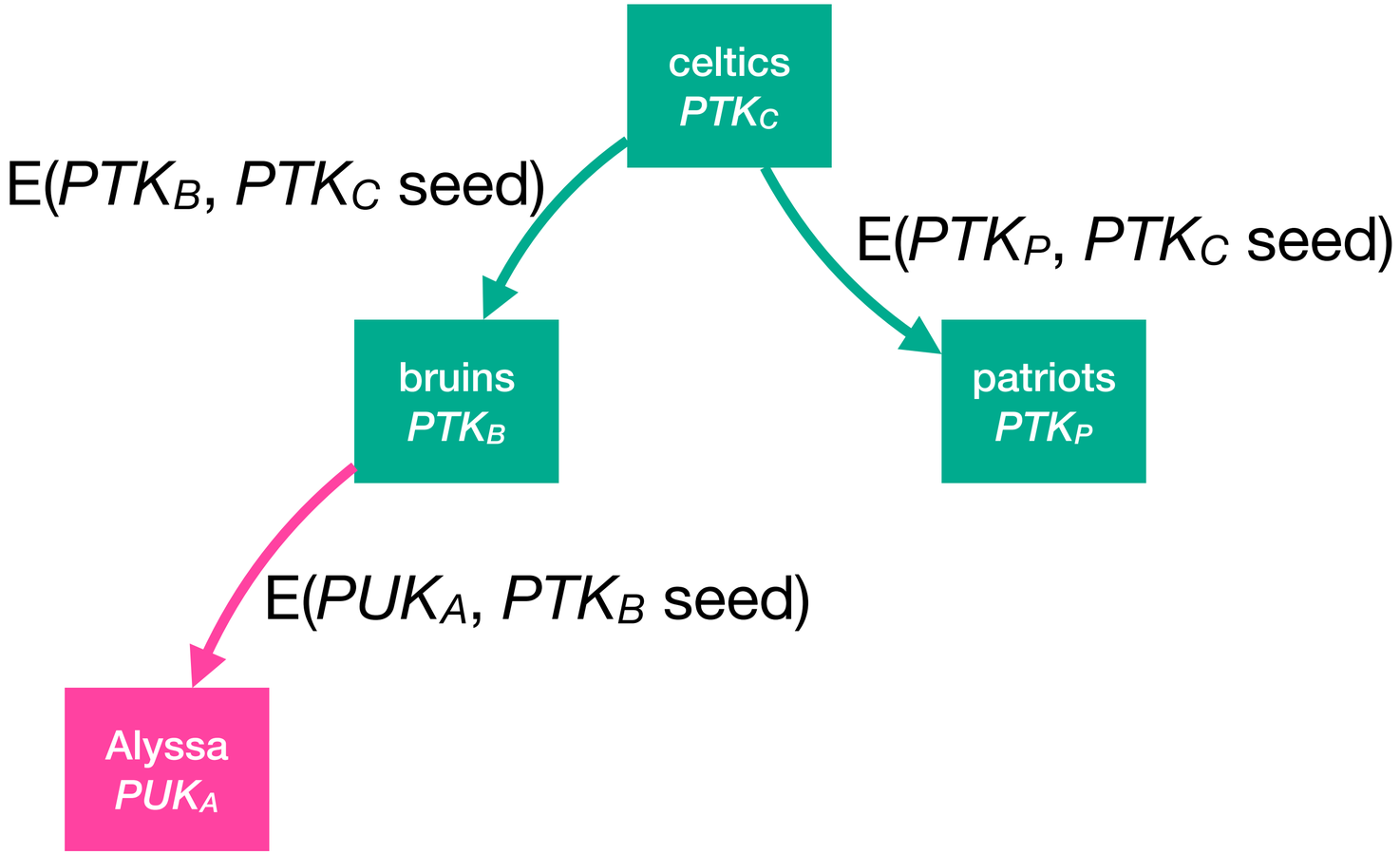
Transparency Tree

Bruins Included



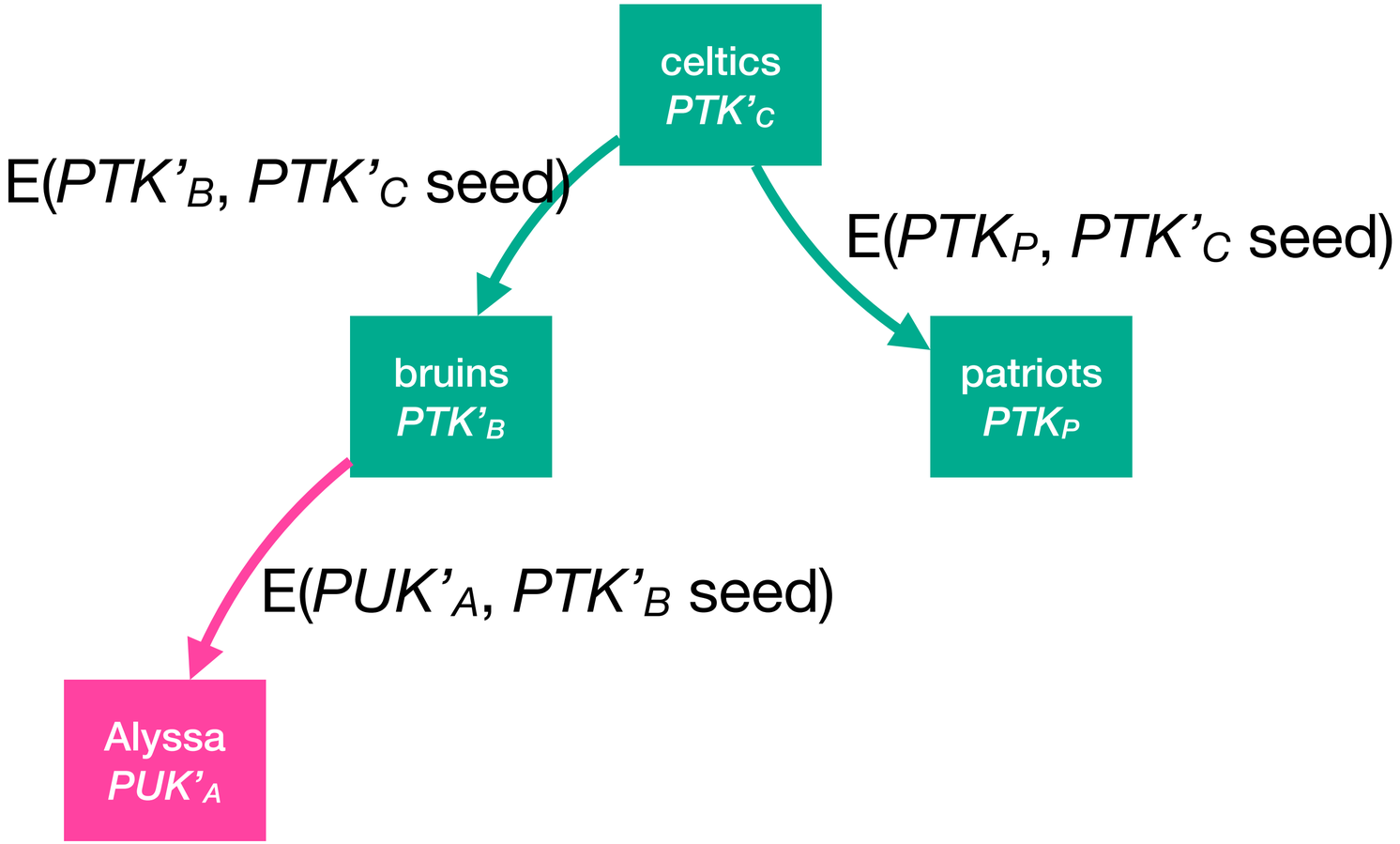
Teams

Nearly-Arbitrary Graphs



Teams

What happens when Alyssa revokes a device?



Teams

When Does Key Rotation Happen?

- What about on the critical path?
  - Consider self-revocation
  - Consider a “non-admin” of a team revoking a device
  - Consider clients can’t trust servers to provide push notifications
- Background polling of all reachable teams
  - Poll each team, and rotate when needed.
  - Clients keep “membership chains” to keep track of which teams a user (or team) is a member of. Why can’t the server do this?

Outline

✓ ~~Motivation~~

✓ ~~Threat model~~

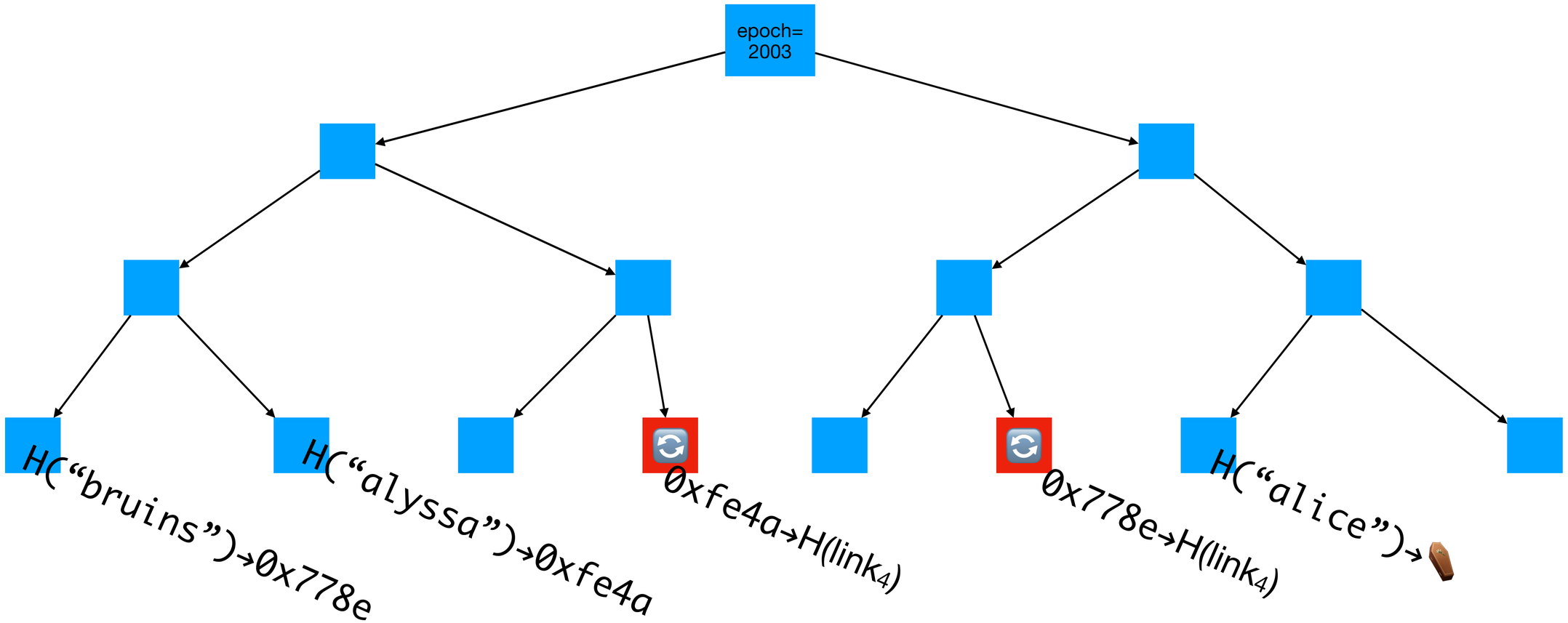
✓ ~~Devices & Users~~

✓ ~~Teams~~

- Transparency Tree (redux)
- Federation

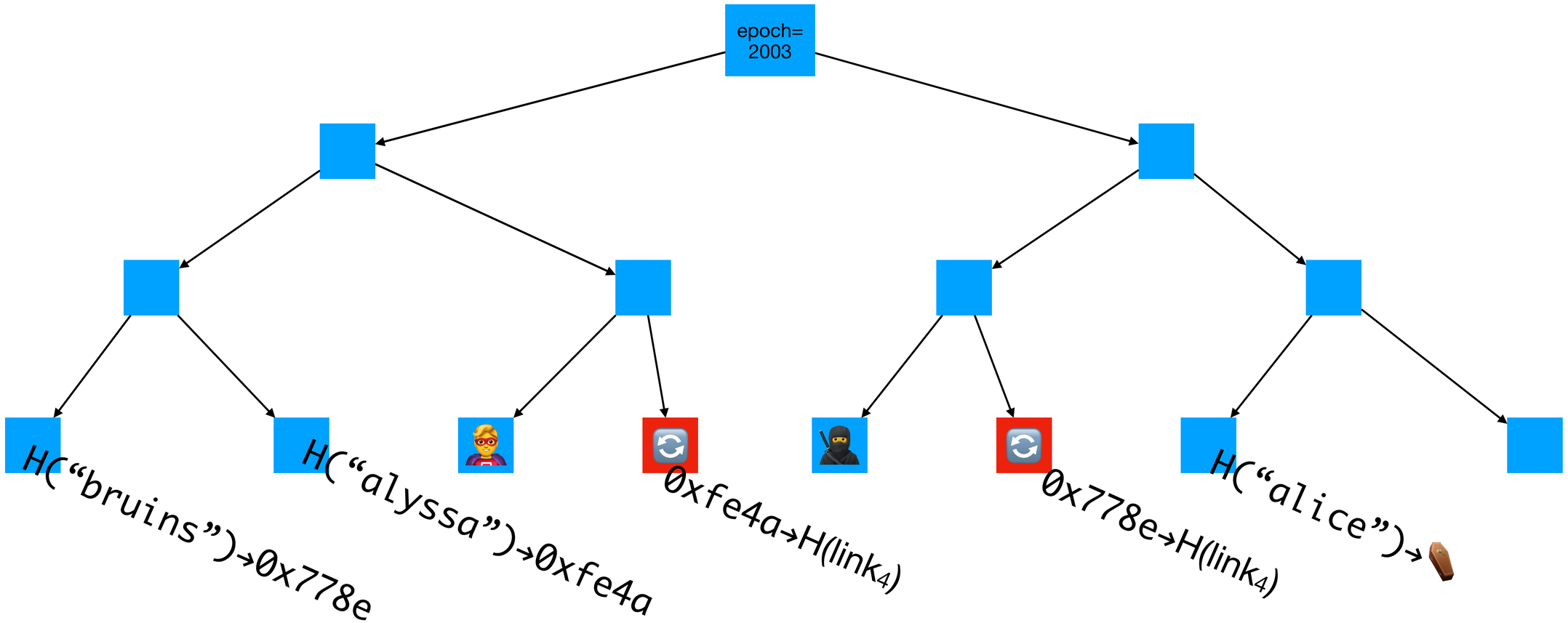
Transparency Tree

Correlated Changes



Transparency Tree

Correlated Changes



Transparency Tree

Simultaneously Hide and Advertise Changes?

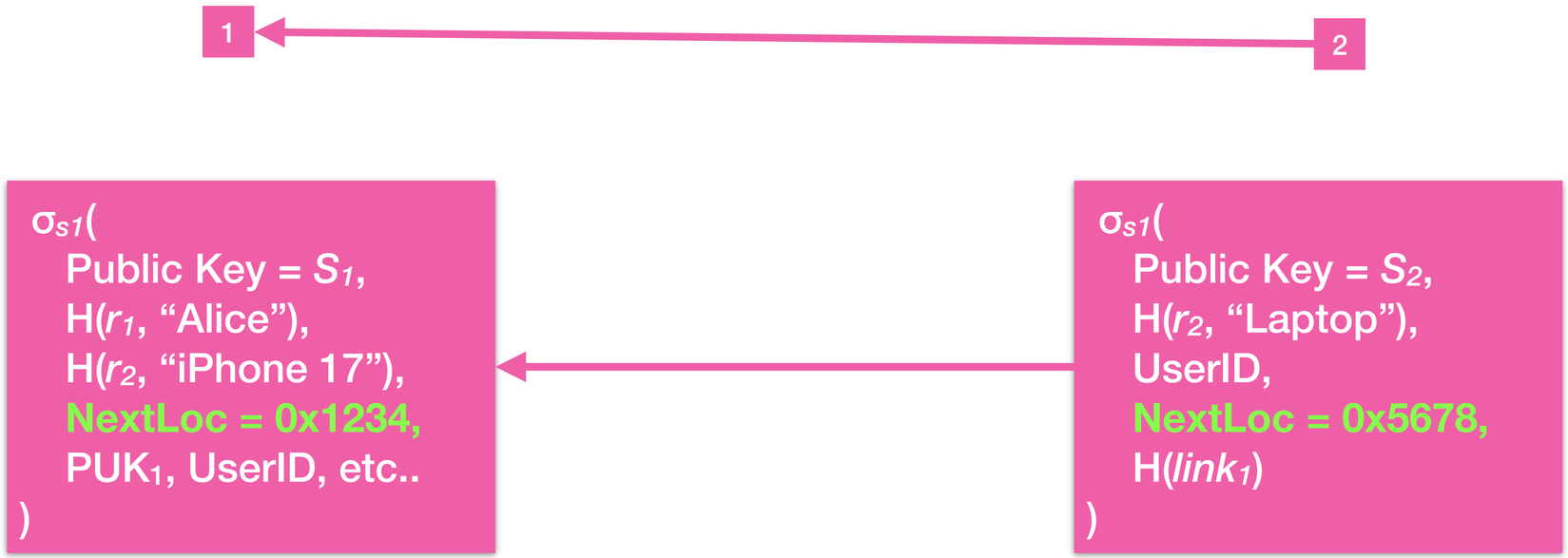
- Keybase: Store `0xfe4a` → $H(link_4)$ at position `0xfe4a`
- CONIKs 2015 / SEEMless 2018 / ELEKTRA 2024 use VRFs
  - Server has (pk,sk) pair
  - VRF(`0xfe4a`, sk) -> { `0xaabb`, proof }
  - Verify(`0xfe4a`, `0xaabb`, proof, pk)
  - Store `0xaabb` → $H(link_4)$ in tree
  - Doug can't guess where Alyssa is in tree (though he can verify it)
  - Problem: hard to rotate (pk,sk) if compromised

Transparency Trees Redux

User “Sigchain” Take 2

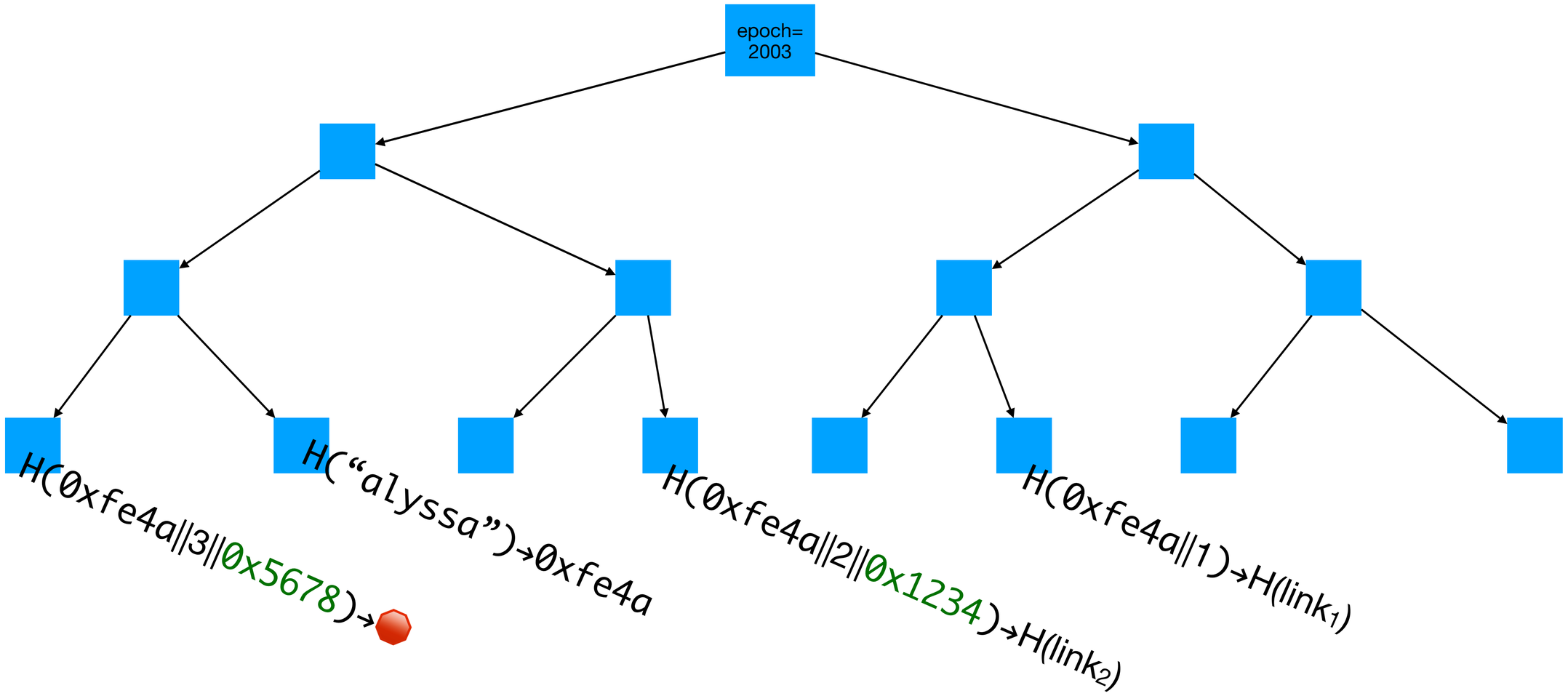
Transparency Trees Redux

User “Sigchain” Take 2



Transparency Tree

Full Location Hiding



Transparency Tree

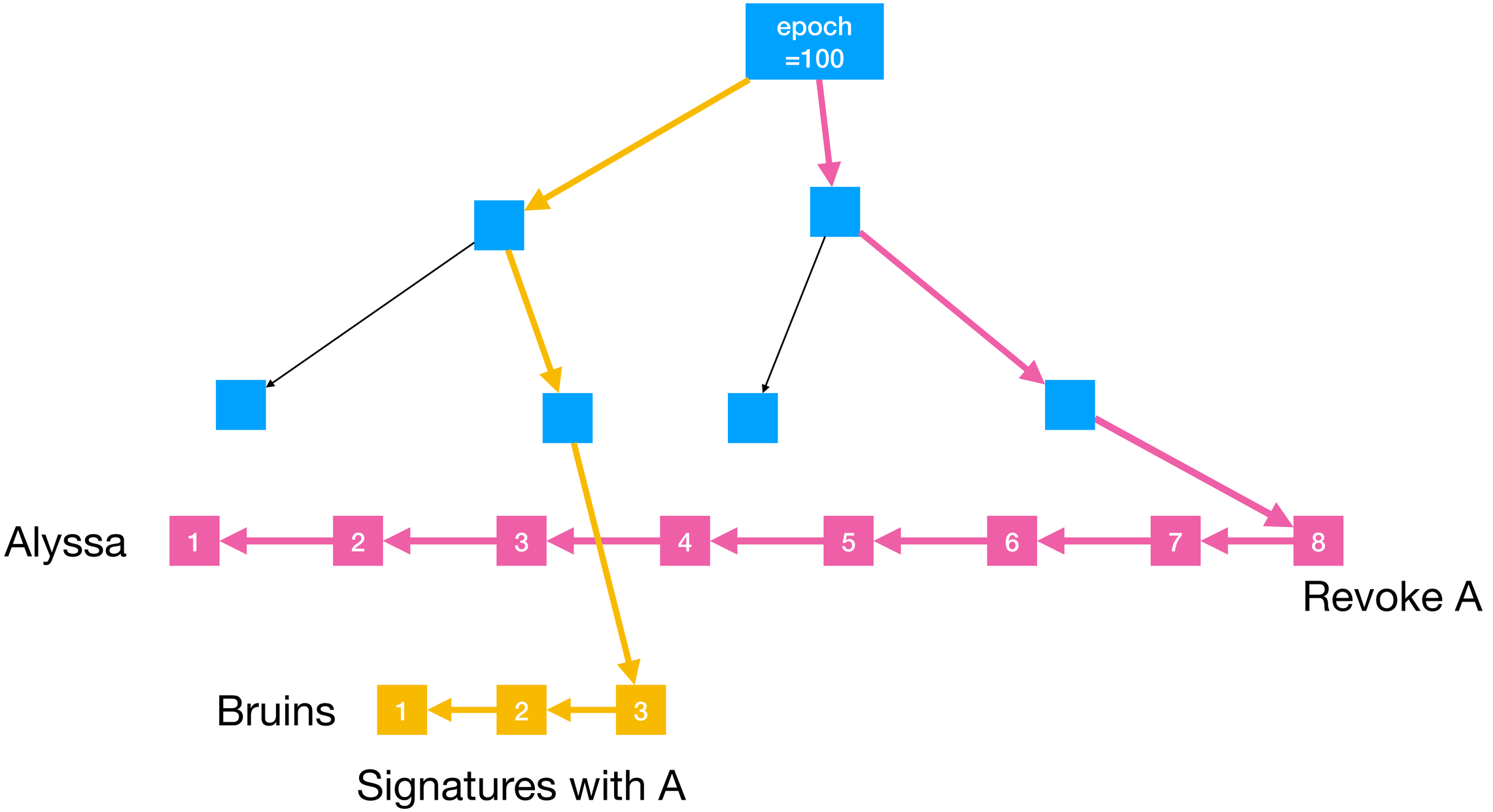
Signature Ordering

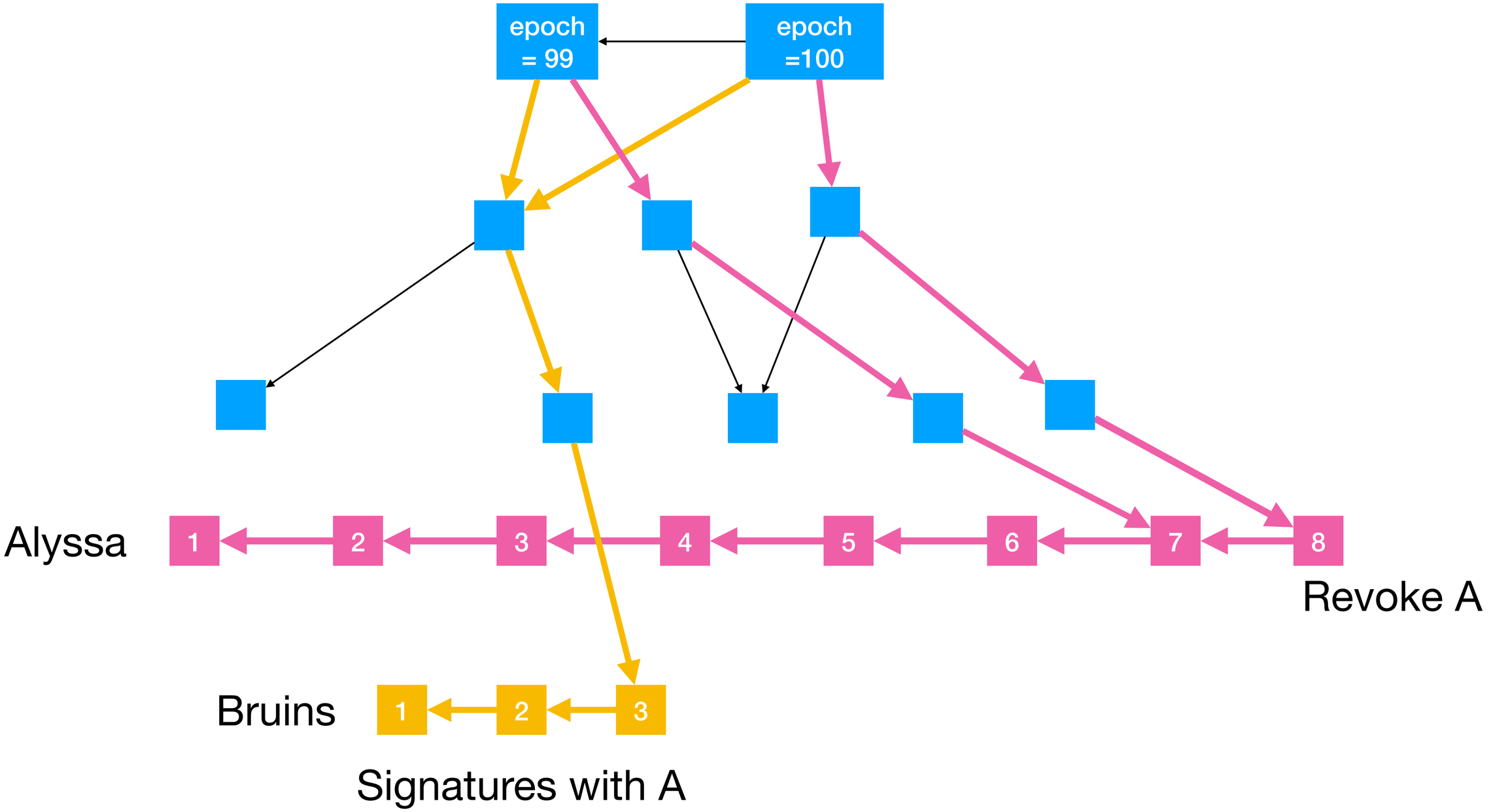
- Problem: Alyssa signs a signature with key A time t_1
  - Revokes A at time $t_2 > t_1$
  - Bob verifies the signature at time $t_3 > t_2$
- Question: how can Alyssa prove to Bob that the signature happened before the revoke?
  - Imagine a bad guy finds Alyssa's lost, revoked phone, guesses the PIN, accesses the secret key, signs a *back-dated* signature and colludes with the server to accept it.

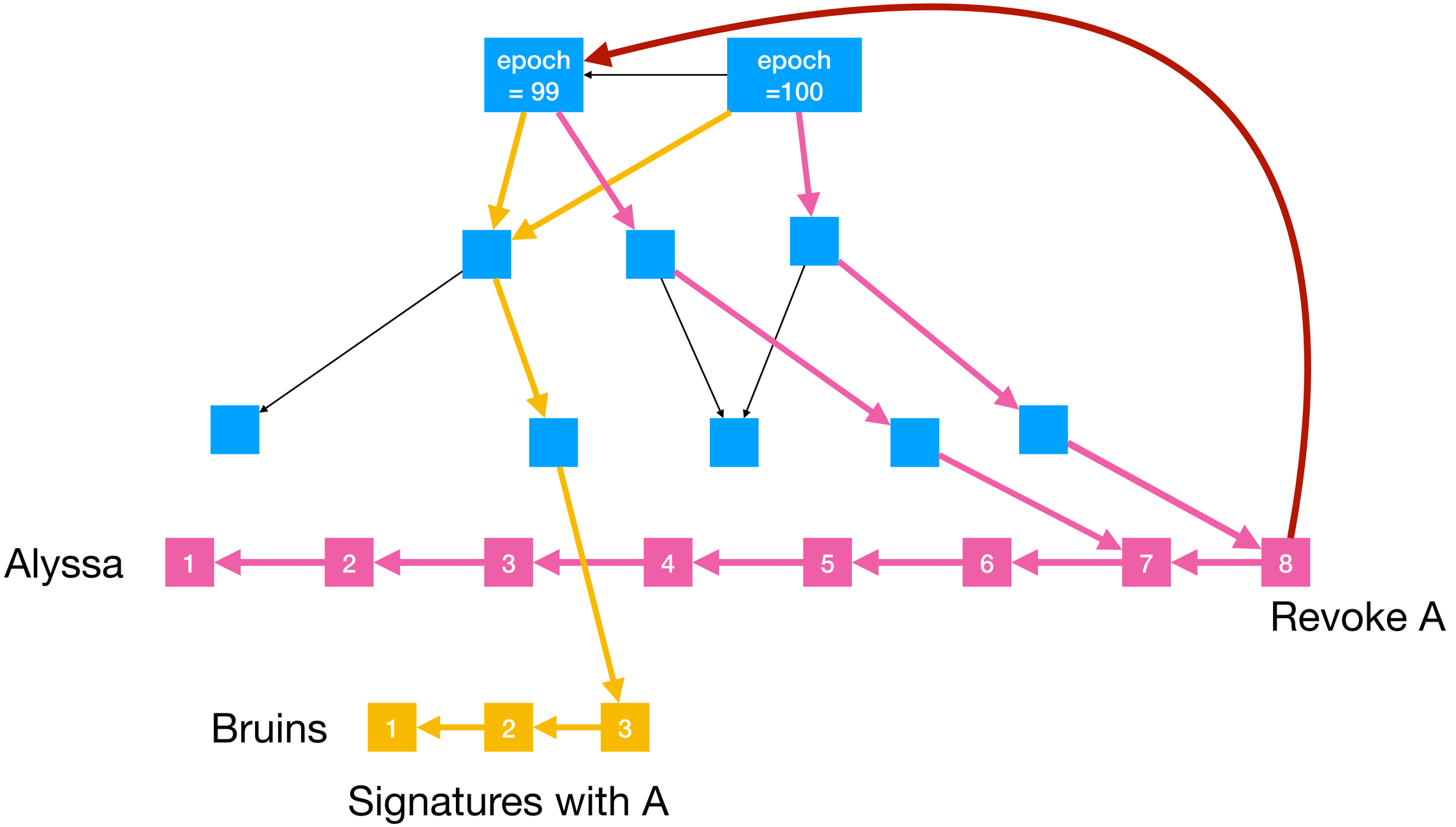
Transparency Tree

Signature Ordering

- For signatures in Alyssa's sigchain, the revoke signs over the hash of all previous links, thereby proving the revoke happened after earlier signatures
- For Alyssa's signatures on a team (like "Bruins"), no way to order the two events.
  - Perhaps a malicious server, with the captured client, intends to add a new member to Bruins and claim it happened before the revoke.


epoch
=100
Alyssa
1
2
3
4
5
6
7
8
Revoke A
Bruins
1
2
3
Signatures with A


epoch
= 99
epoch
=100
Alyssa
1
2
3
4
5
6
7
8
Revoke A
Bruins
1
2
3
Signatures with A


epoch
= 99
epoch
=100
Alyssa
1
2
3
4
5
6
7
8
Revoke A
Bruins
1
2
3
Signatures with A

Outline

✓ ~~Motivation~~

✓ ~~Threat model~~

✓ ~~Devices & Users~~

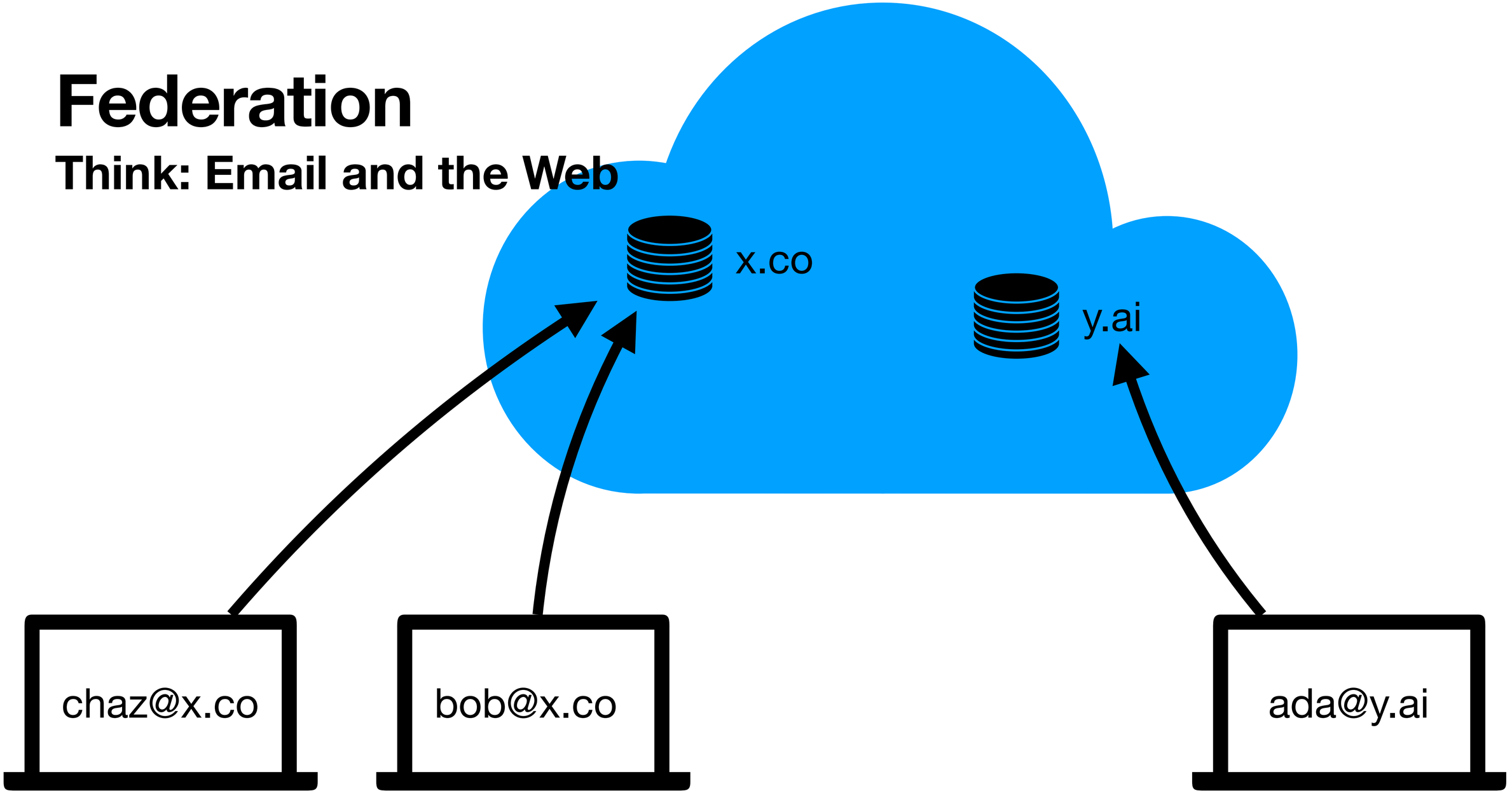
✓ ~~Teams~~

✓ ~~Transparency Tree (redux)~~

- Federation

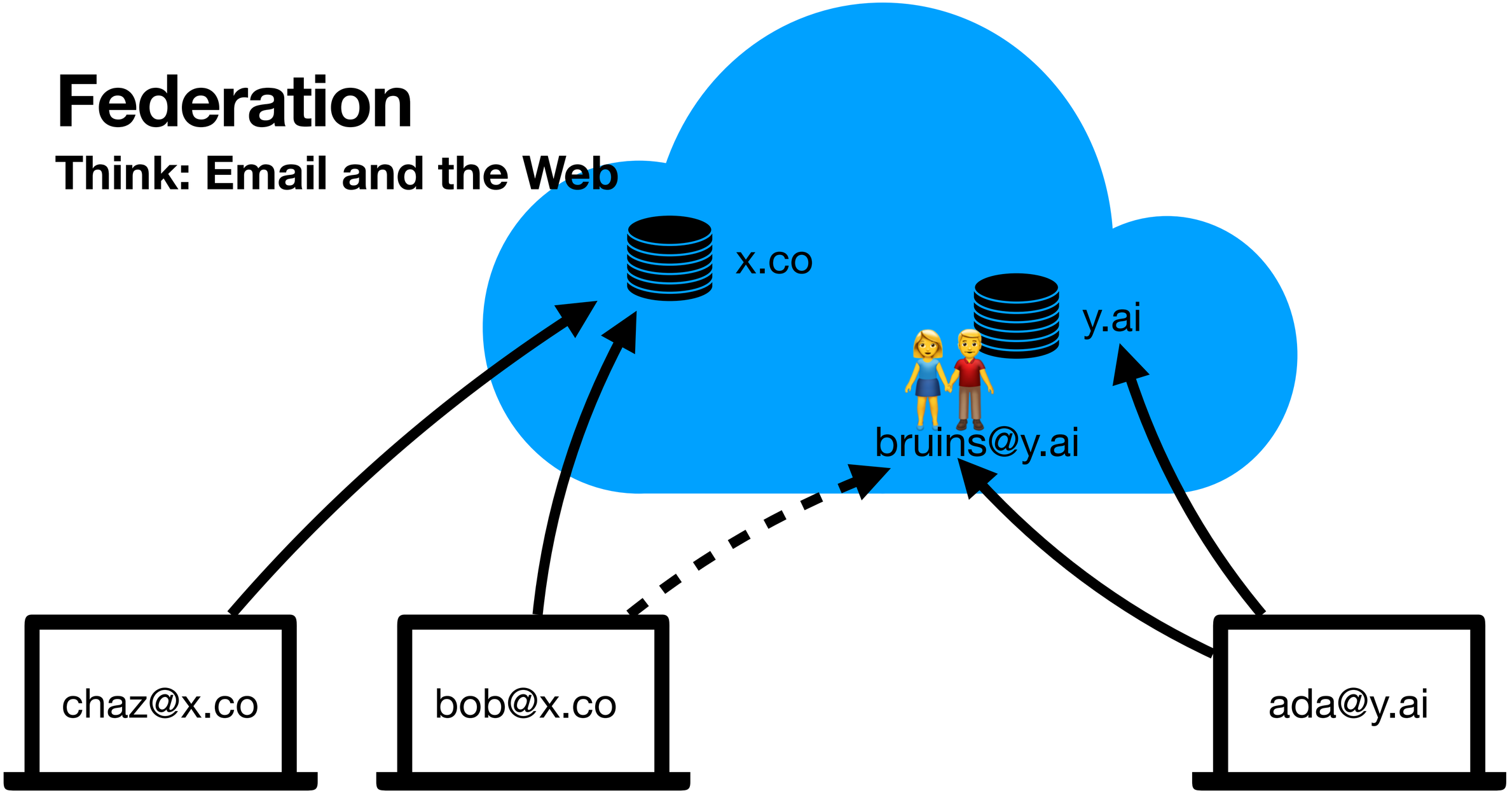
Federation

Think: Email and the Web



Federation

Think: Email and the Web



Federation

Why bother?

Federation

Why bother?

- “Vendor lock-in” scenario (think: Facebook vs. early Web)
- Different vendors might have different costs, policies on privacy, posture toward LE
  - Might have different uptime guarantees
  - A market of possibilities for various trade-offs
- With self-hosting, no external third-party dependencies (think: DataDog or AWS)

Federation

New complexities

- Server agility:
  - Need a mechanism to upgrade server keys or change server DNS name
  - Keybase would just upgrade the client
- Cyclical dependencies between teams, and how to reach a stable key distribution:
  - No “lock the world” possibility across federation
- Upgrades become a lot harder: need a good story for migration at multiple layers of the stack

Summary

- ✓ Motivation
- ✓ Threat model
- ✓ Devices & Users
- ✓ Teams
- ✓ Transparency Tree (redux)
- ✓ Federation

https://foks.pub

Q&A?