SCIENCE
PASSION
TECHNOLOGY

Enhancing Security Through Transparency

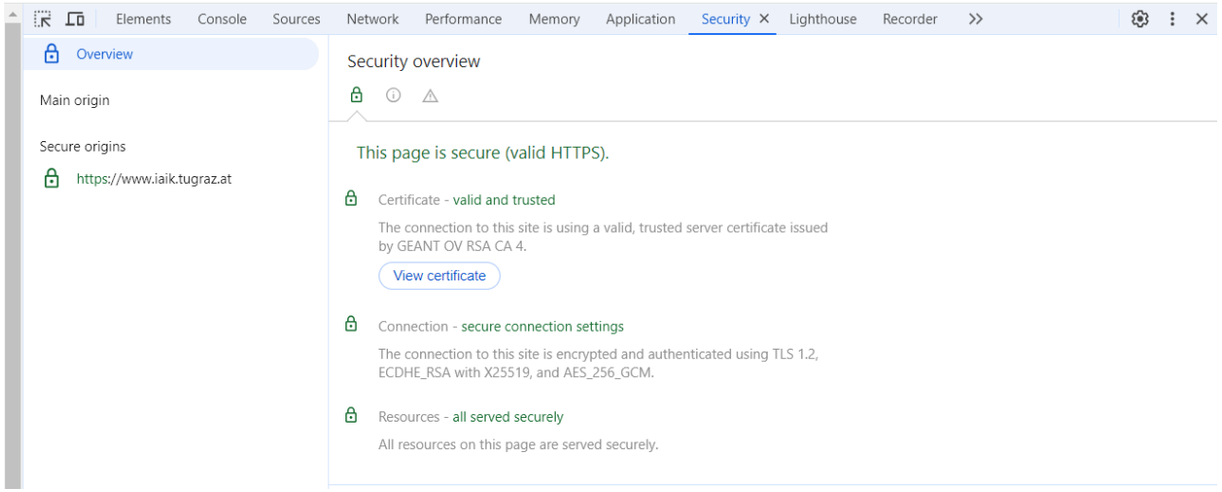
SEAD 2025

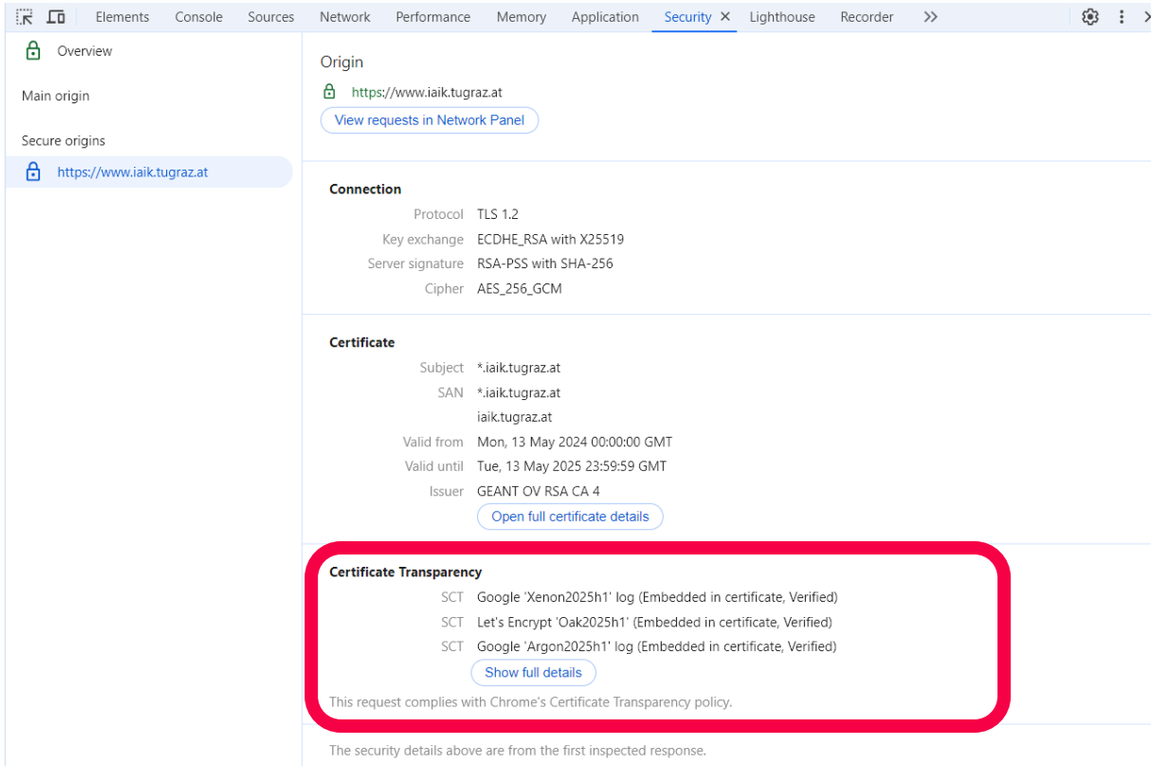
Edona Fasllija
May 8, 2025

edona.fasllija@tugraz.at

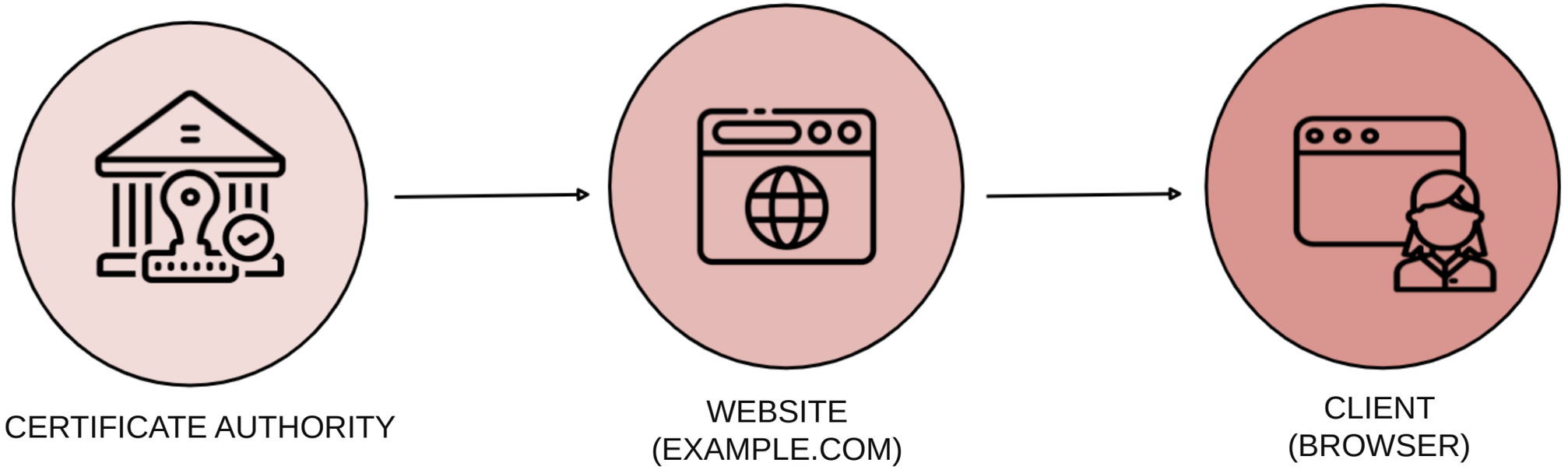
How is all of this related to you?

Browsing the Web





Web PKI



Icons: Flaticon.com

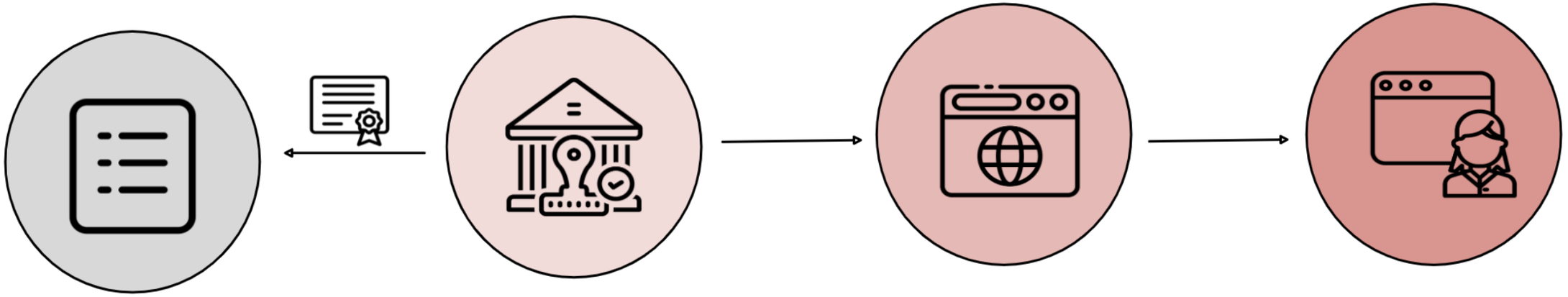
Malicious Certificates

ROGUE CERTIFICATE AUTHORITY

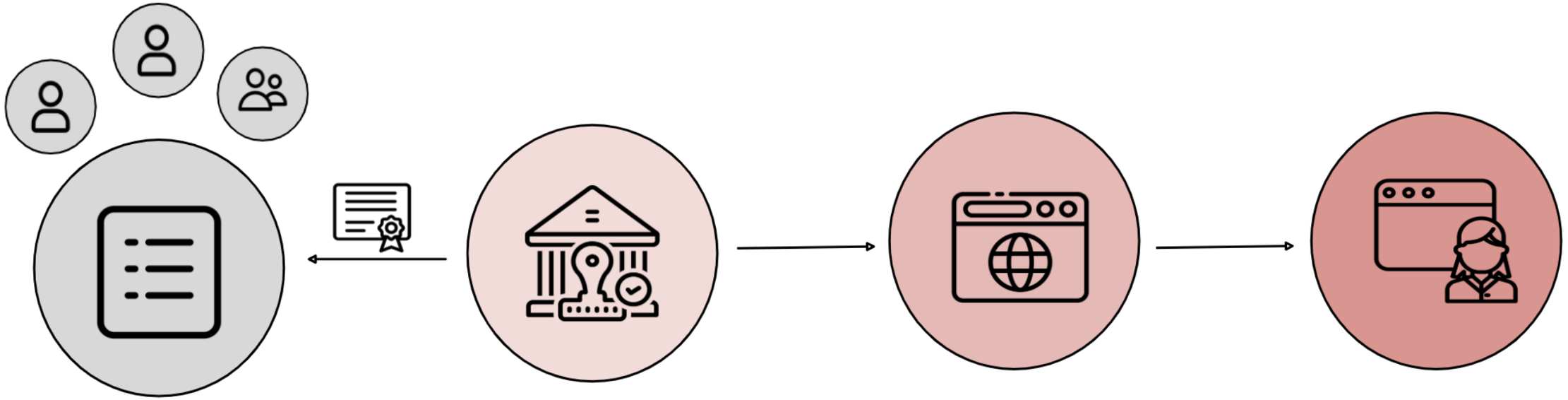
WEBSITE (EXAMPLE.COM)

CLIENT (BROWSER)

Certificate Transparency (CT)



Certificate Transparency (CT)



LOG SERVER

CERTIFICATE AUTHORITY

WEBSITE
(EXAMPLE.COM)

CLIENT
(BROWSER)

“Transparency Technology provides accountability for data.”

F. Valsorda. Modern transparency logs (RWC 2024)

Transparency-Enabled (Verifiable) Systems

- *Retain* the *trusted* authority
- *Anyone* can check that it’s not *misbehaving*
- *Add* on a ***transparency*** layer
- *Maintain* the seamless user experience

Primitives

The ‘root’ of truth for your data

Transparency Log : just a tamper-evident append-only list

- List
- Append-Only
- Tamper-evident

[foo, bar, baz]

[foo, bar, baz, **qux**]

[**cat**, bar, baz,qux]
[bar, baz, qux]

Requirements:

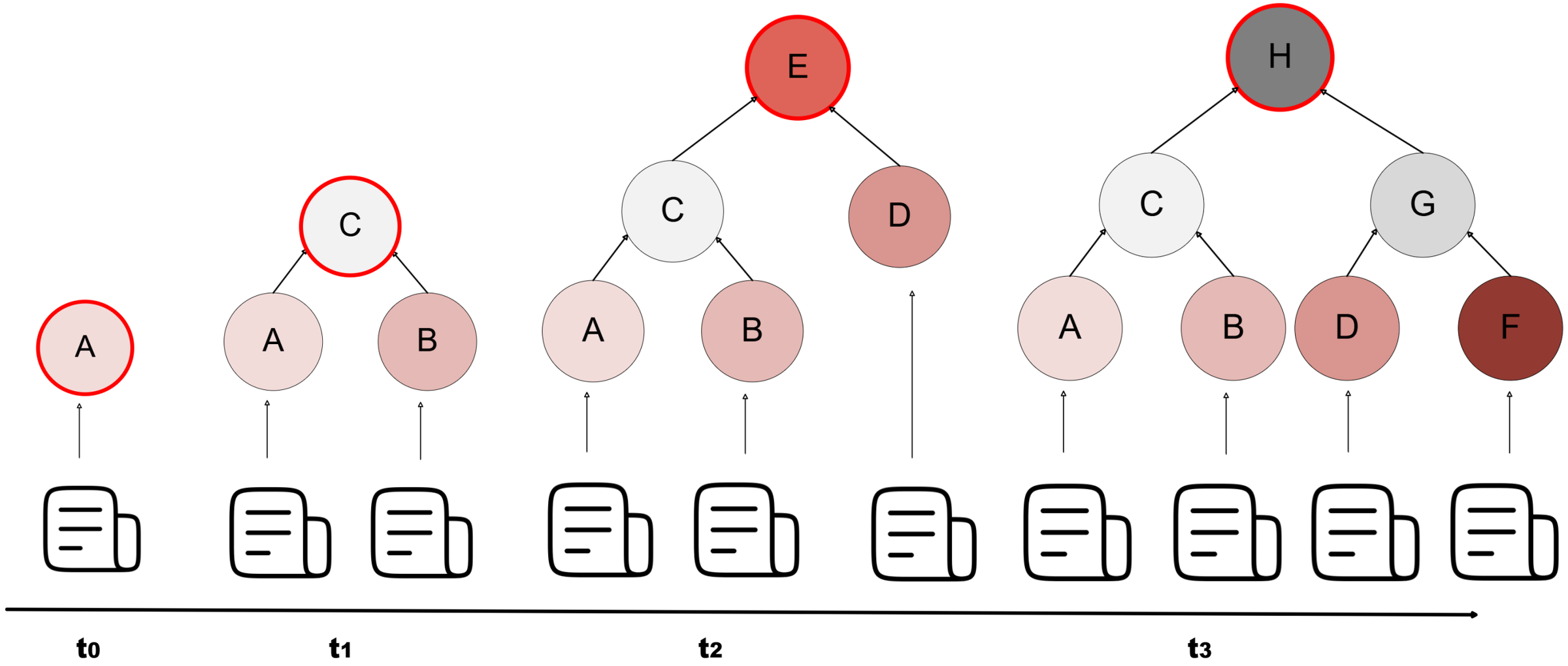
- Efficient *proof* that a record is *in* the log
- Efficient *proof* that an earlier log is a *prefix* of the current log
- Efficient *iteration* over the records in the log

✓ Browsers verify that a certificate is recorded in a log

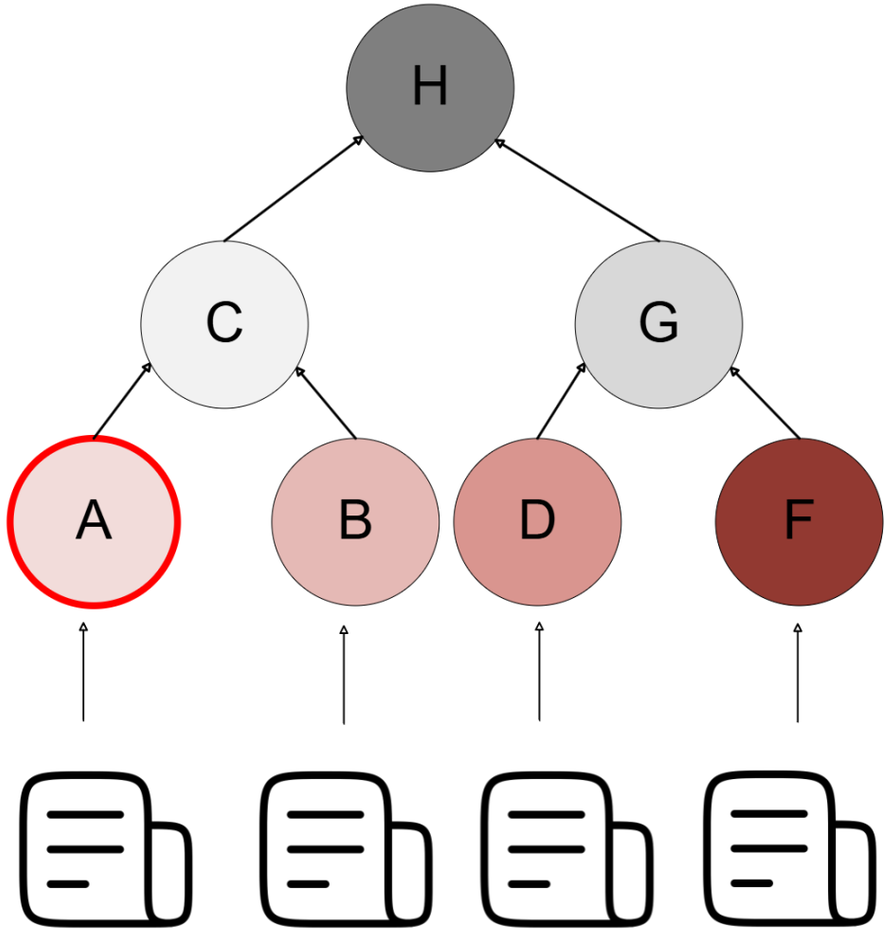
✓ A certificate can not disappear from the log undetected

✓ Anyone can scan the log to detect *misissued* certificates

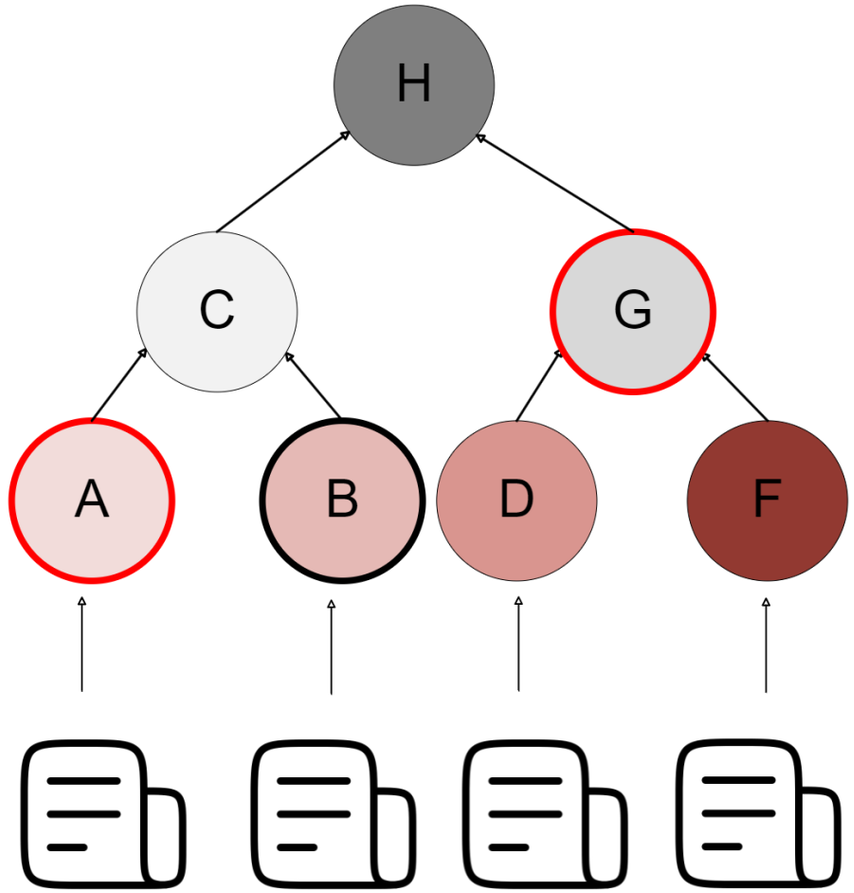
Merkle Tree



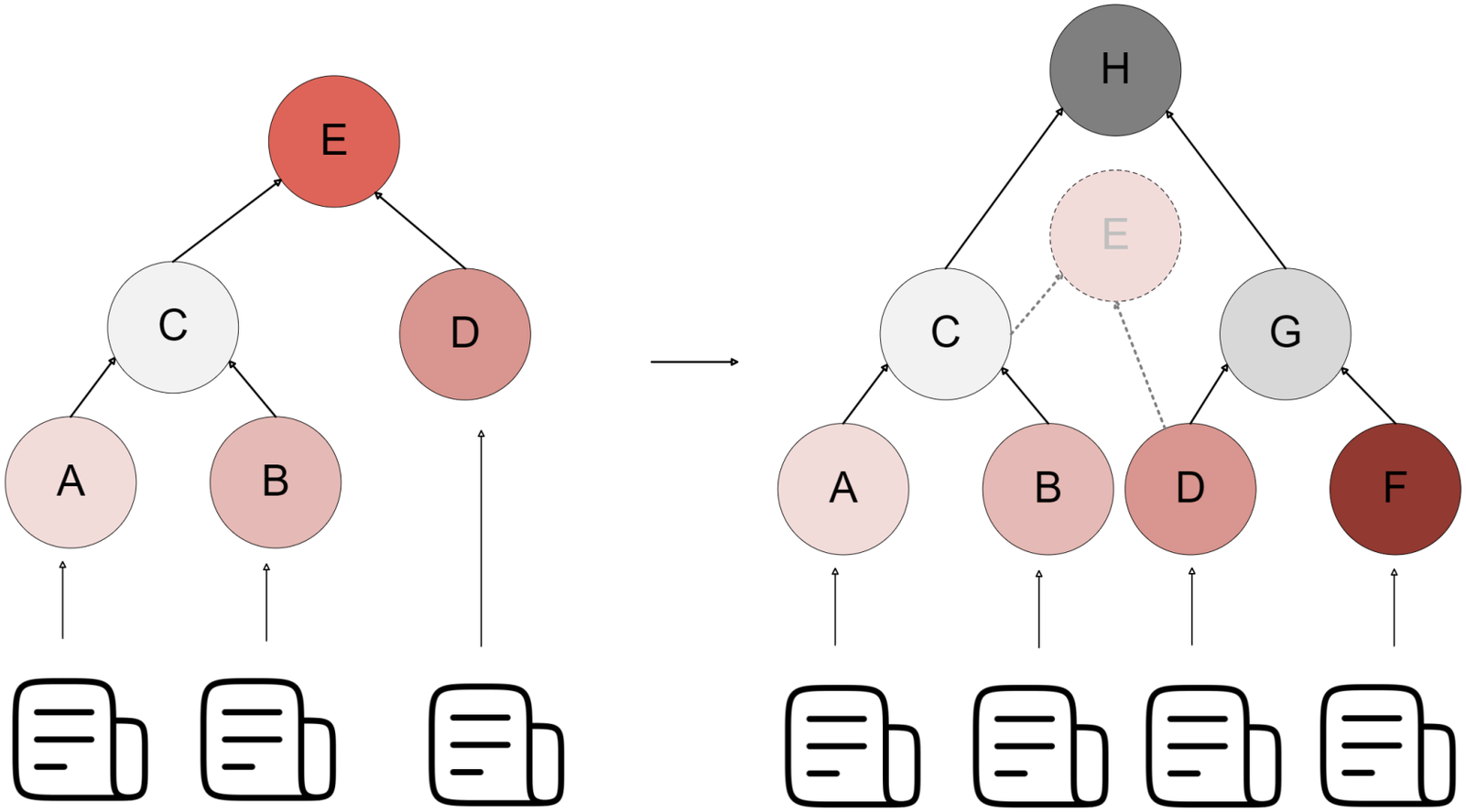
Inclusion Proof / Proof of Membership



Inclusion Proof / Proof of Membership

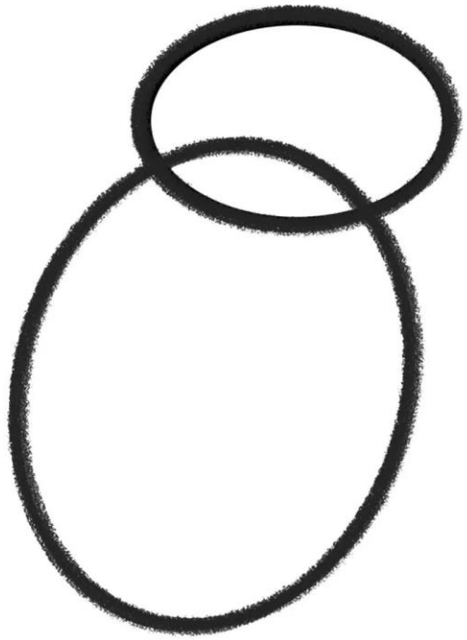


Consistency Proof / Proof of Append-Only



Transparency Ecosystems

How to design a Transparent Ecosystem



Step 1: Draw some circles

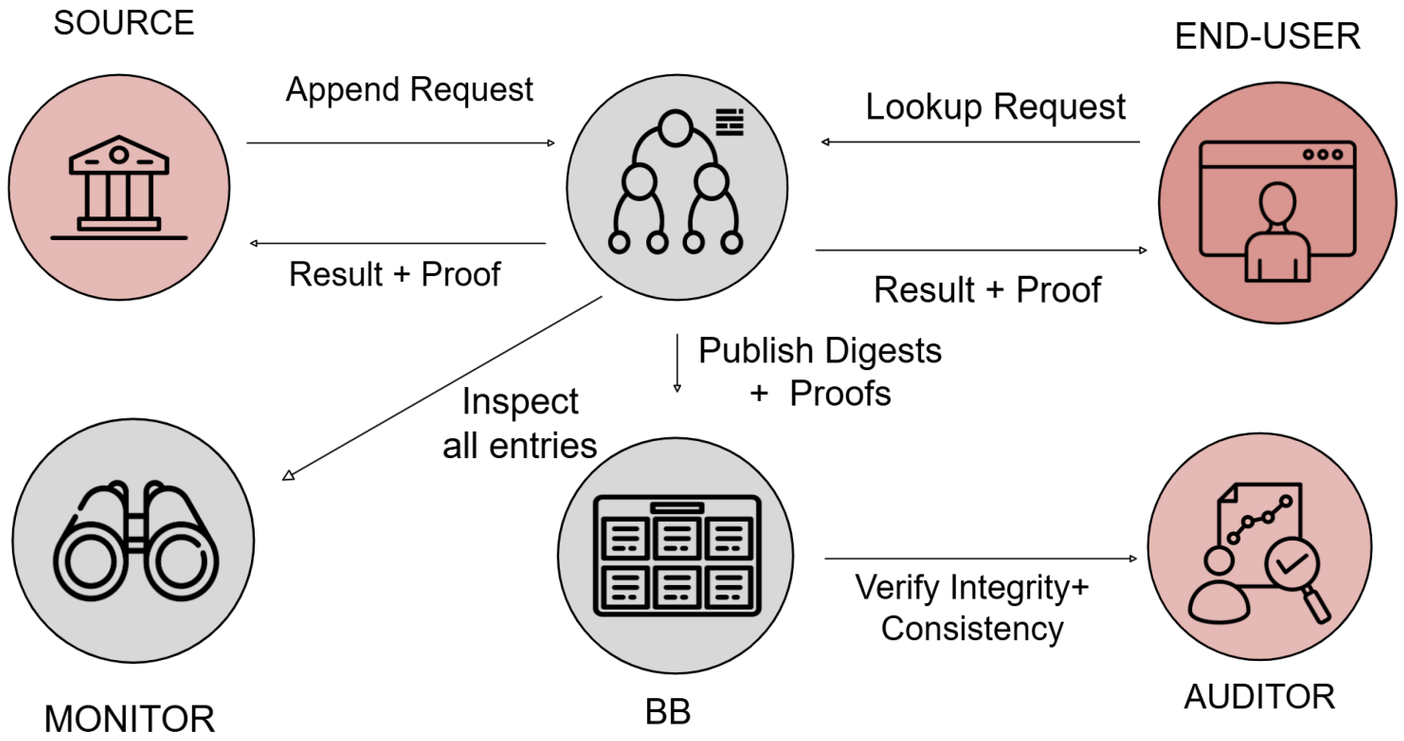
Step 2: Draw the rest of the owl!

https://timoelliott.com/blog/2023/05/why-ai-is-like-drawing-an-owl.html

Missing Pieces: End-to-end Transparent Systems

- Auditing
  - Point-in-time and global
  - Can be done by *anyone*
- Monitoring
  - Synchronous
  - Per-user
- Split-view Protection

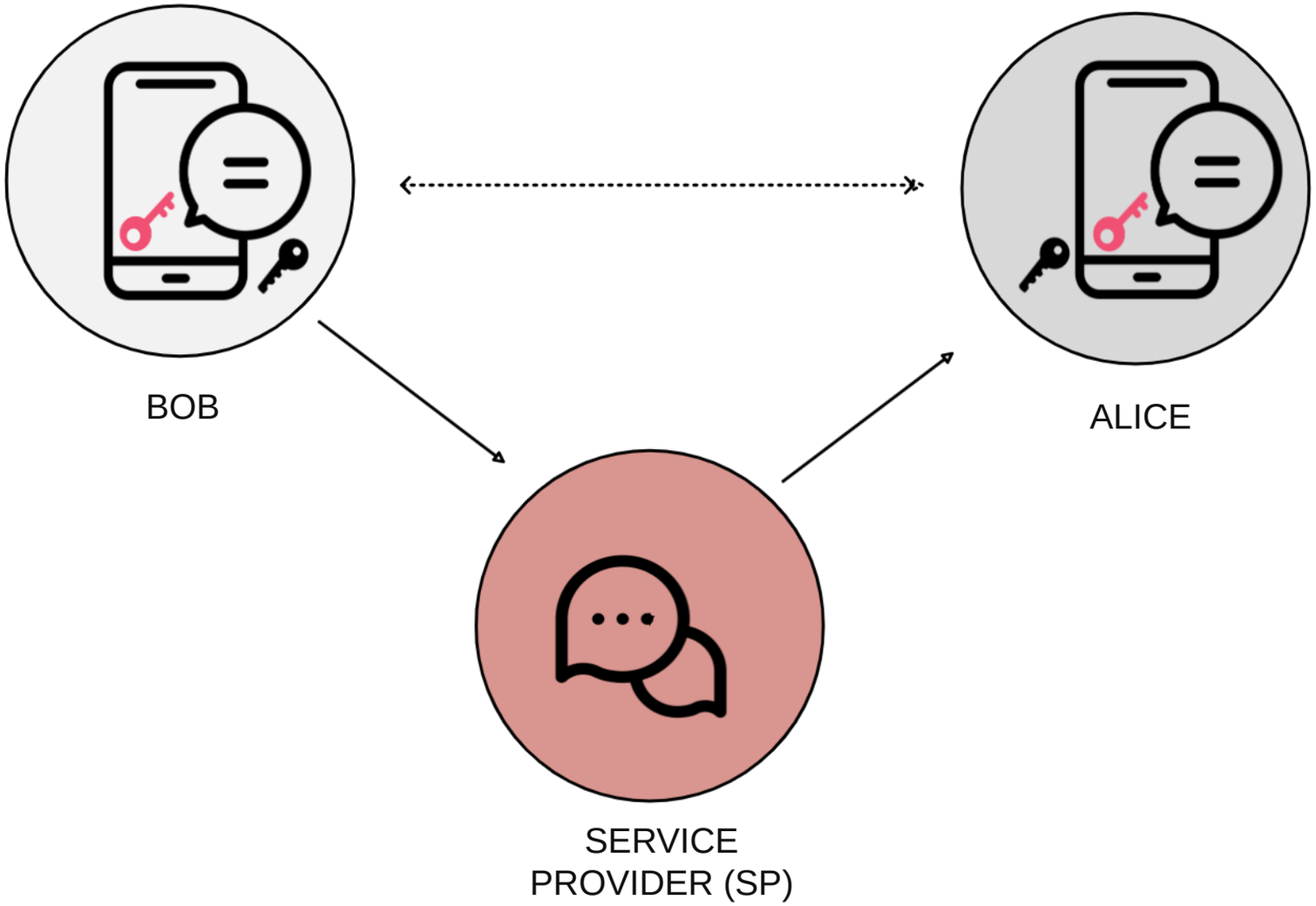
Transparency Ecosystems



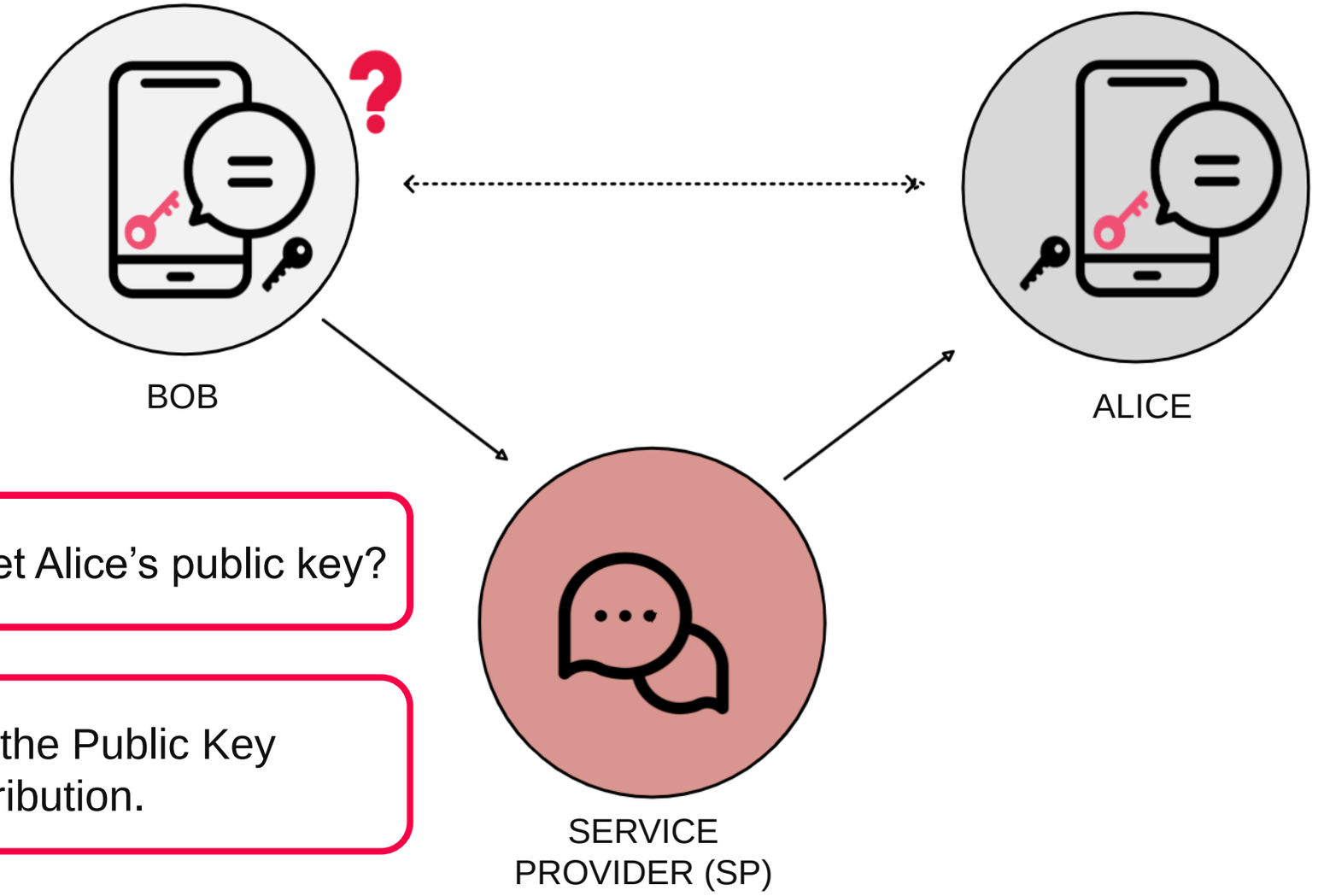
Other Transparency Systems

Why Making Key Management Transparent is So Challenging

End-to-end Encrypted Messaging



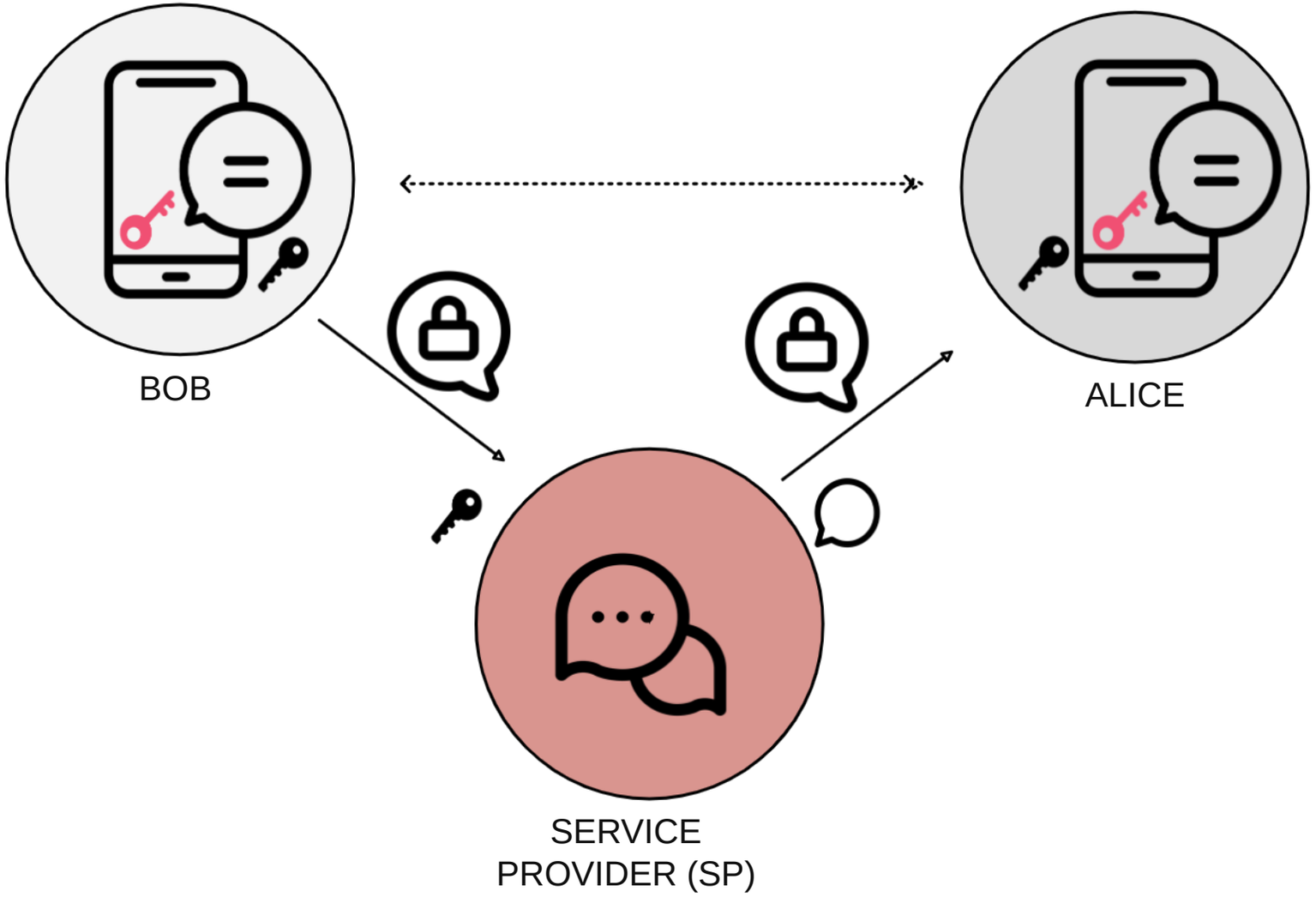
End-to-end Encrypted Messaging



How does Bob get Alice's public key?

The SP manages the Public Key Directory and distribution.

Undetectable Man-in-the-middle



Contact verification

QR Code Scanning:

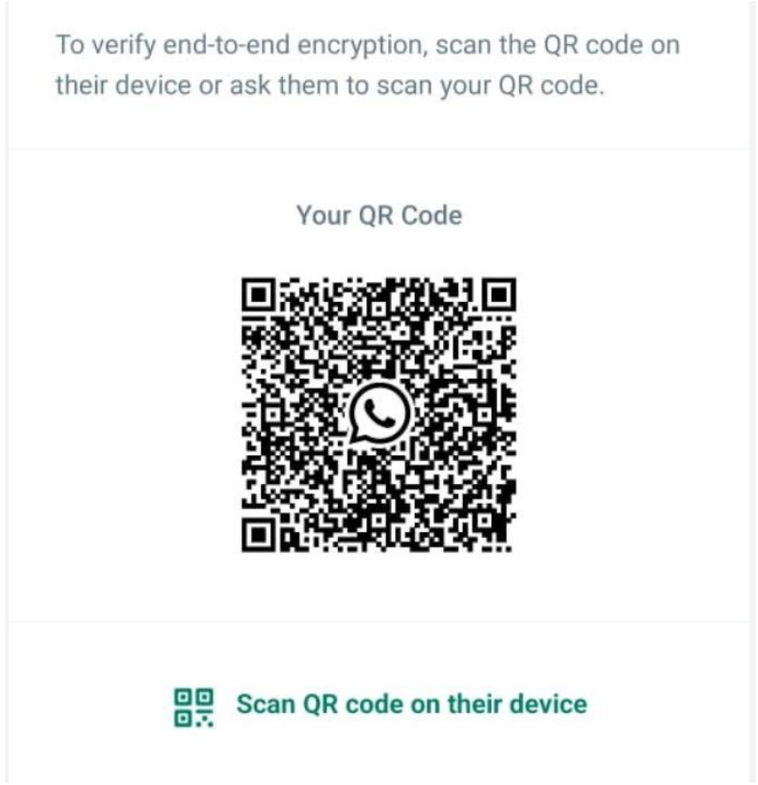
- Physical proximity required
- Rerun verification with *all contacts* when a user's key changes

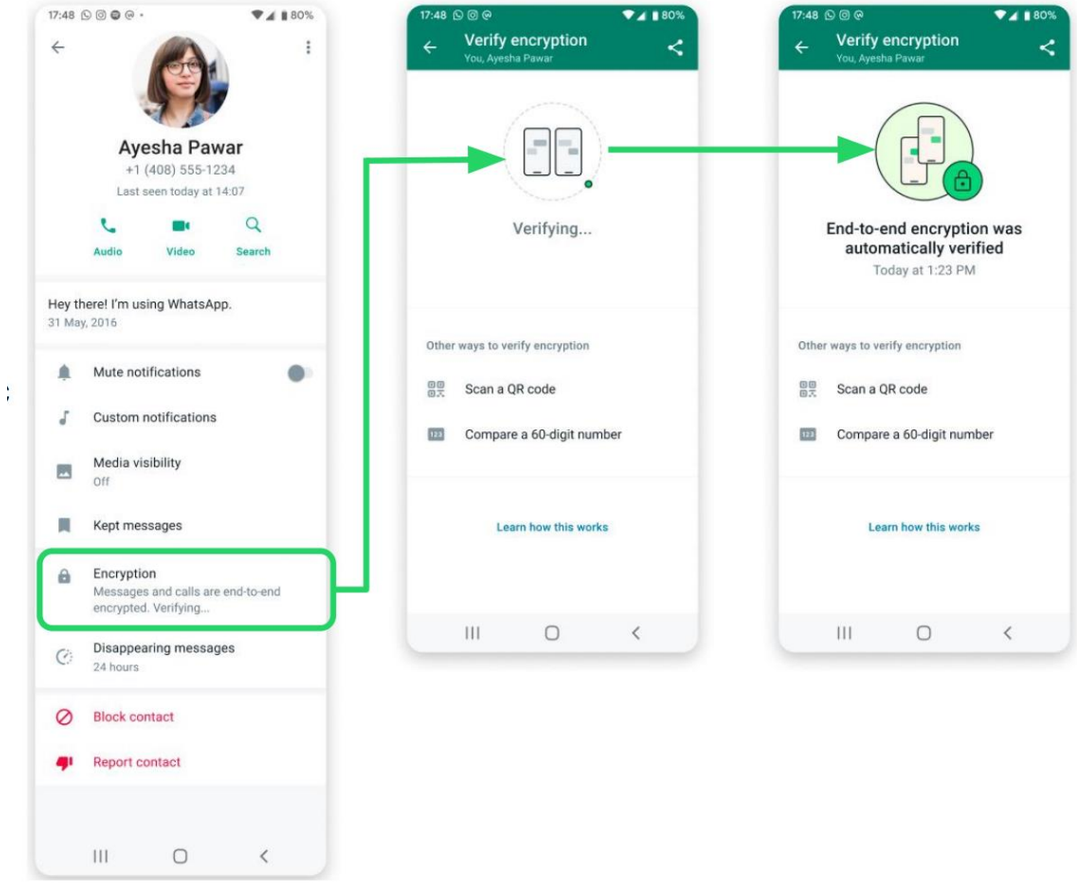
Your key changes whenever you:

- switch to a new phone
- factory-reset a phone
- uninstall and reinstall the app

End-to-End Encrypted Messaging

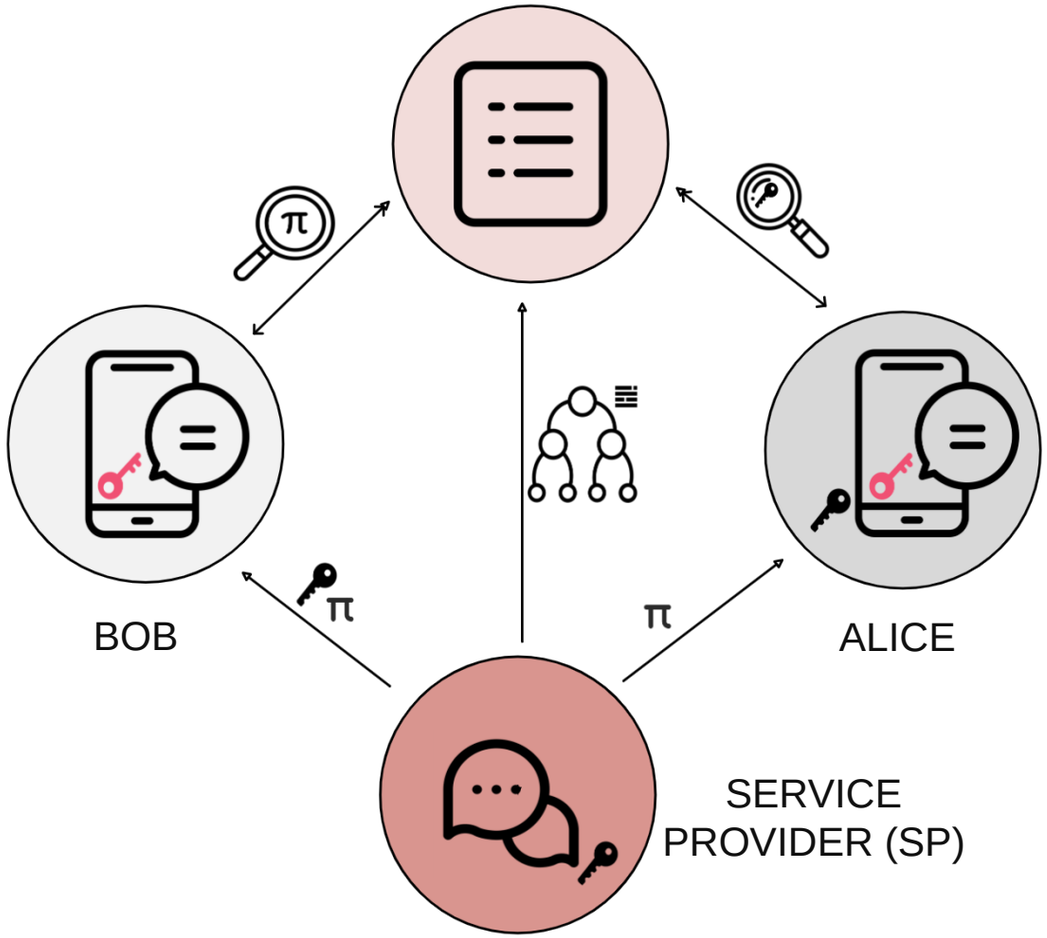
Contact Verification





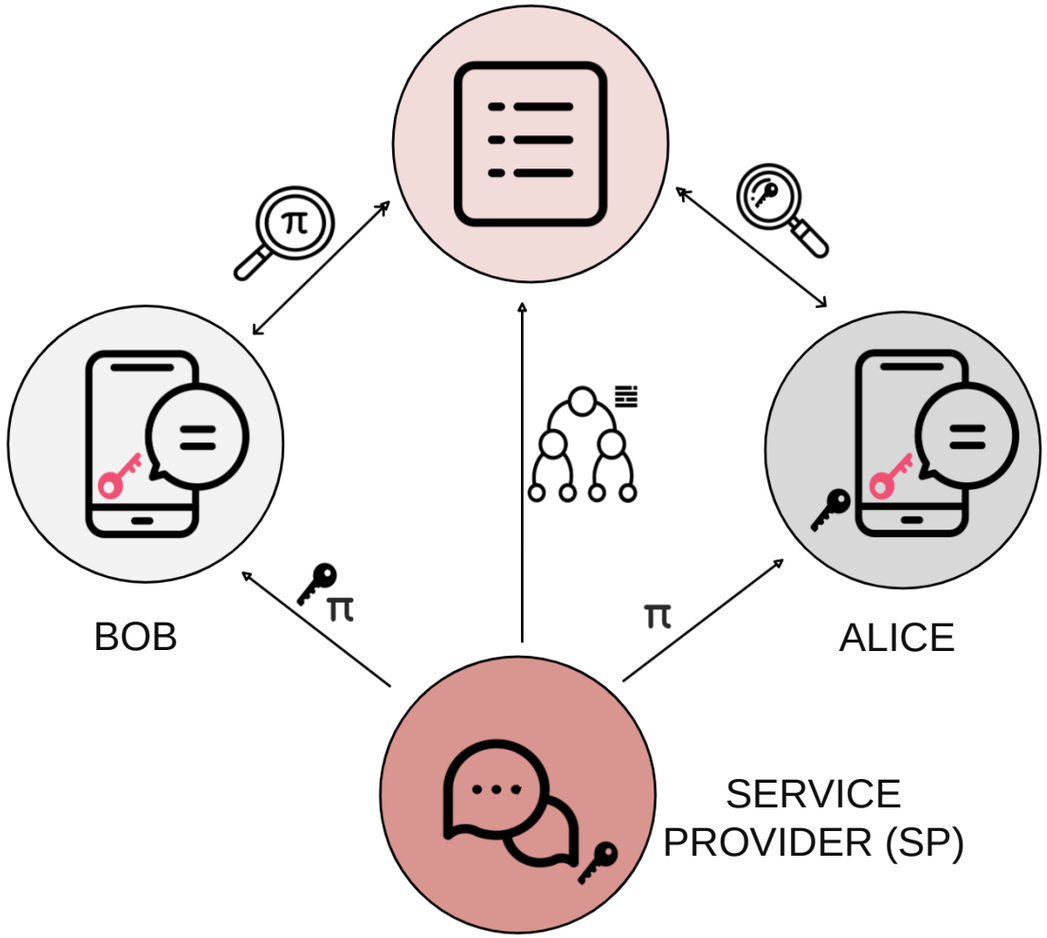
WhatsApp Key Transparency Overview. White Paper.

Key Transparency (KT)



- The SP regularly posts *commitments* to Key Directory.
- Bob queries Alice's PK and gets *proof* that it is correct.
- Alice's device in the background regularly *monitors* her key w.r.t. the commitments.

Key Transparency (KT) - Requirements



- Directory:
  - ***Map*** usernames ↦ public keys
- Publish *commitments* to a directory
- *Proof* that the returned pk is ***correct***.

Logs vs Maps

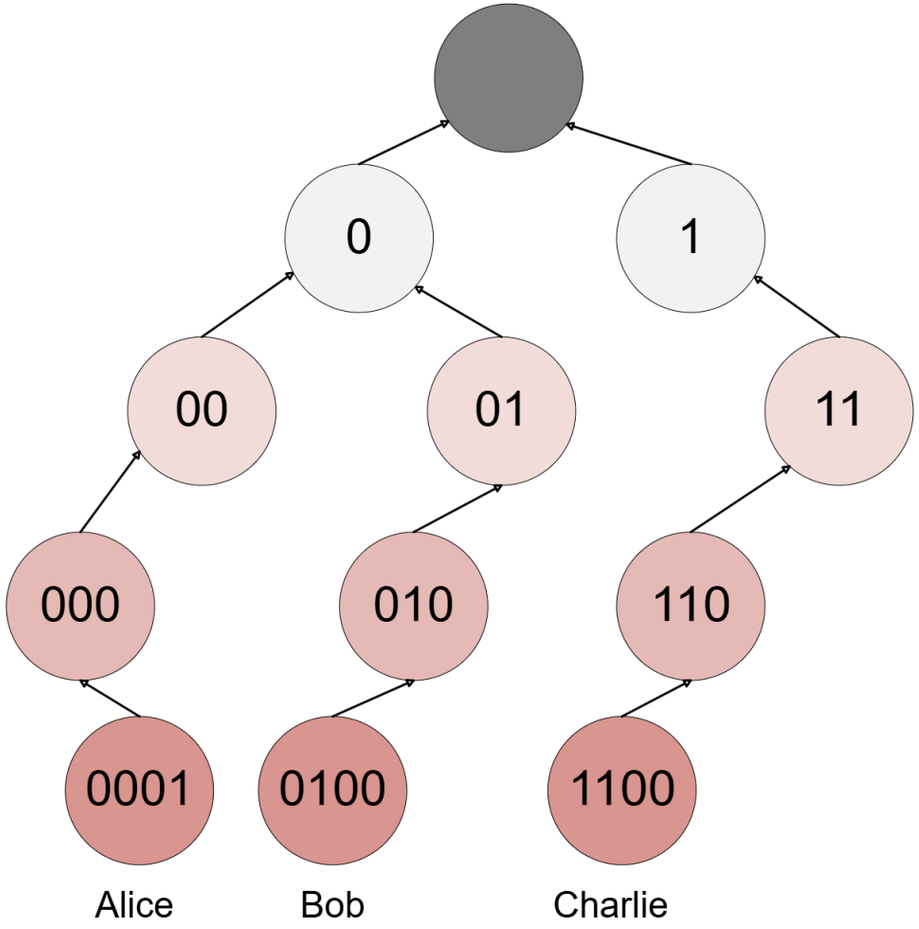
Logs

- History Tree
  - Grows from left to right
  - Efficient append-only proofs
  - Not so efficient to lookup specific entries

Map

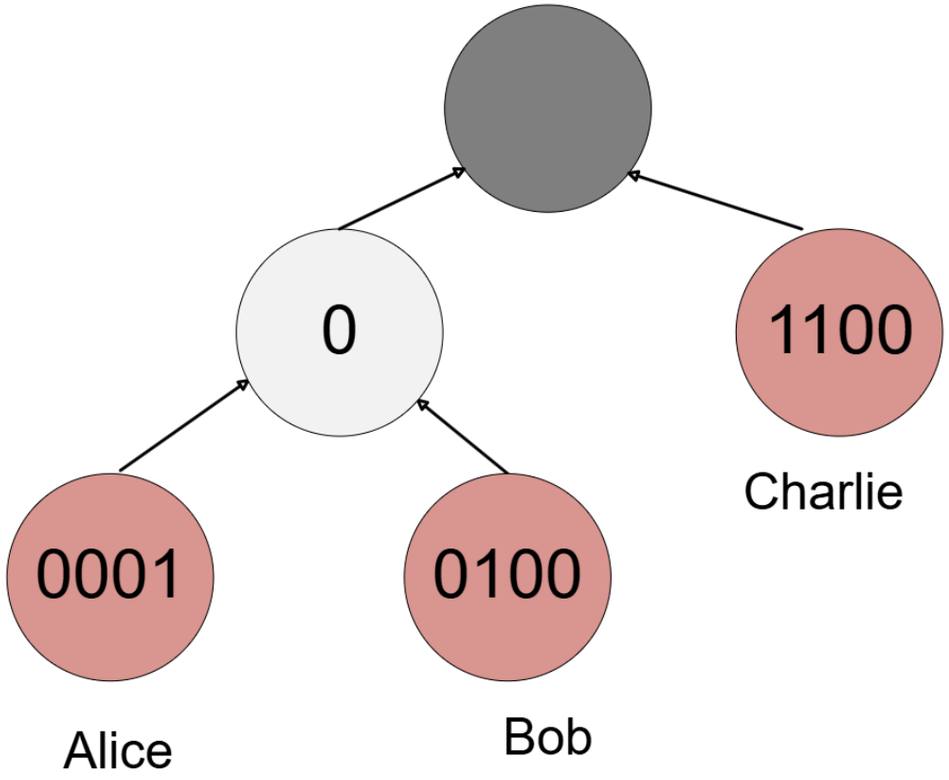
- Prefix Tree
  - (Key, Value) Store
  - Lexicographical order of leaf nodes
  - Efficient to lookup values

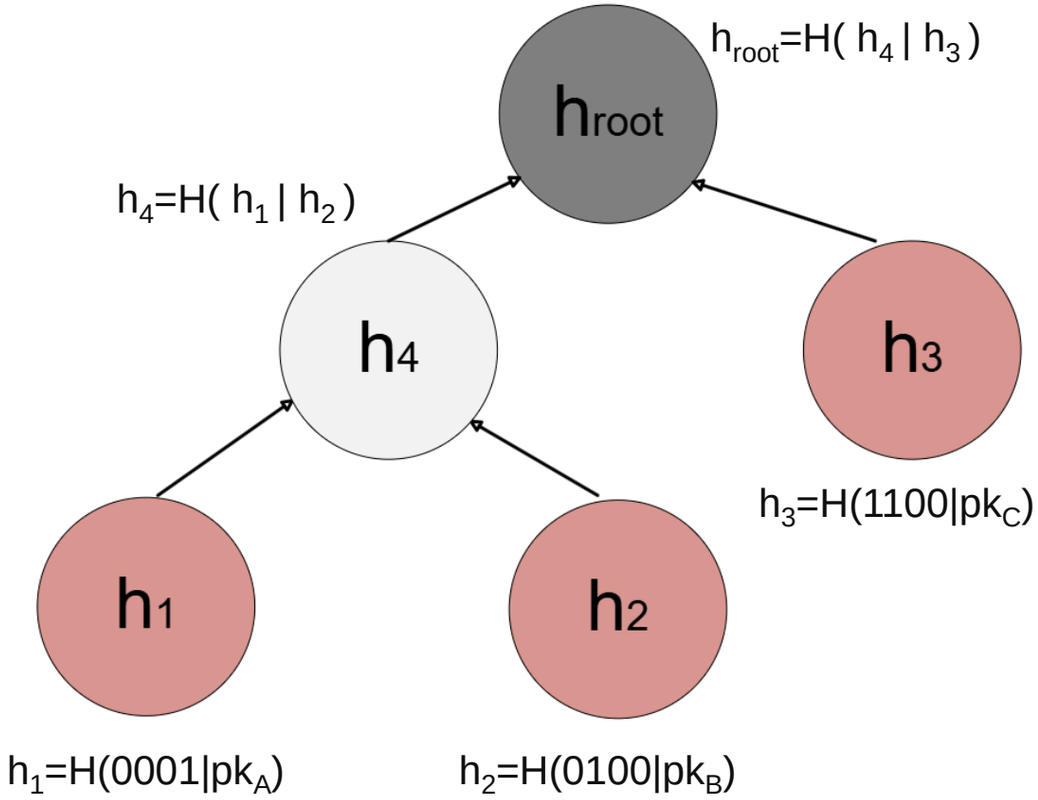
Verifiable Maps – Sparse Merkle Trees



- Dictionary:
  - {(Alice, pk_A), (Bob, pk_B), Charlie, pk_C)}
  - H(Alice) = 0011
  - H(Bob) = 0100
  - H(Charlie)=1100

Sparse Merkle Tree Construction

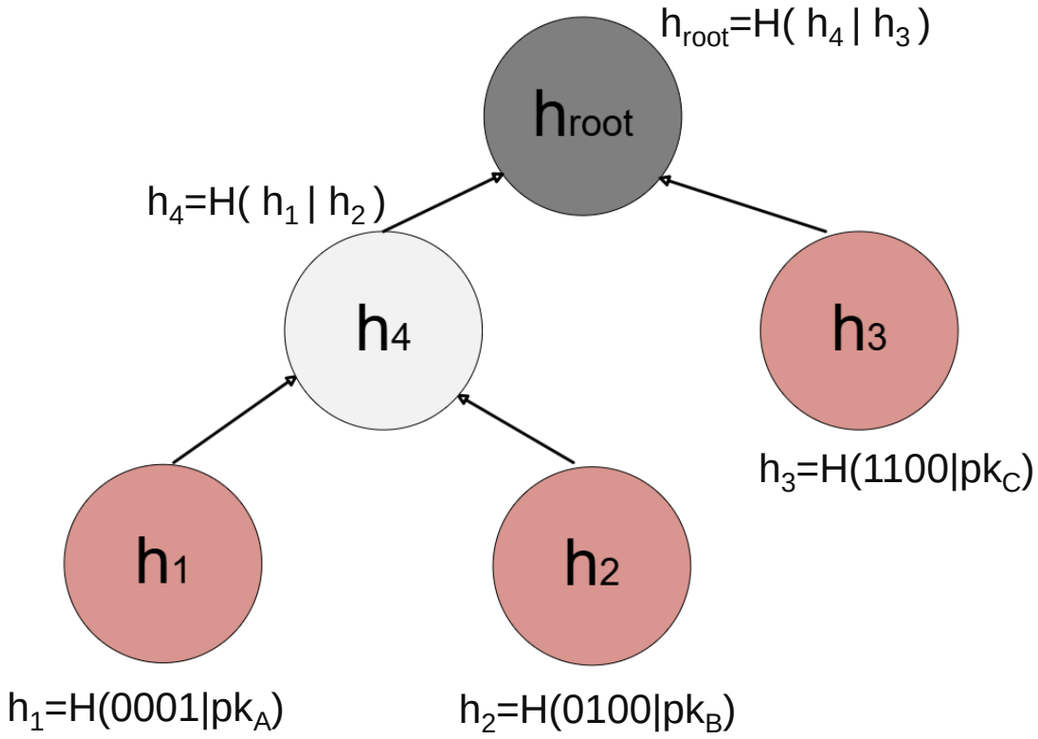




Approach limitations:

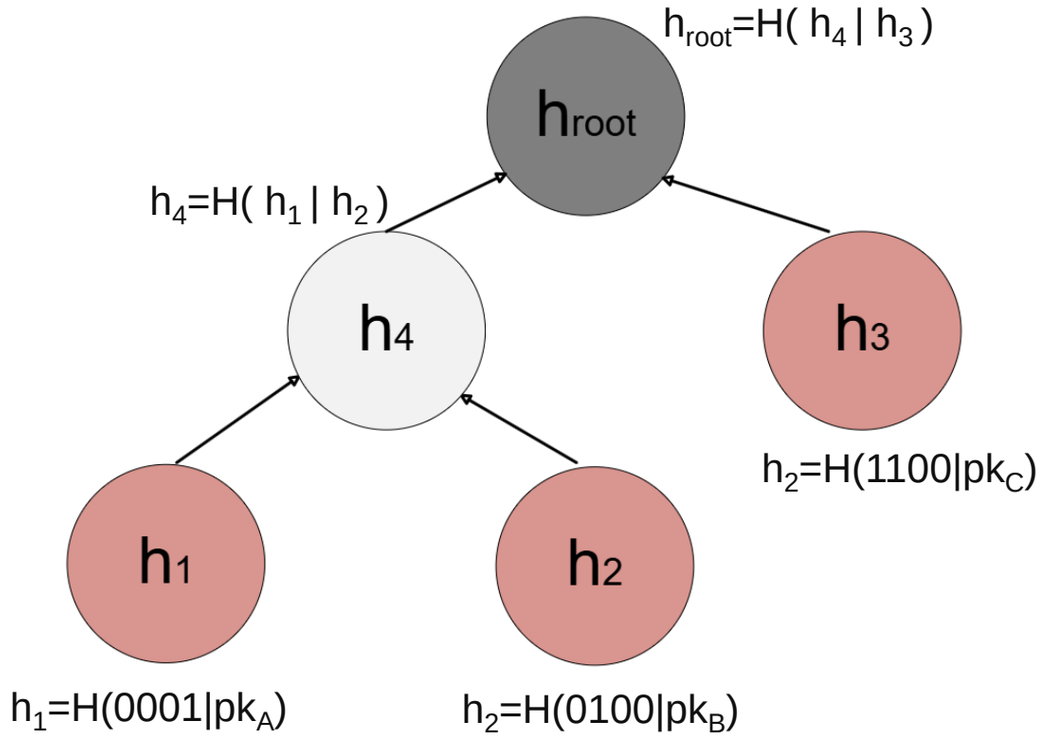
Privacy:

- Brute force search reveals usernames
  - Phone numbers, email addresses
- Reveals when keys are updated
- Even when not directly queried



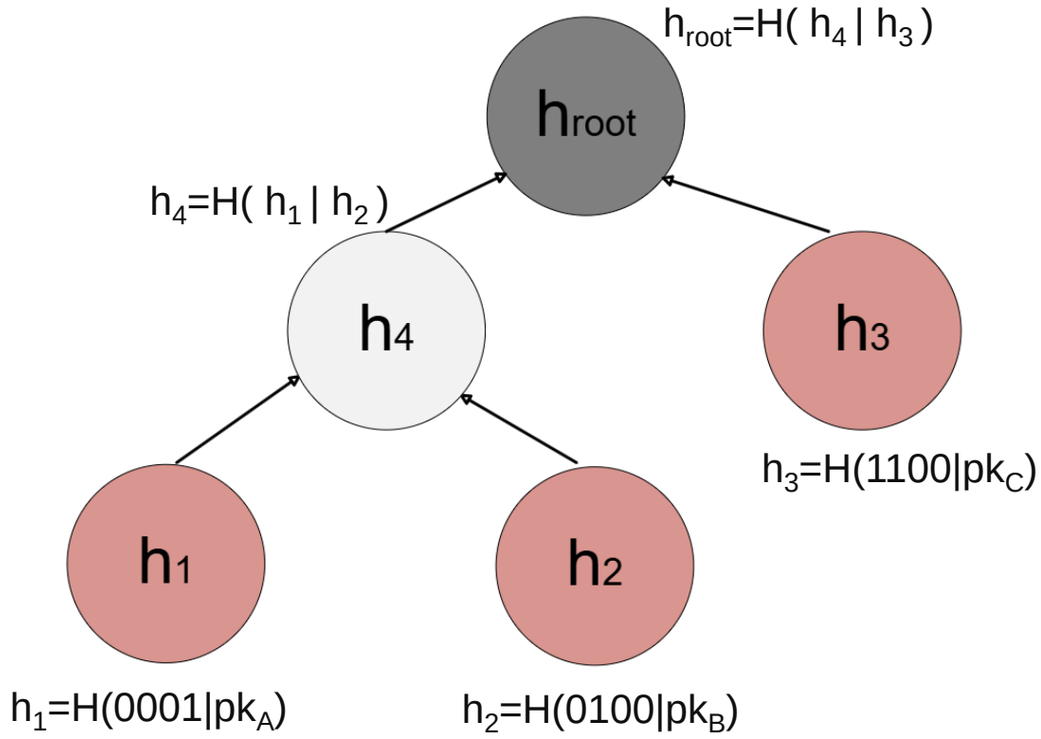
Approach limitations:

- Dictionary:
  - {(Alice, pk_A), (Bob, pk_B), Charlie, pk_C)}
  - H(Alice) = 0011
  - H(Bob) = 0100
  - H(Charlie)=1100
  - H(Mallory)=0011
- Querying Alice and Bob ↦ Mallory is not in the database
- Querying Bob reveals Alice's key updates

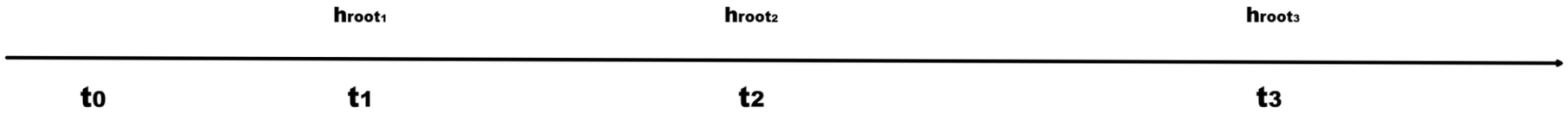


Masking usernames with VRFs

- Dictionary:
  - {(Alice, pk_A), (Bob, pk_B), Charlie, pk_C)}
  - VRF_k(Alice) = 0011
  - VRF_k(Bob) = 0100
  - VRF_k(Charlie)=1100
  - VRF_k(Mallory)=?
- SP needs to prove the VRF is computed correctly.



Periodic Publish

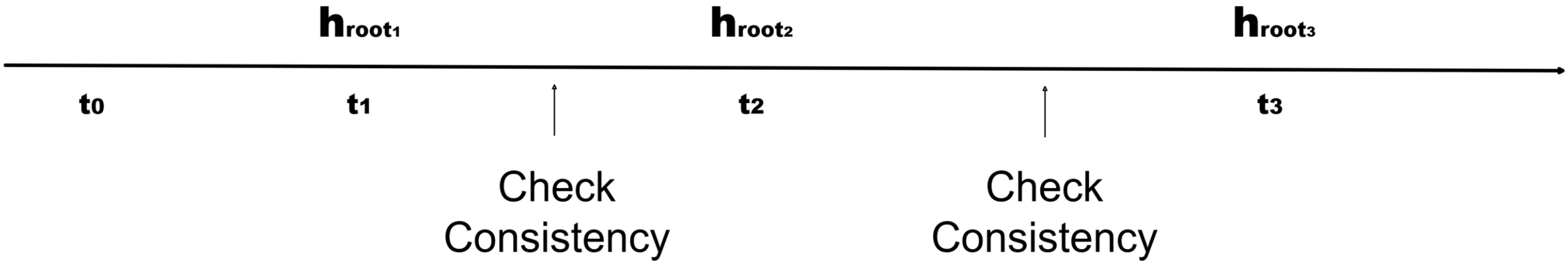


Username	PK
Alice	$pk_{A,1}$, t_1
Bob	$pk_{B,1}$, t_1

Username	PK
Alice	$pk_{A,1}$, t_1, $pk_{A,2}$', t_2
Bob	$pk_{B,1}$, t_1

Username	PK
Alice	$pk_{A,1}$, t_1 $pk_{A,2}$', t_2
Bob	$pk_{B,1}$, t_1 $pk_{B,2}$', t_3

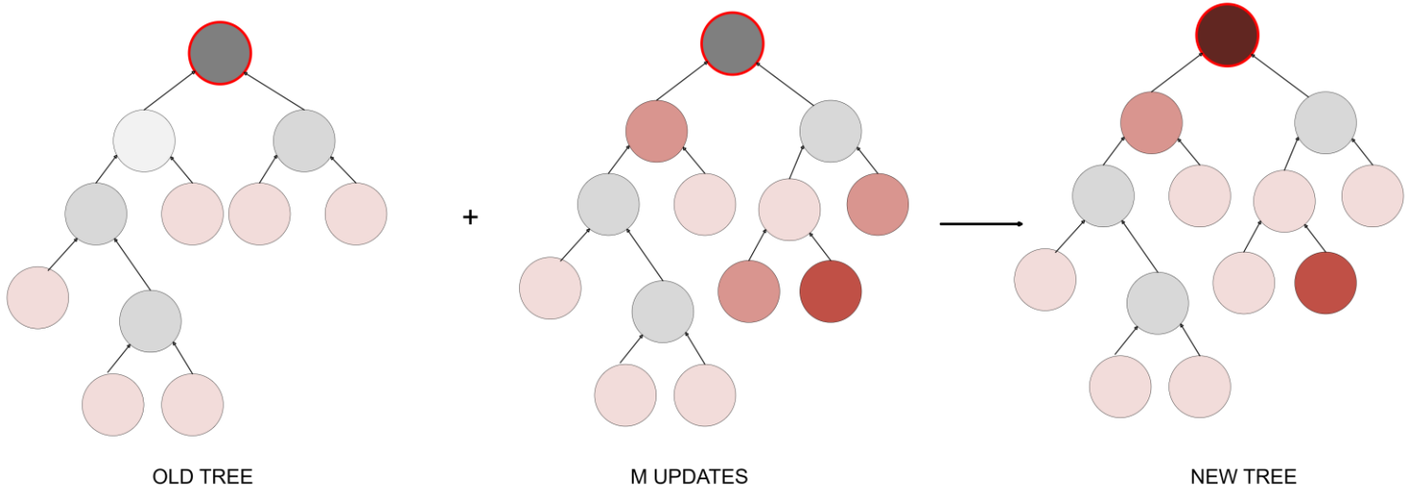
Periodic Publish – Auditing Append-only Consistency



Auditor checks:

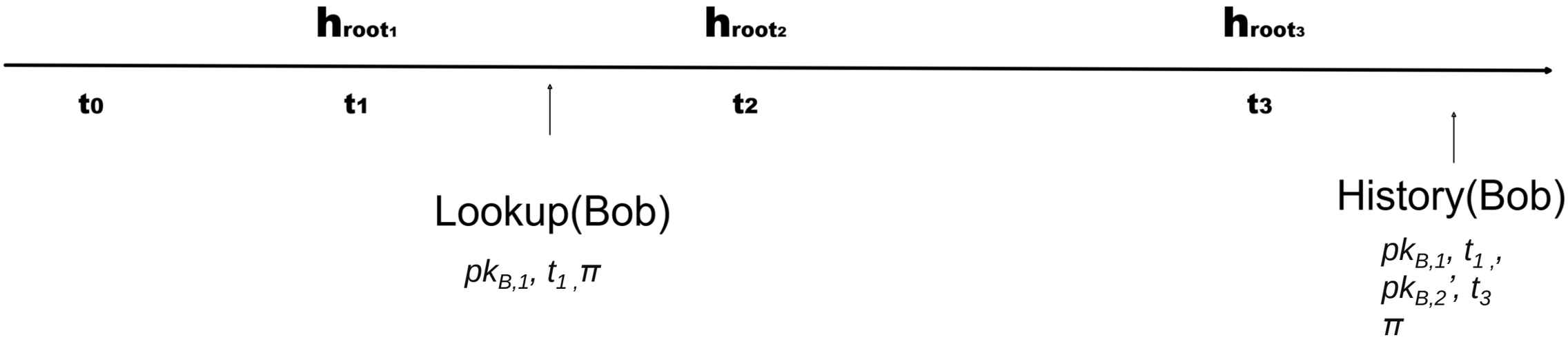
- Leaves of old tree are subset of new one
- New leaves have correct epoch t

Large Append-Only Proofs



- Consistency Proofs
  - Contain only leaf values (hashes), not the raw public keys themselves
  - However, proofs are O(MlogN) in size,
    - M = number of updates/epoch
    - N = total number of leaves in the tree

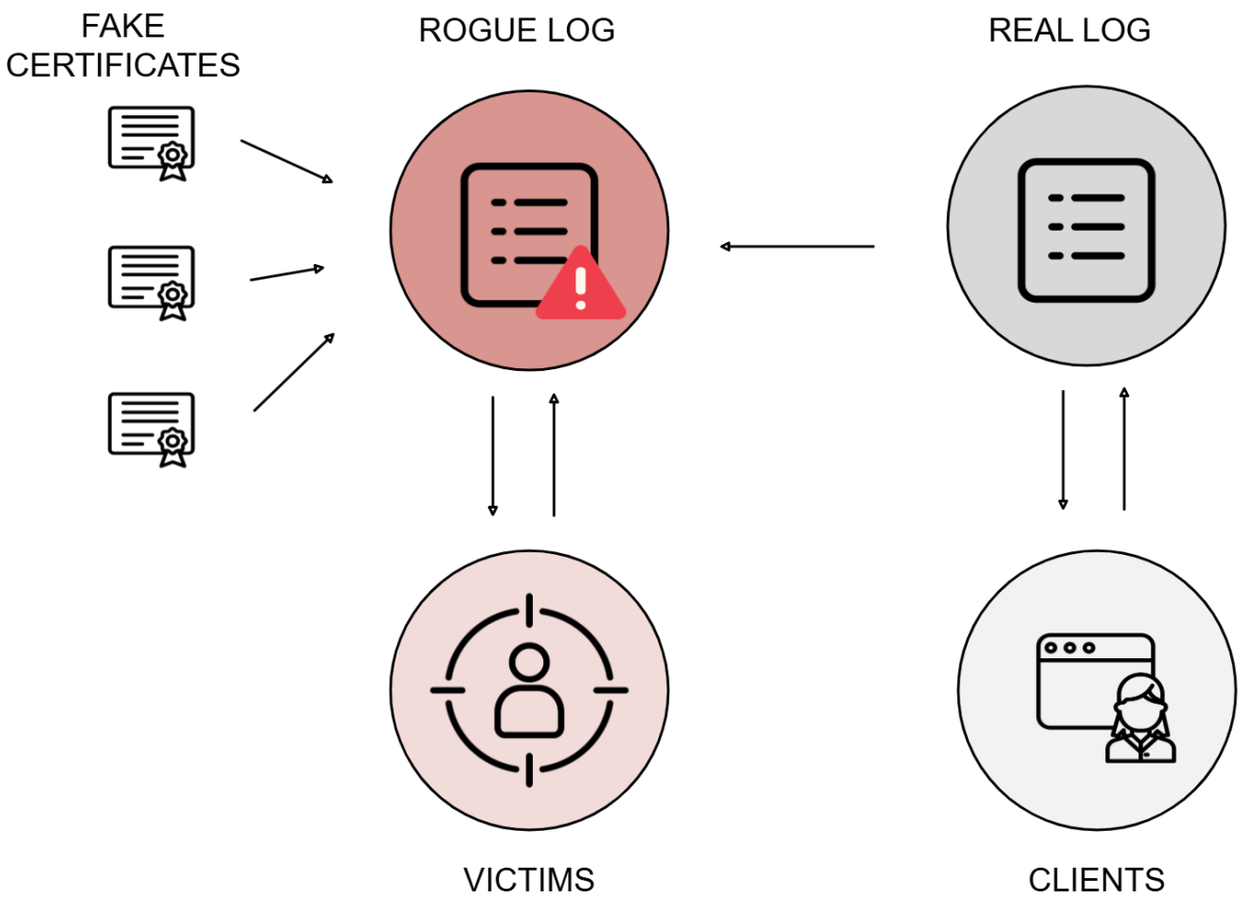
Client Queries



Log Equivocation

Consistency is key

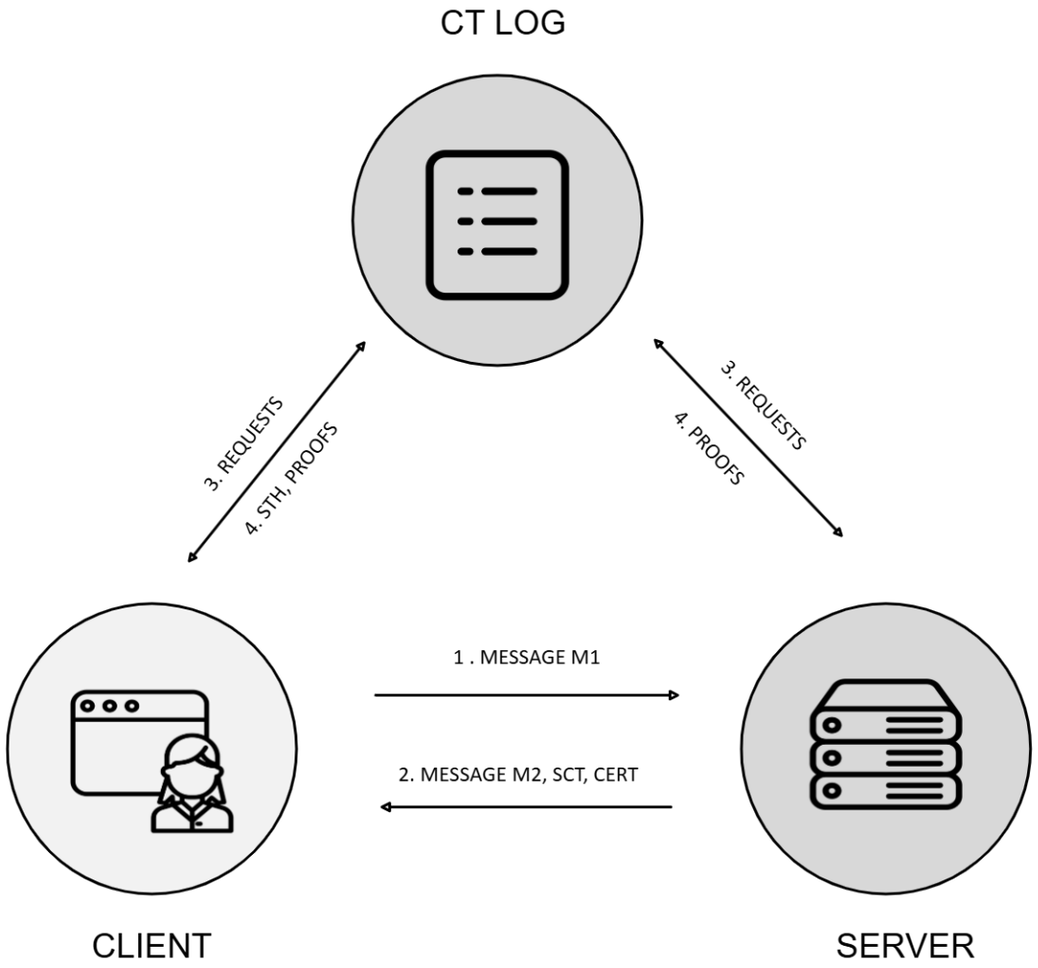
Split View Attack



How do you post commitments?

- Bulletin Board *consistently* viewed by **all** users
- Remove the Bulletin Board, and instead
  - Gossiping
  - Witnessing
- Alternatives:
  - 3rd Party
  - Blockchain
  - Trusted Hardware

Gossiping



- Out-of-band
  - Extra-infrastructure
- In-band
  - Piggybacking
  - Modification of protocol

Witnessing

- *Proactive* vs reactive (Gossip)
- Multiple Independent Parties
- Part of a *M-of-N* Trust Policy

- ***Check*** consistency proofs
- ***Co-sign*** Commitments (checkpoints)

Other Applications

More Transparency Applications

- Binary Transparency
- Signature Transparency
- Firmware Transparency
- AI model Transparency

Open Problems

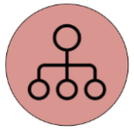
Open Problems

Real-time Detection of Misbehavior:

How can we develop faster, automated systems that continuously monitor transparency logs and promptly flag suspicious activity?

Privacy-Preserving Logging Schemes:

How can transparency logging schemes be designed to preserve privacy?



Verifiable Data Structures:

What is the optimal verifiable data structure for Transparency Systems?

Reducing costs of monitoring and auditing:

How can the proofs offered by Transparency systems be made more compact, to allow for quick audits and verifications?

Evolution of bar of trust

- {THING}
- {THING} + {THING}.sig
- {THING} is ***logged***
- {THING} is ***transparent***

- *Someone* is *accountable* for {THING}
- {THING} is ***discoverable***
- **Claims** about {THING} are ***falsifiable***
- There exist **entities** able to *verify* claims about {THING}.

Enjoy your weekend!