ELEKTRA: Efficient Lightweight multi-dEvice Key TRAnsparency

Julia Len
Cornell Tech

Melissa Chase
Microsoft Research

Esha Ghosh
Microsoft Research

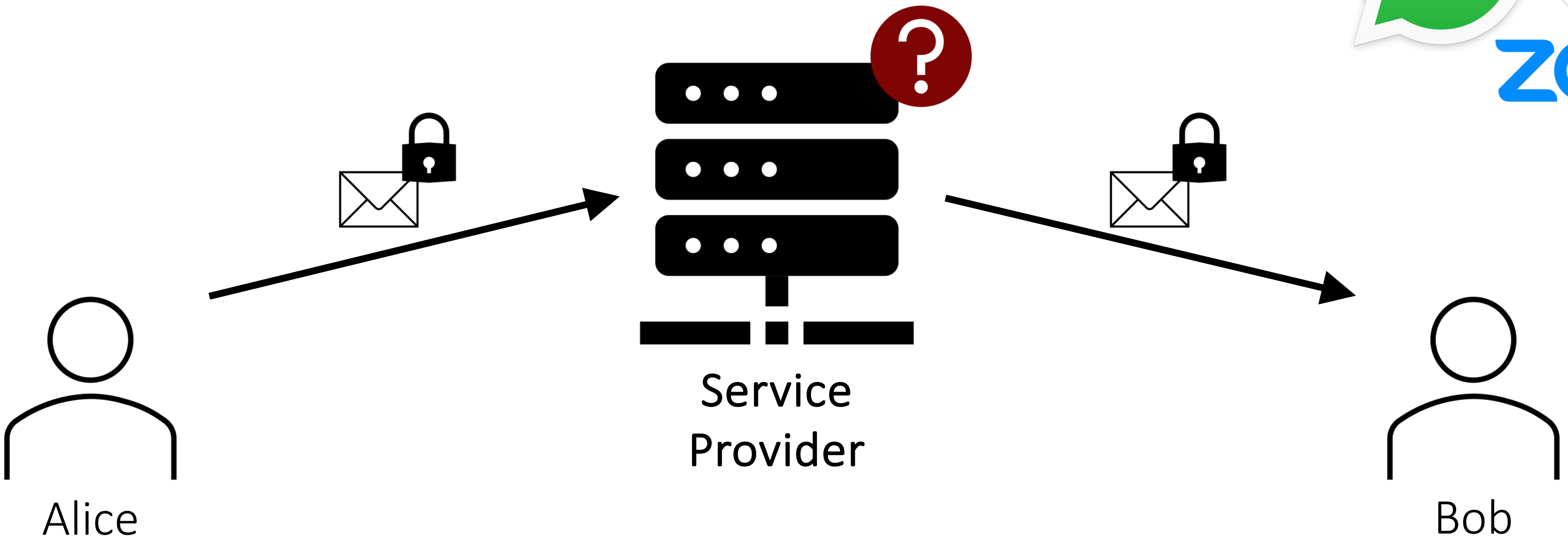
Daniel Jost
New York University

Balachandar Kesavan
Zoom Video Communications

Antonio Marcedone
Zoom Video Communications

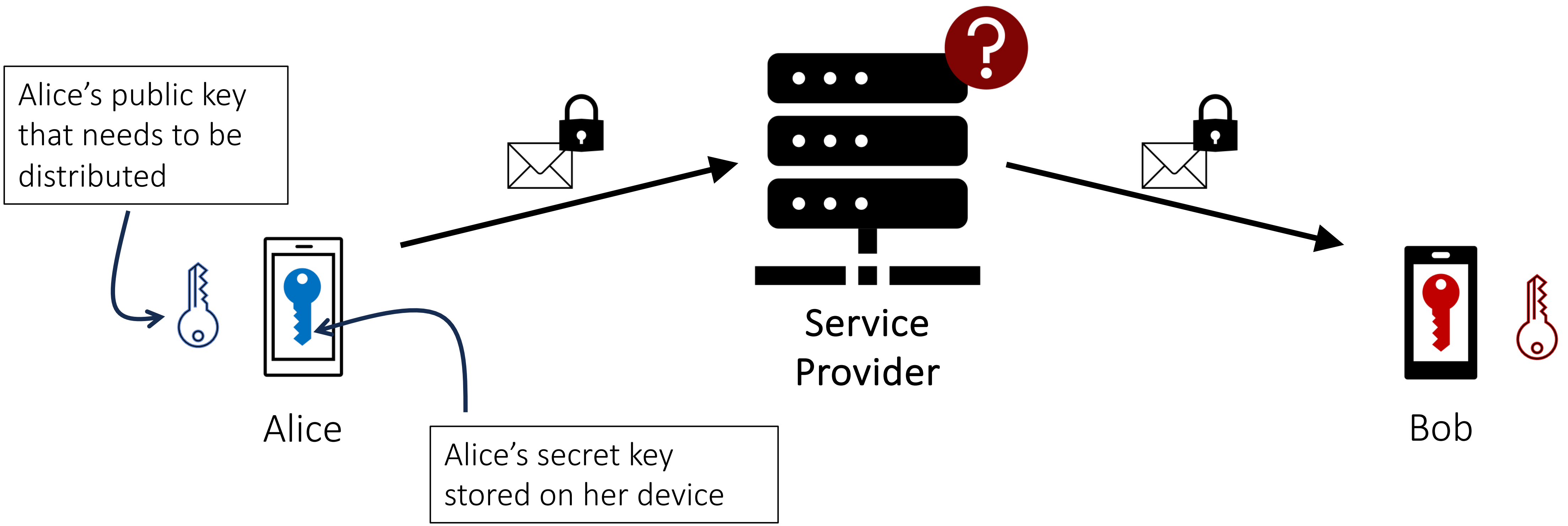
CCS 2023
November 29, 2023

End-to-end encryption (E2EE)



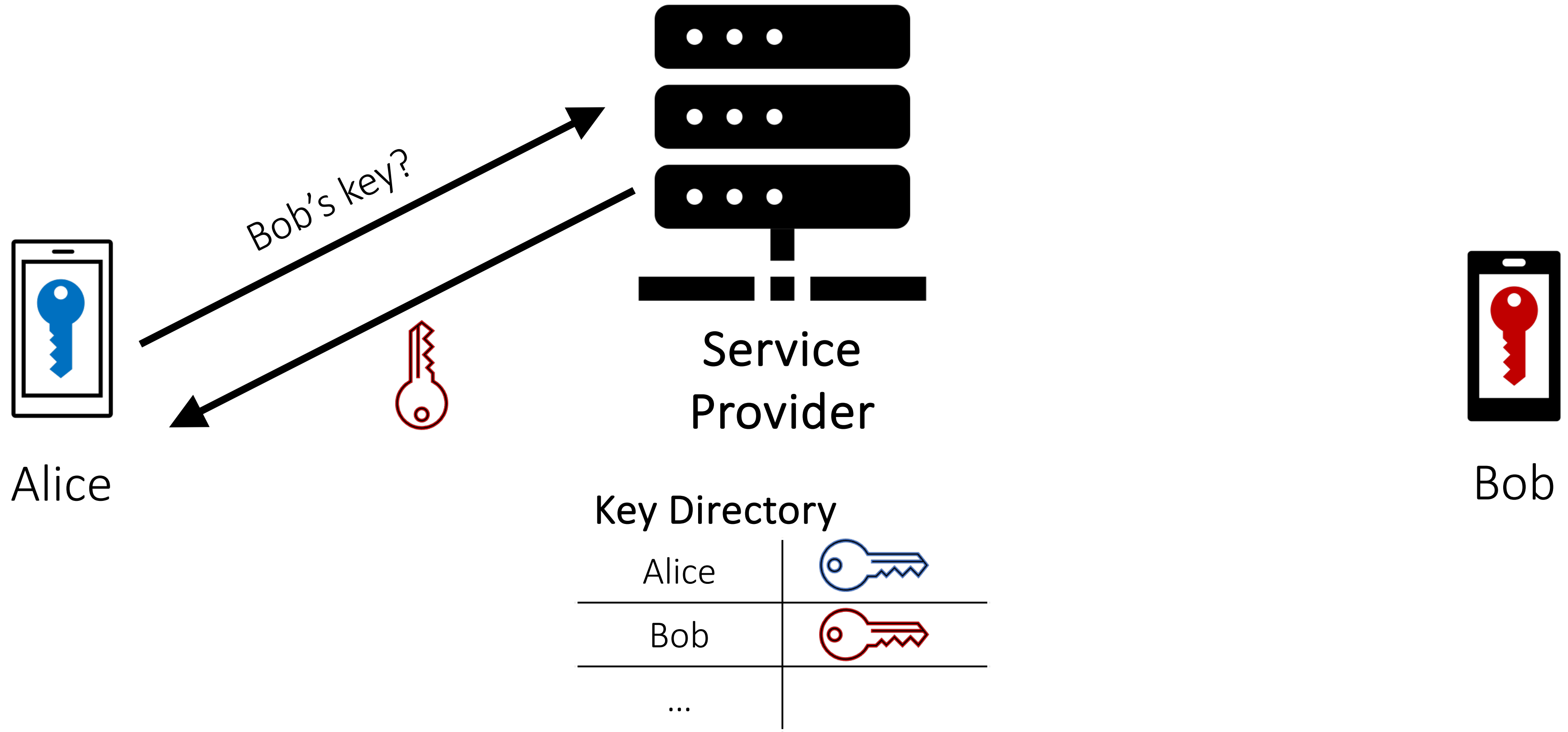
- We expect from E2EE that the Service Provider cannot decrypt the communication

End-to-end encryption (E2EE)



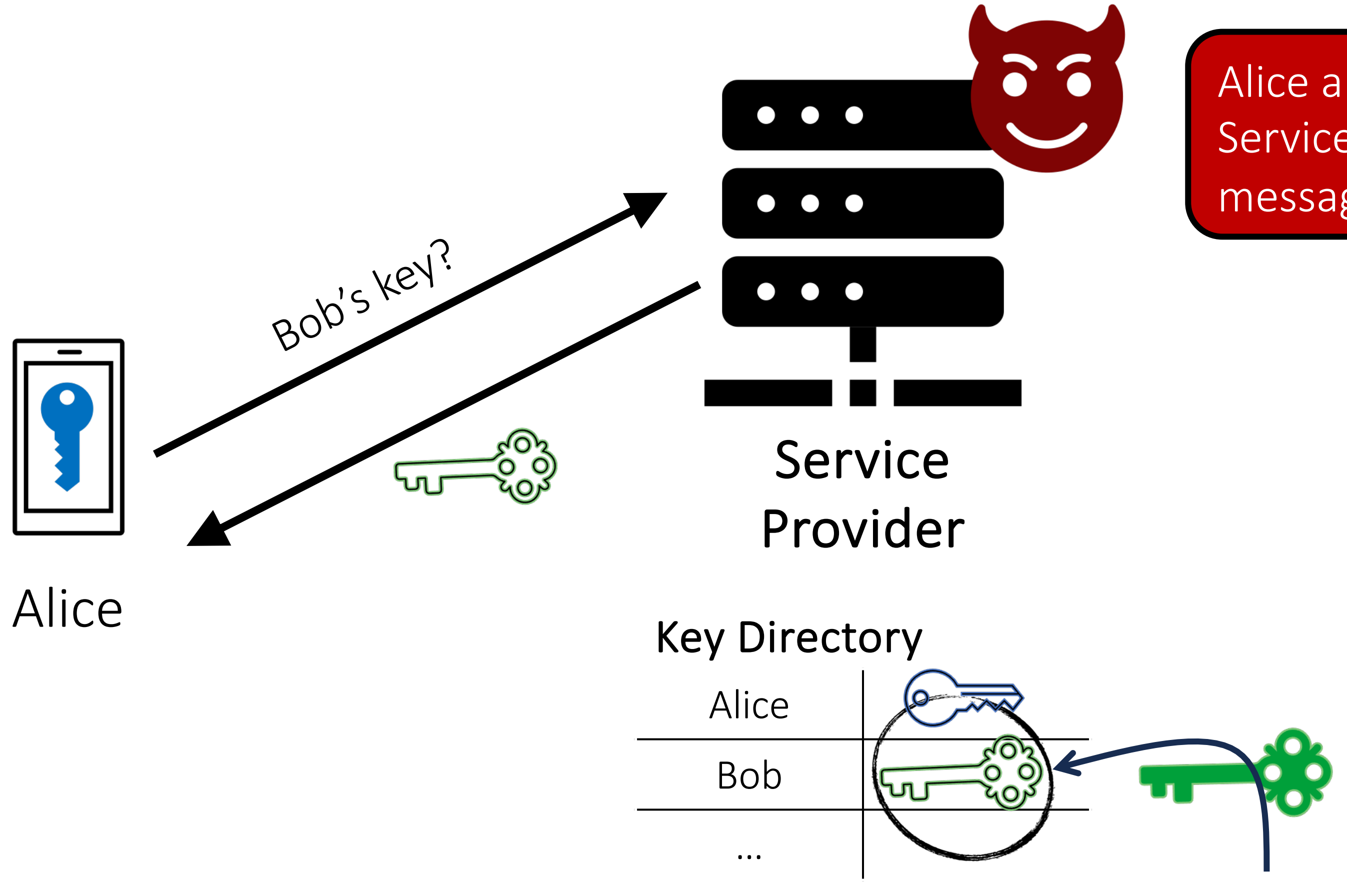
- We expect from E2EE that the Service Provider cannot decrypt the communication
- But how do Alice and Bob get each other's keys?

End-to-end encryption (E2EE)



- Public keys are sent to the Service Provider to be stored and distributed

Meddler-in-the-Middle (MitM) attack

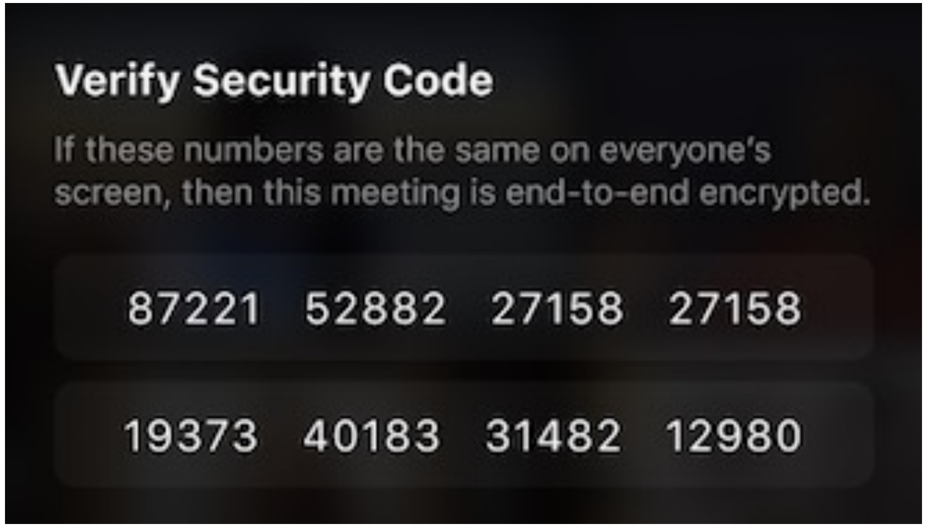


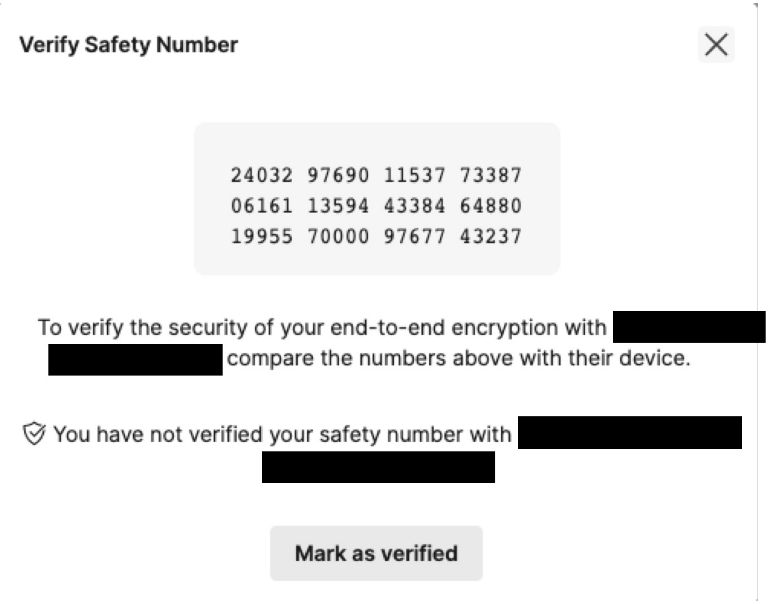
Alice and Bob have no idea that the Service Provider can read their messages!

Service Provider replaces Bob’s key with one it knows

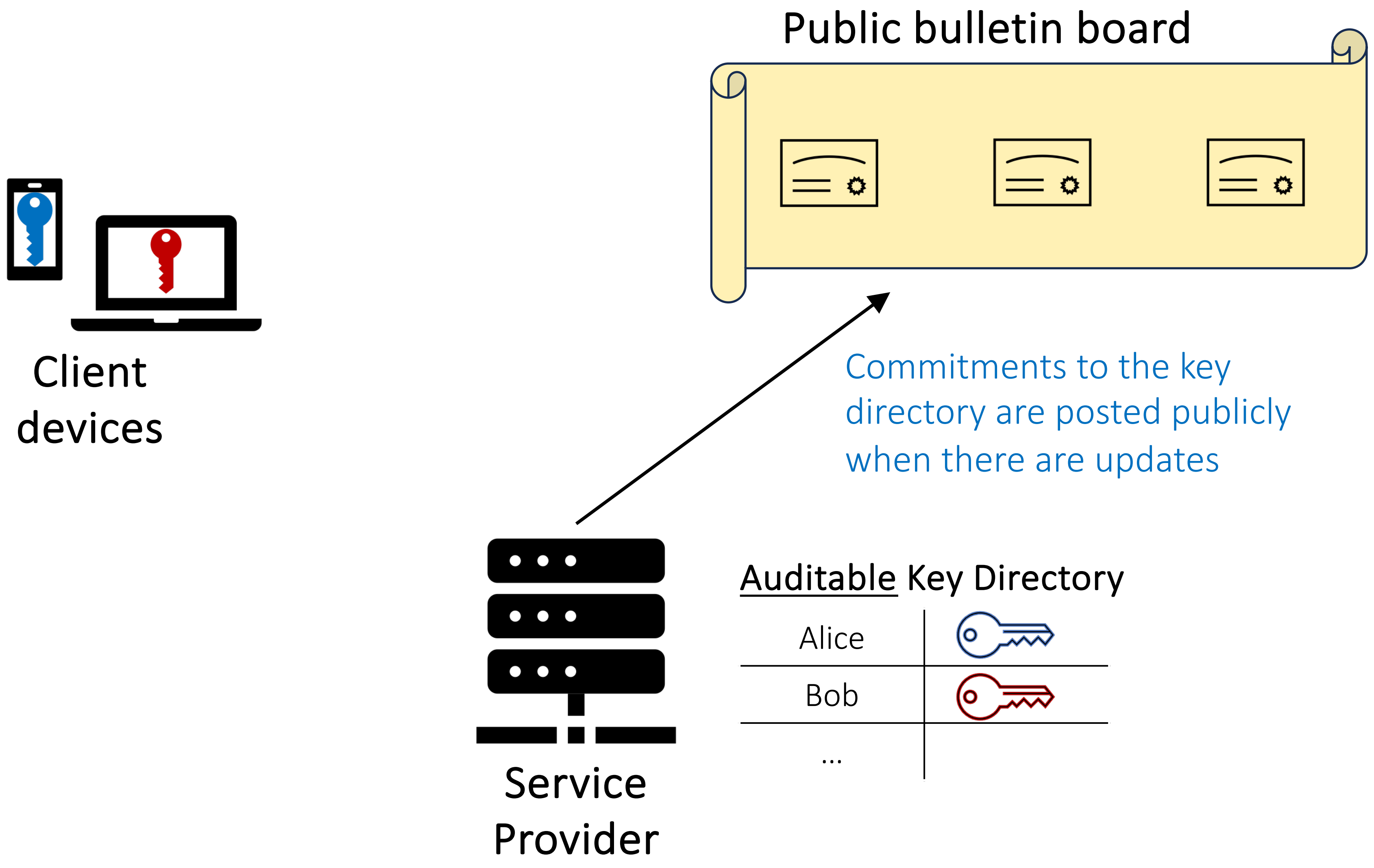
Current solutions offload security onto users

- Major messaging apps like Signal and WhatsApp have users compare a 60-digit "safety number" either manually or by scanning a QR code
- Major video calling apps like Zoom and Microsoft Teams have users read out "security codes"

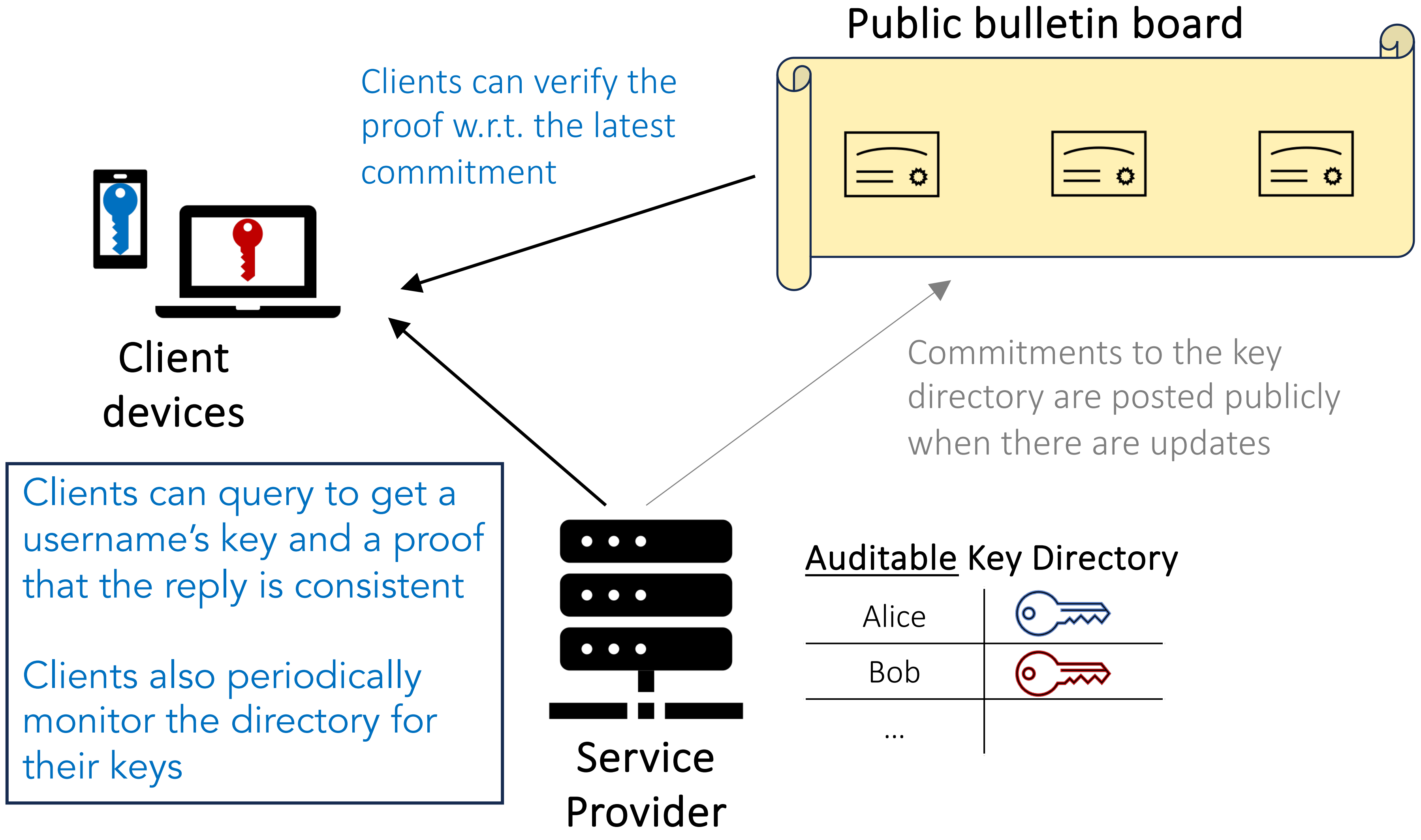




Key transparency



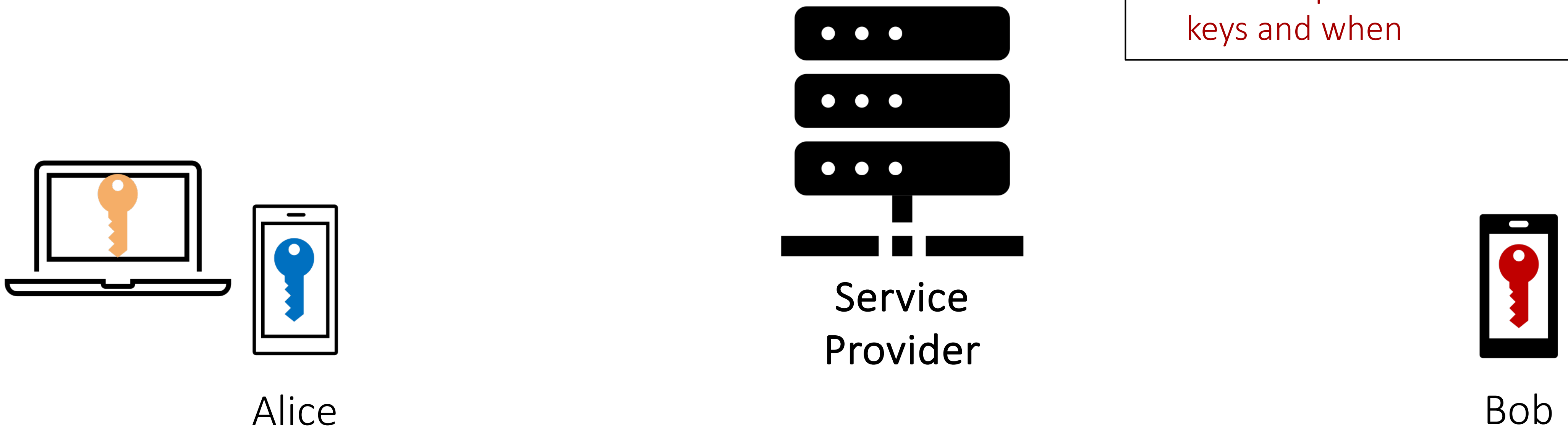
Key transparency



Key transparency: Privacy

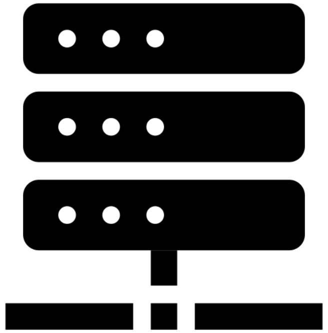
<u>Bad for privacy</u>:

- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when



Key transparency: Privacy

<u>Bad for privacy</u>:
- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when



Service Provider

Alice

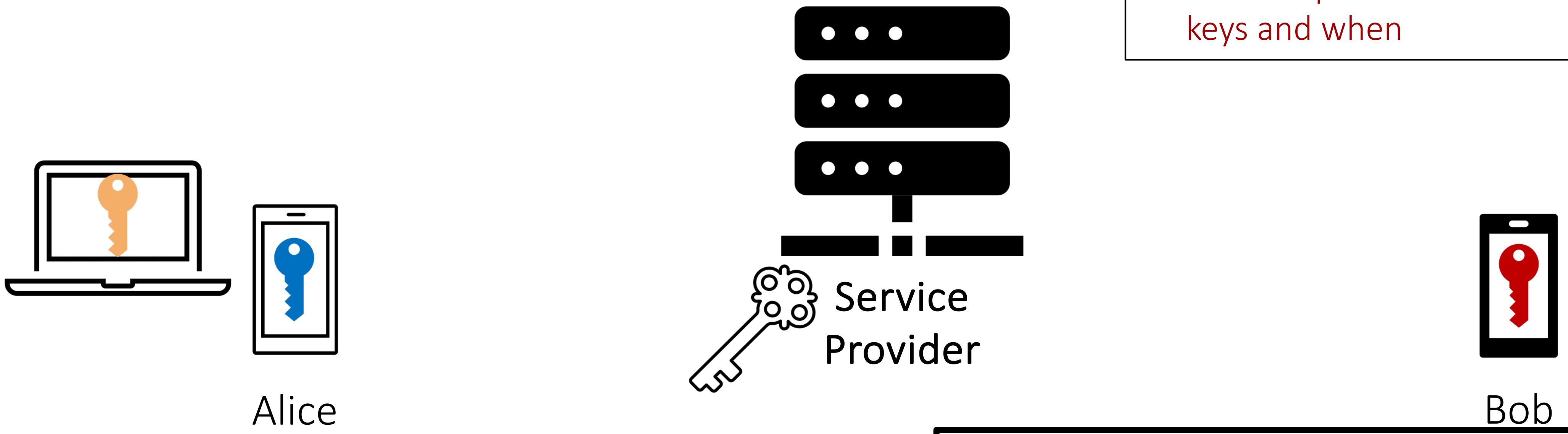
Bob

<u>Privacy goals</u>:
- Sensitive information about when users register or update keys should not get leaked to external clients
- Clients should not learn information about other keys in the directory aside from the one they query
- No privacy from the server

Key transparency: Privacy

Bad for privacy:

- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when



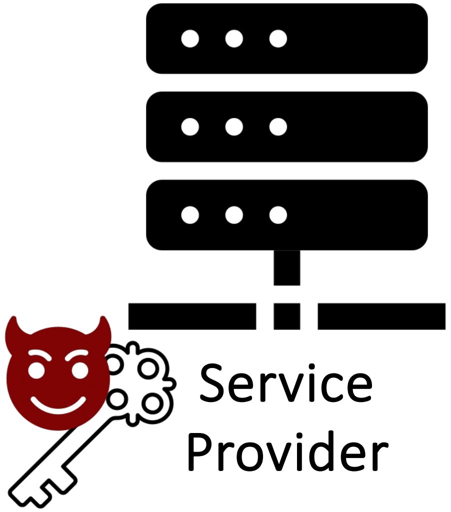
Privacy goals:

- Sensitive information about when users register or update keys should not get leaked to external clients
- Clients should not learn information about other keys in the directory aside from the one they query
- No privacy from the server

Prior work in key transparency guarantees privacy by having the Service Provider choose a secret key and then computing a Verifiable Random Function (VRF) over data.

Key transparency: Privacy

Bad for privacy:
- Learning information about other keys in the directory during key lookups!
- External parties learn who updates their keys and when



Prior work in key transparency guarantees privacy by having the Service Provider choose a secret key and then computing a Verifiable Random Function (VRF) over data.

But this key could get compromised!
So we want privacy to be recoverable even after server compromise, a.k.a. post-compromise security

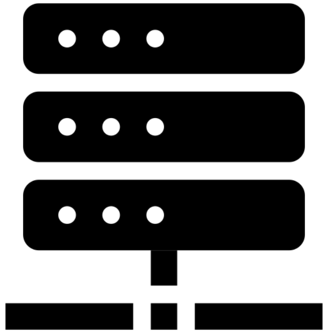
Privacy goals:
- Sensitive information about when users register or update keys should not get leaked to external clients
- Clients should not learn information about other keys in the directory aside from the one they query
- No privacy from the server

Key transparency: Multiple devices

Prior academic work model the single device setting

Mention extensions of mapping username to list of keys

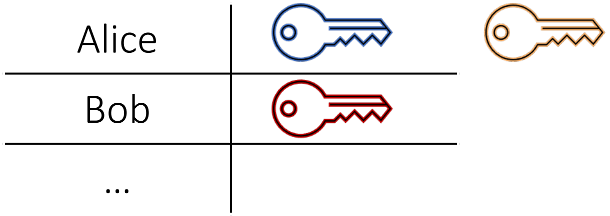
Alice



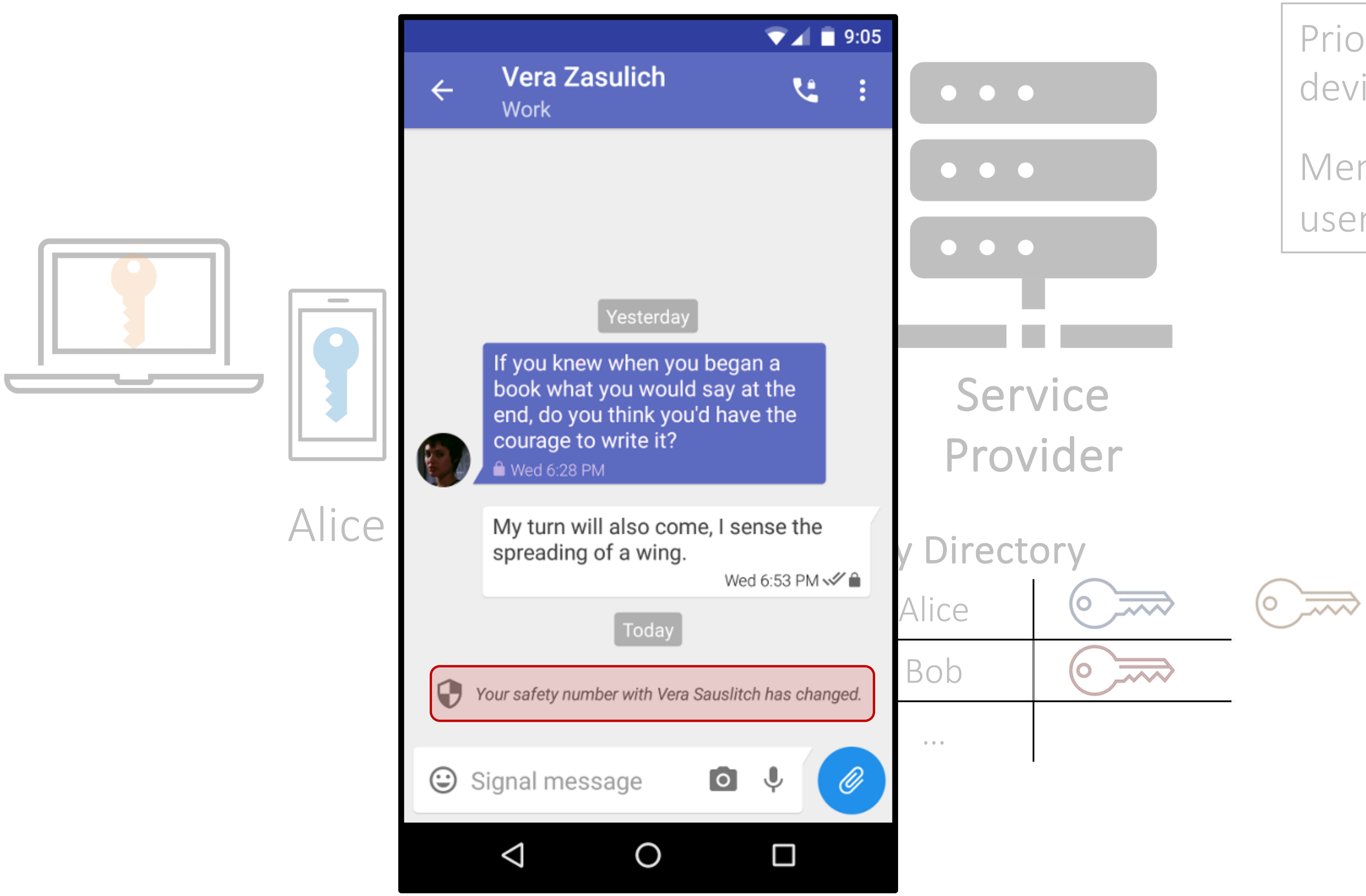
Service Provider

Bob

Key Directory



Key transparency: Multiple devices



Prior academic work model the single device setting

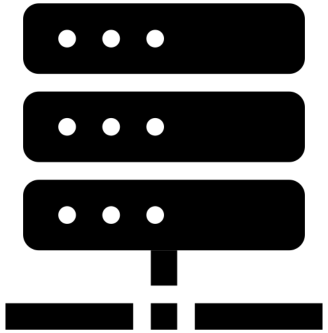
Mention extensions of mapping username to list of keys

Key transparency: Multiple devices

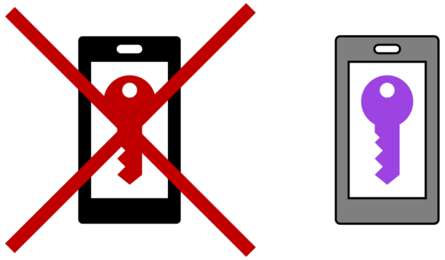
Prior academic work model the single device setting

Mention extensions of mapping username to list of keys

Alice

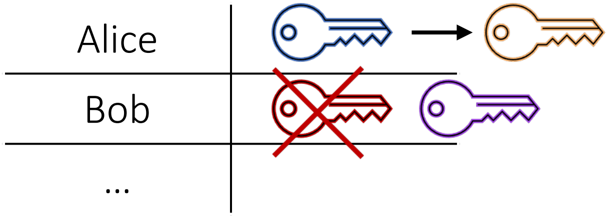


Service Provider



Bob

Key Directory

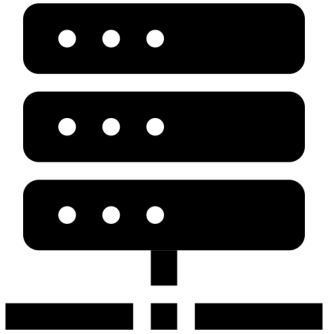


And in practice users can lose devices and need to **reset their accounts**!

Many E2EE apps allow for this, but this hasn't been modeled at all by prior key transparency systems

Key transparency: Multiple devices

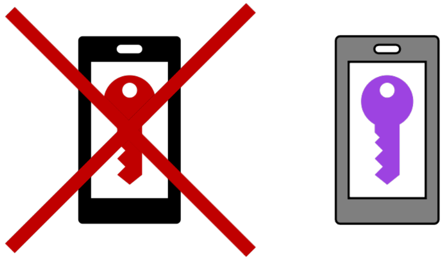
Alice



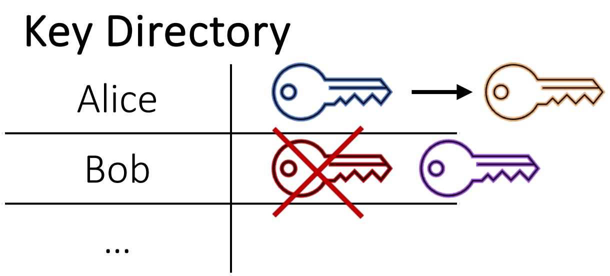
Service Provider

Prior academic work model the single device setting

Mention extensions of mapping username to list of keys



Bob



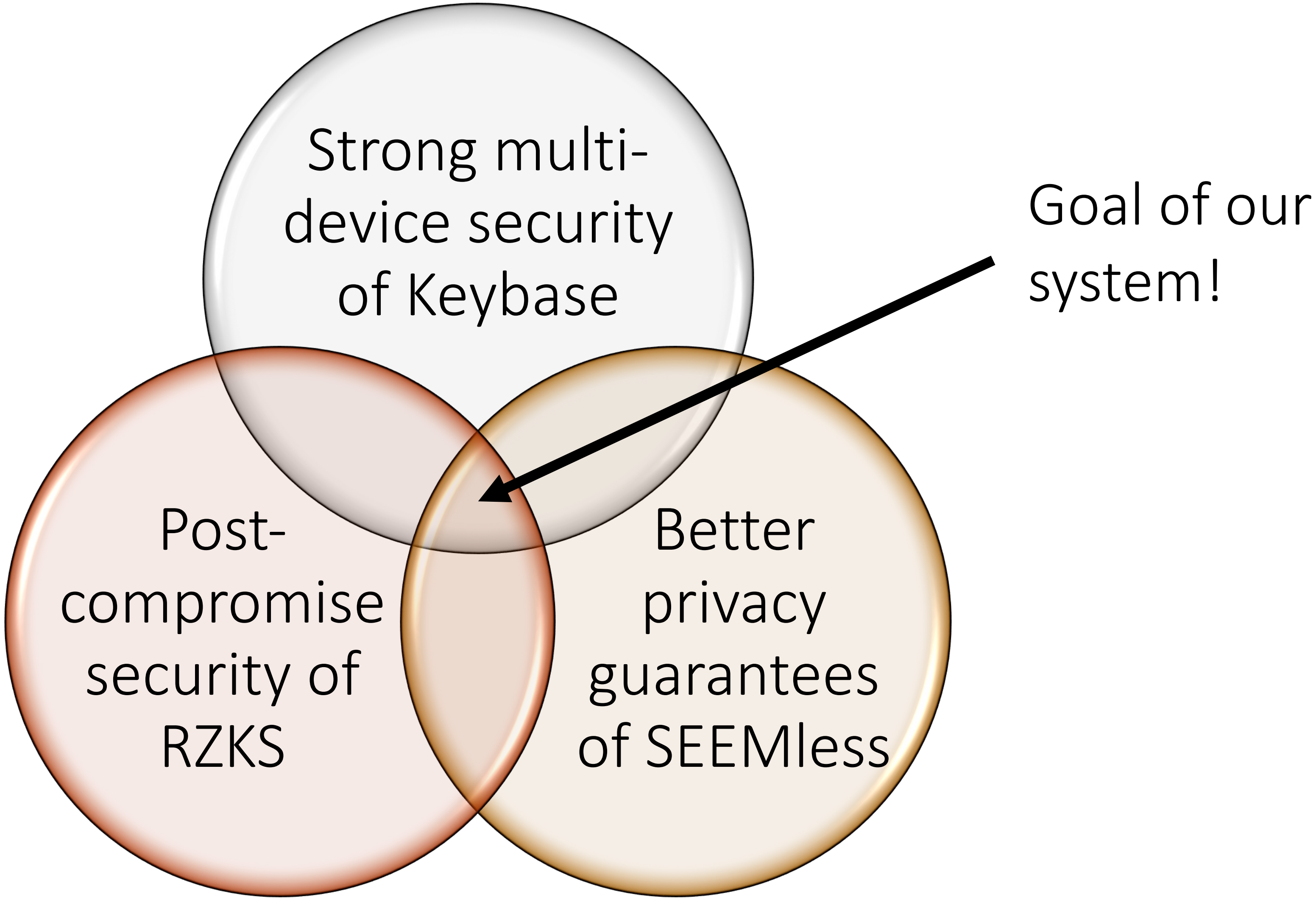
And in practice users can lose devices and need to **reset their accounts**!

Many E2EE apps allow for this, but this hasn't been modeled at all by prior key transparency systems

A better approach:
existing devices sign key updates

Makes MitM harder since the provider either needs to fake an account reset (suspicious!) or forge a signature

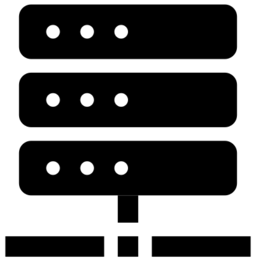
New design desiderata



ELEKTRA: A new key transparency system

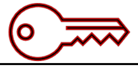
Client devices

- We model *keychains* which capture the evolution of a user's public keys
- User keychains and their updates are stored in a *multi-device verifiable key directory (MVKD)*
- ELEKTRA is our MVKD construction

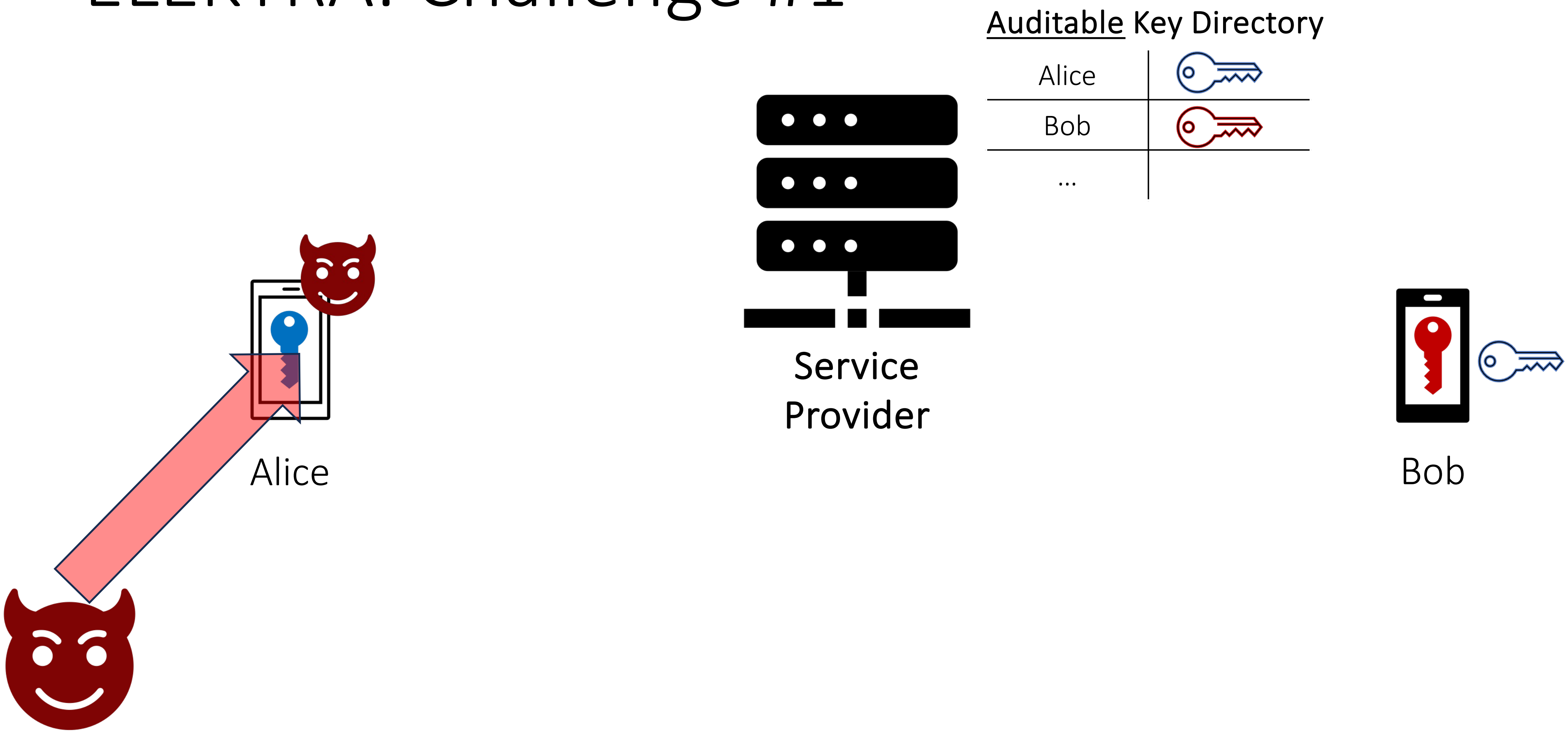


Service Provider

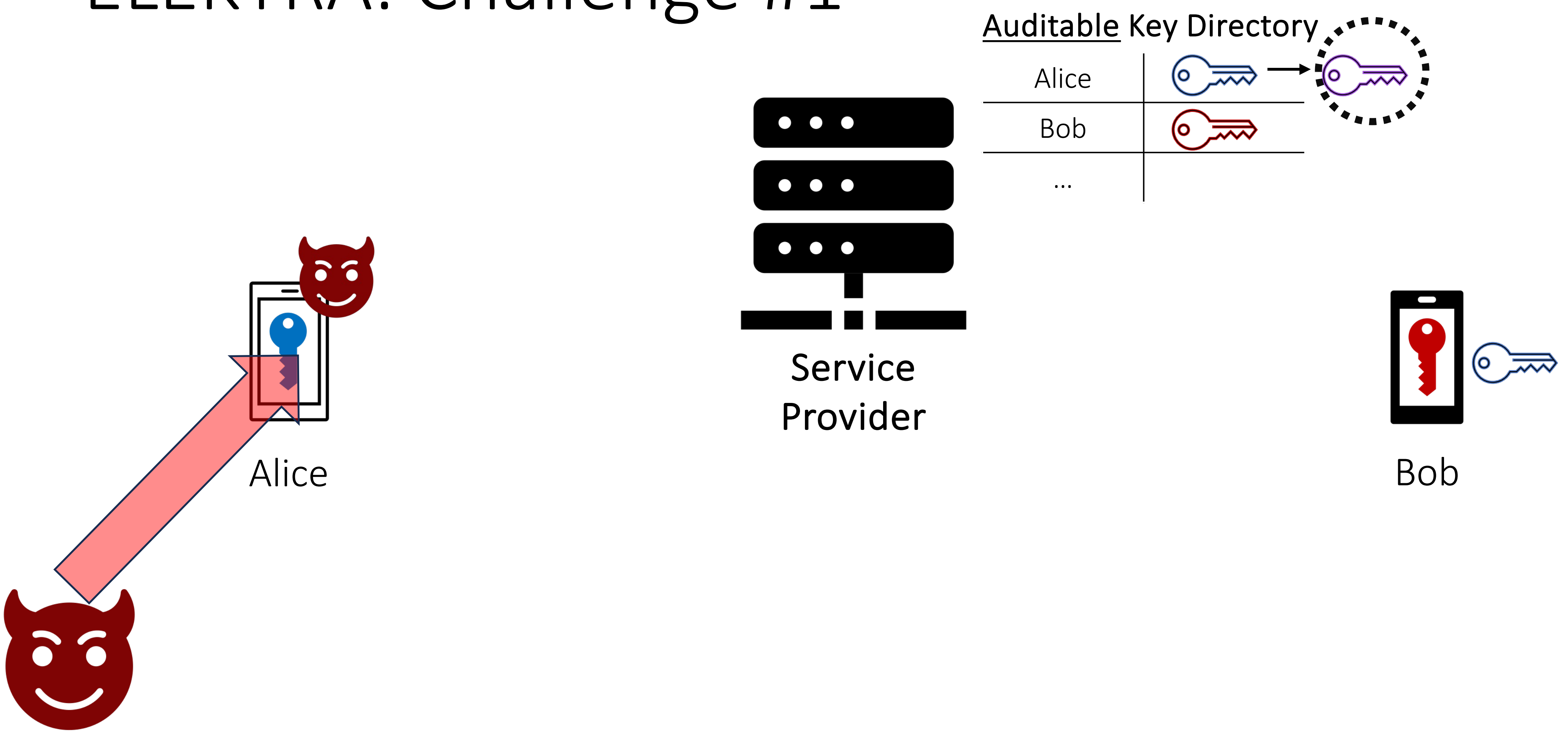
Auditable Key Directory

Alice	
Bob	
…	

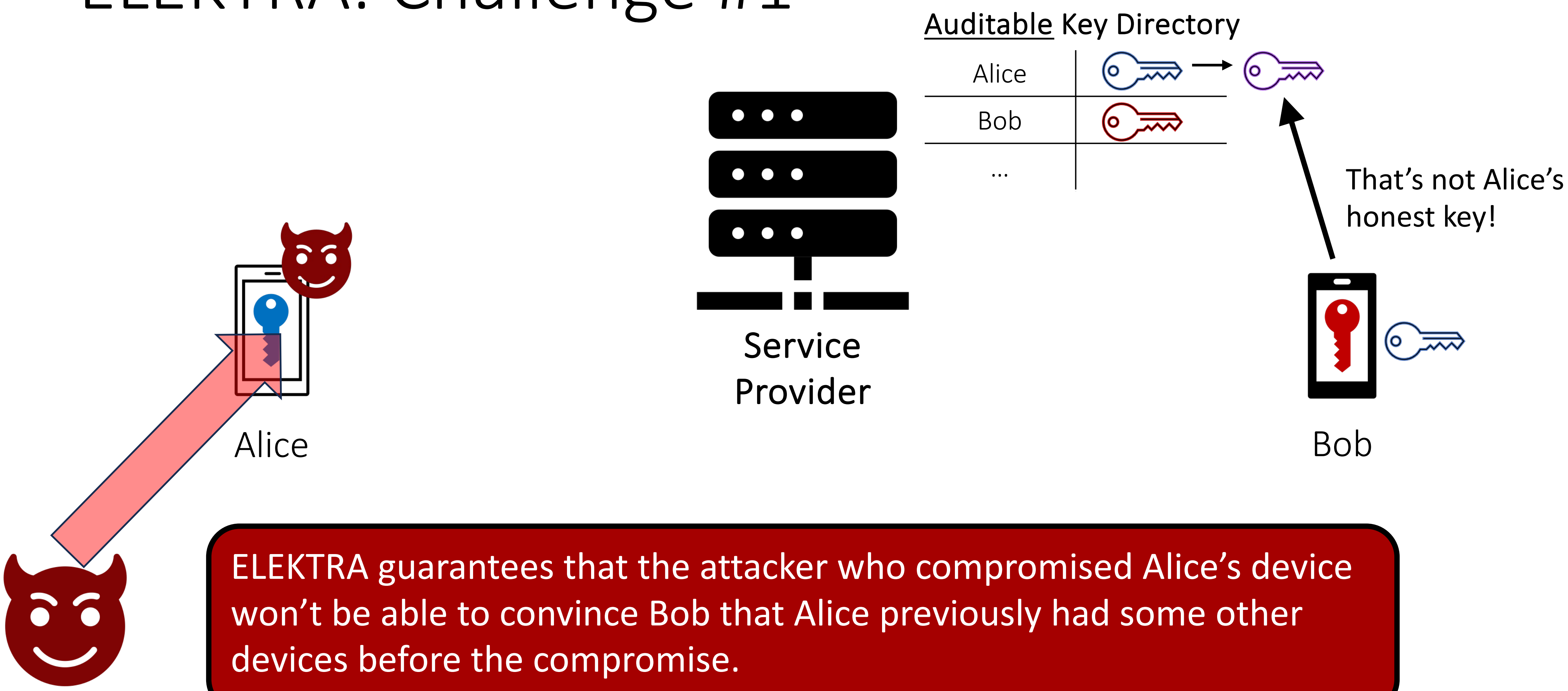
ELEKTRA: Challenge #1



ELEKTRA: Challenge #1

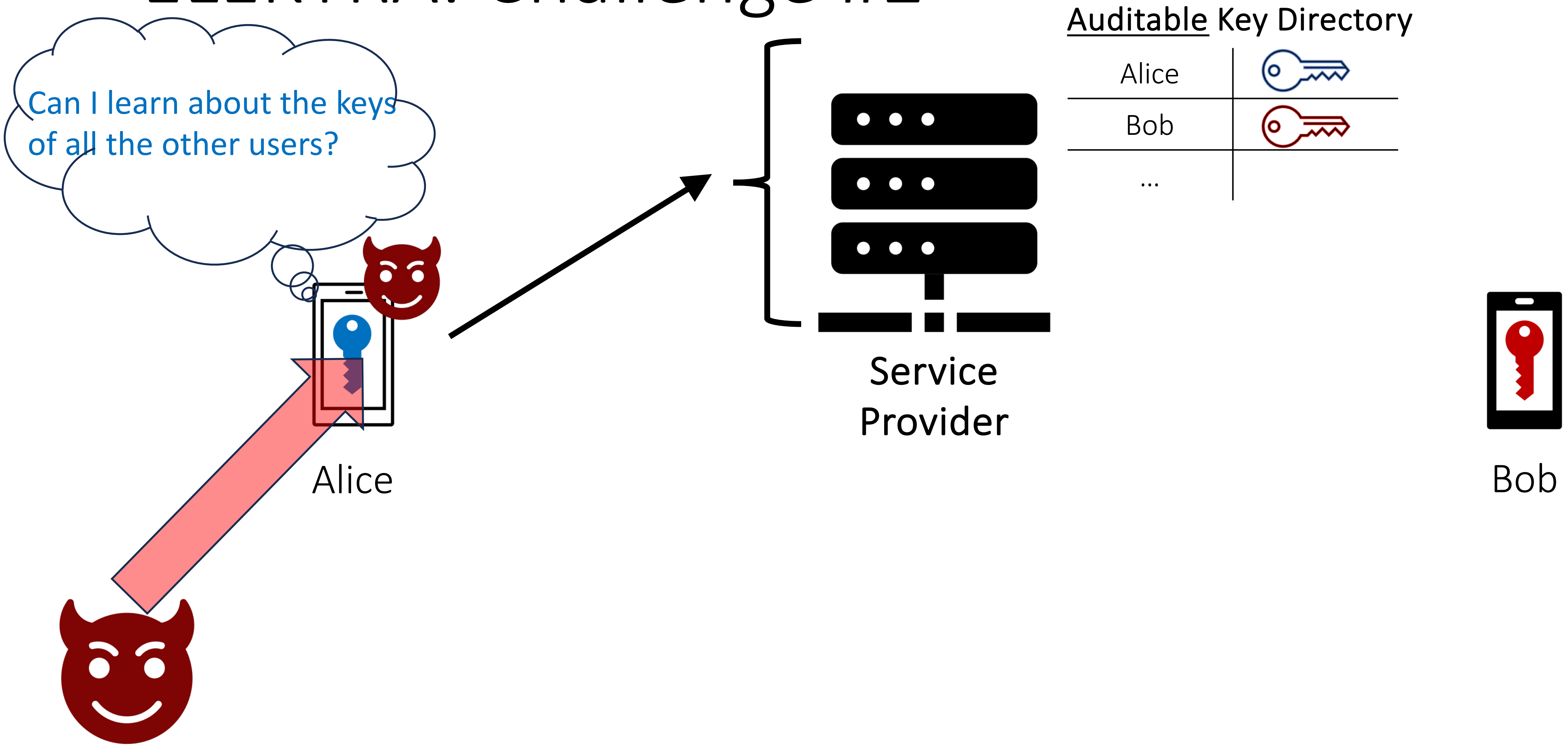


ELEKTRA: Challenge #1

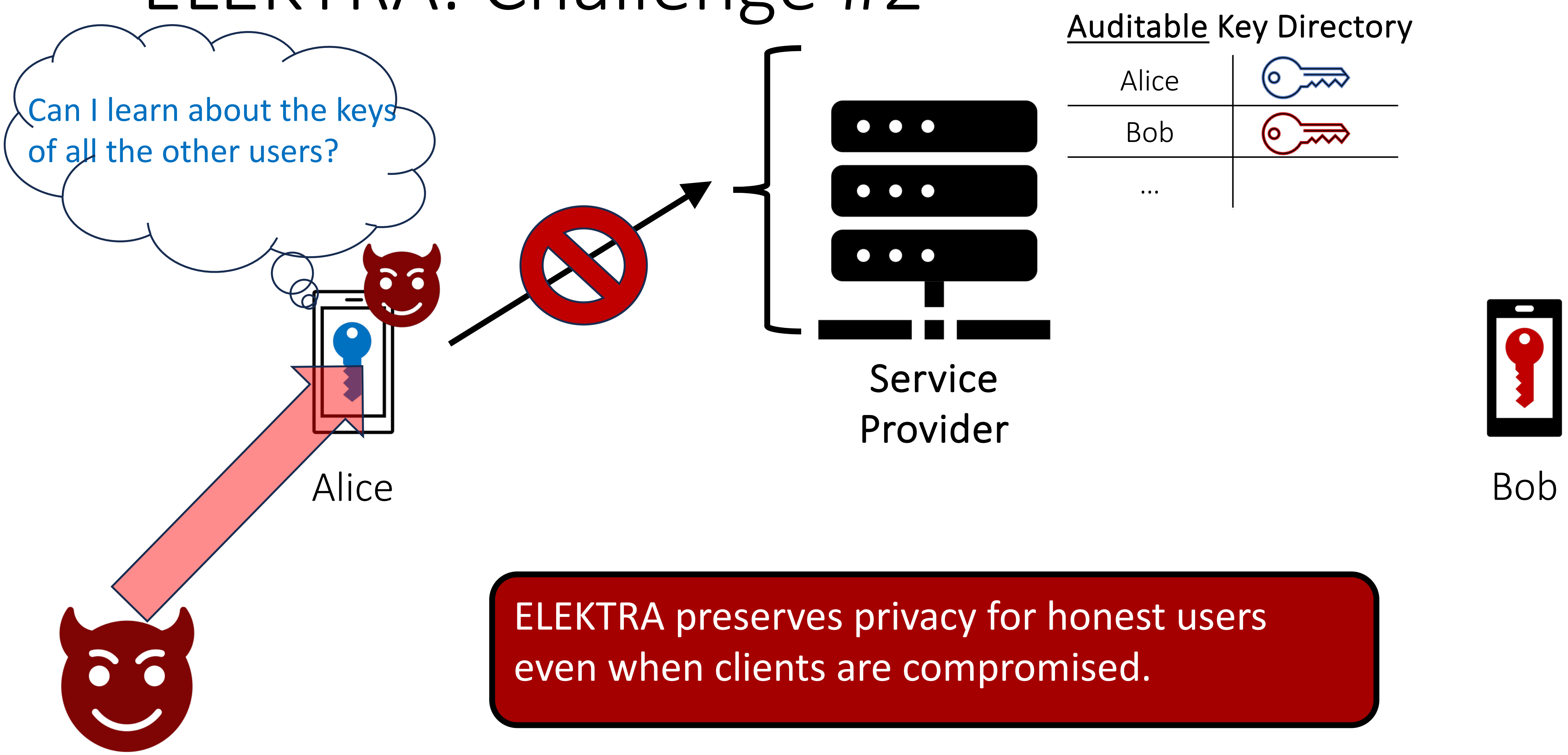


ELEKTRA guarantees that the attacker who compromised Alice's device won't be able to convince Bob that Alice previously had some other devices before the compromise.

ELEKTRA: Challenge #2

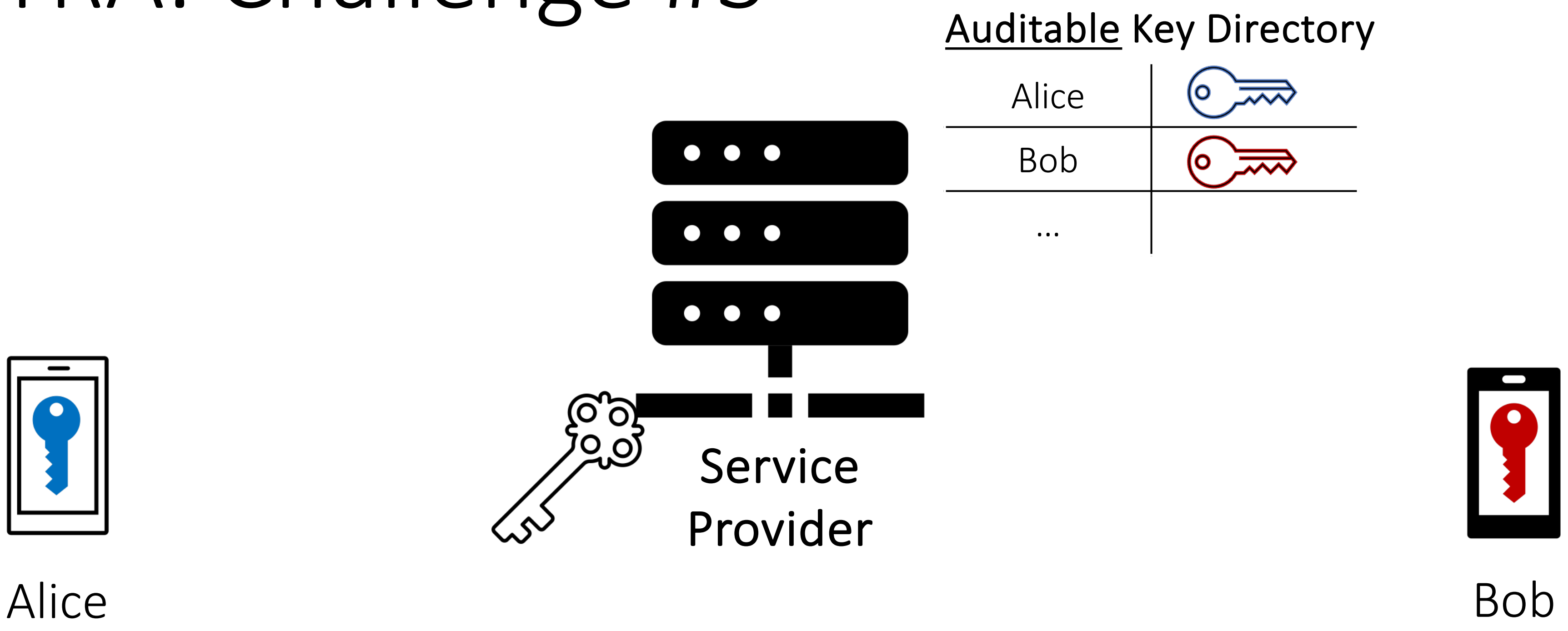


ELEKTRA: Challenge #2

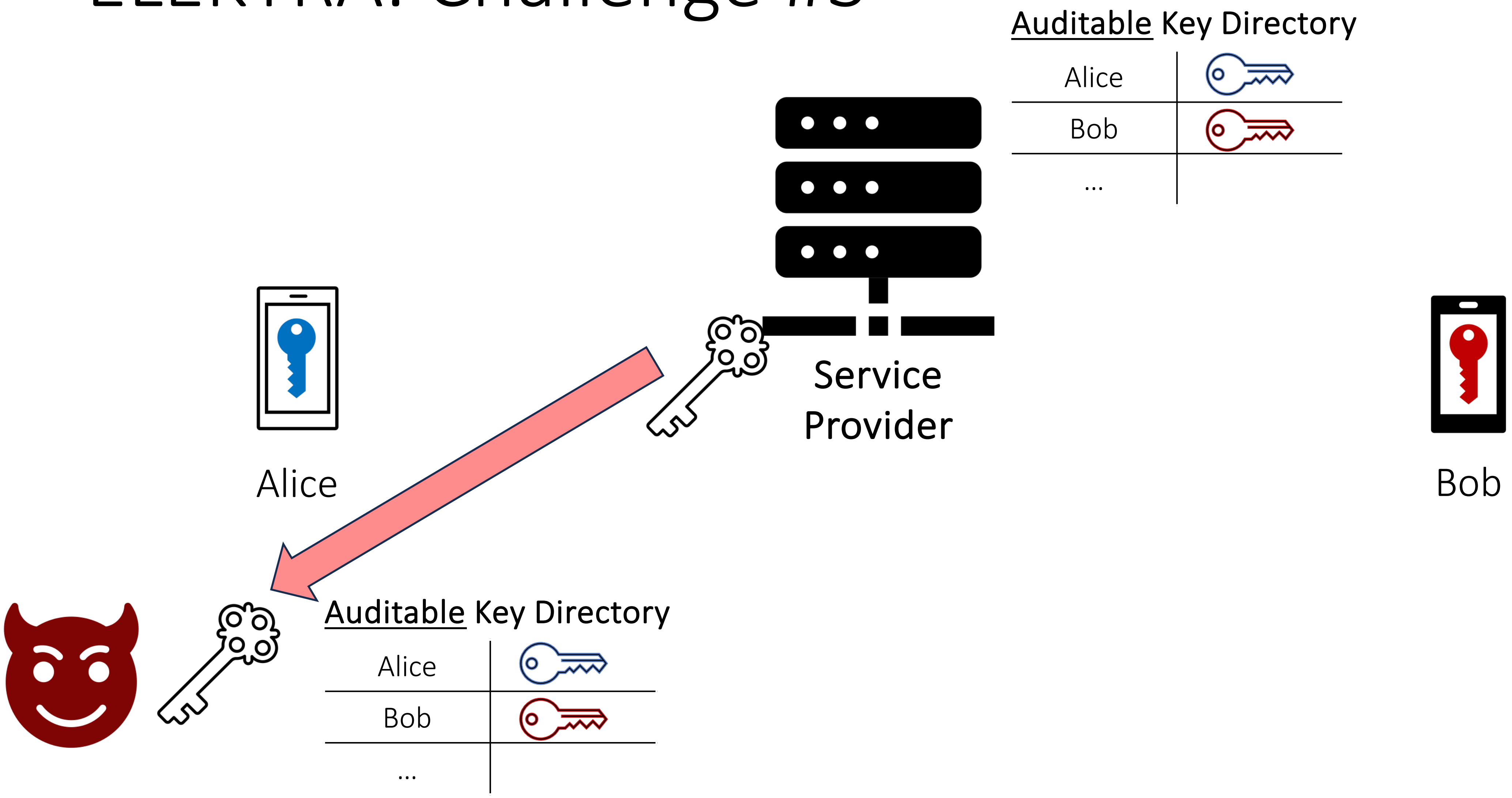


ELEKTRA preserves privacy for honest users even when clients are compromised.

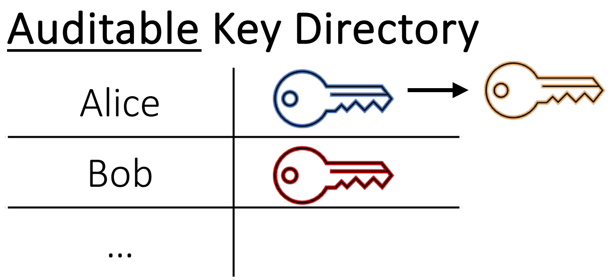
ELEKTRA: Challenge #3

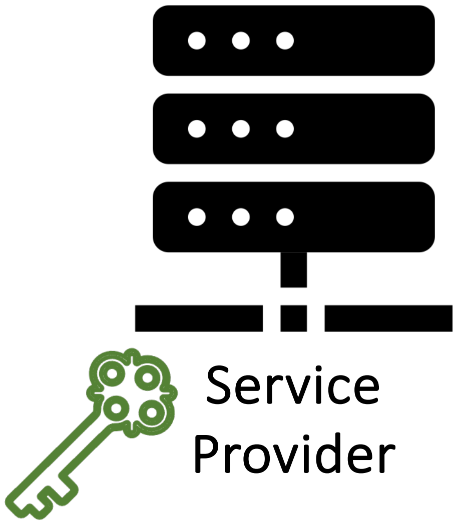


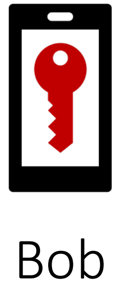
ELEKTRA: Challenge #3

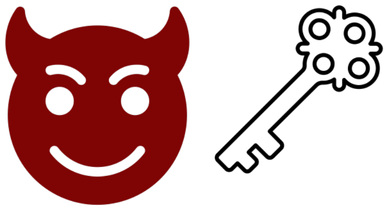


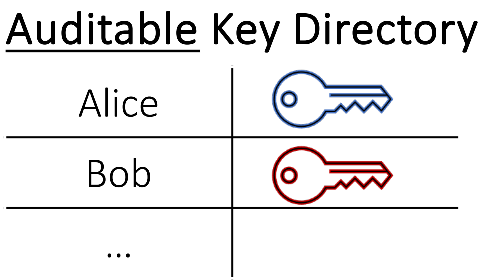
ELEKTRA: Challenge #3











ELEKTRA offers post-compromise security, so the attacker doesn't learn key updates after rotating the Service Provider's key.

ELEKTRA: Rigorous security proofs

Completeness

- Desired functionality for honest parties interacting with an honest server
- Dishonest clients should not be able to affect the protocol for honest clients

Soundness

- Security in the presence of an active and fully compromised server
- We define a stronger form of soundness, which is extractable soundness

Privacy

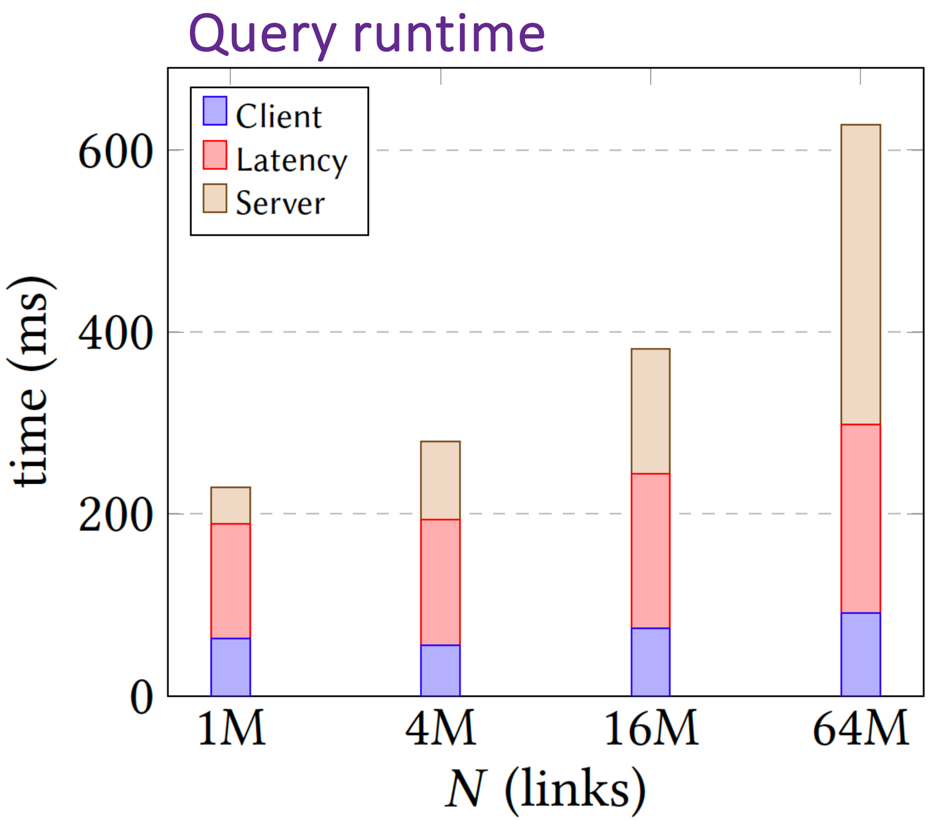
- Algorithms don't leak extra information about the server's state other than some well-defined leakage function
- More complex than prior definitions: we model corrupted clients and a corrupted server (for PCS guarantees)

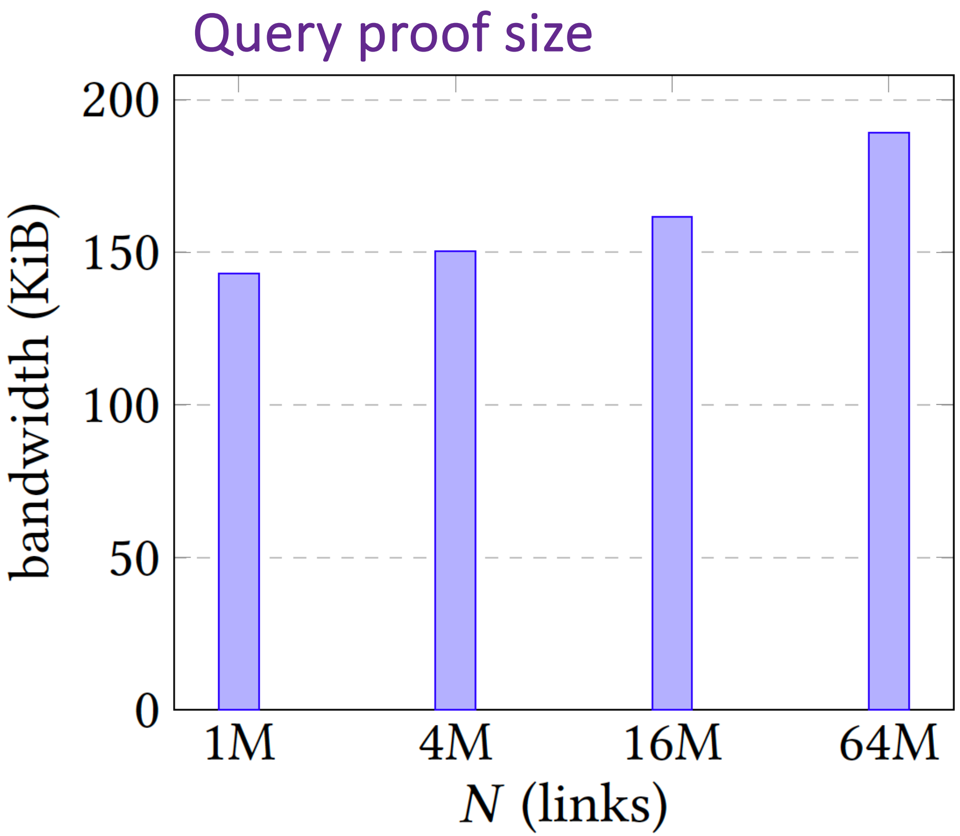
Experiments

- Implementation written in Go
- Server run on AWS instance, client run on Google Pixel 6 phone

Experiments: Query

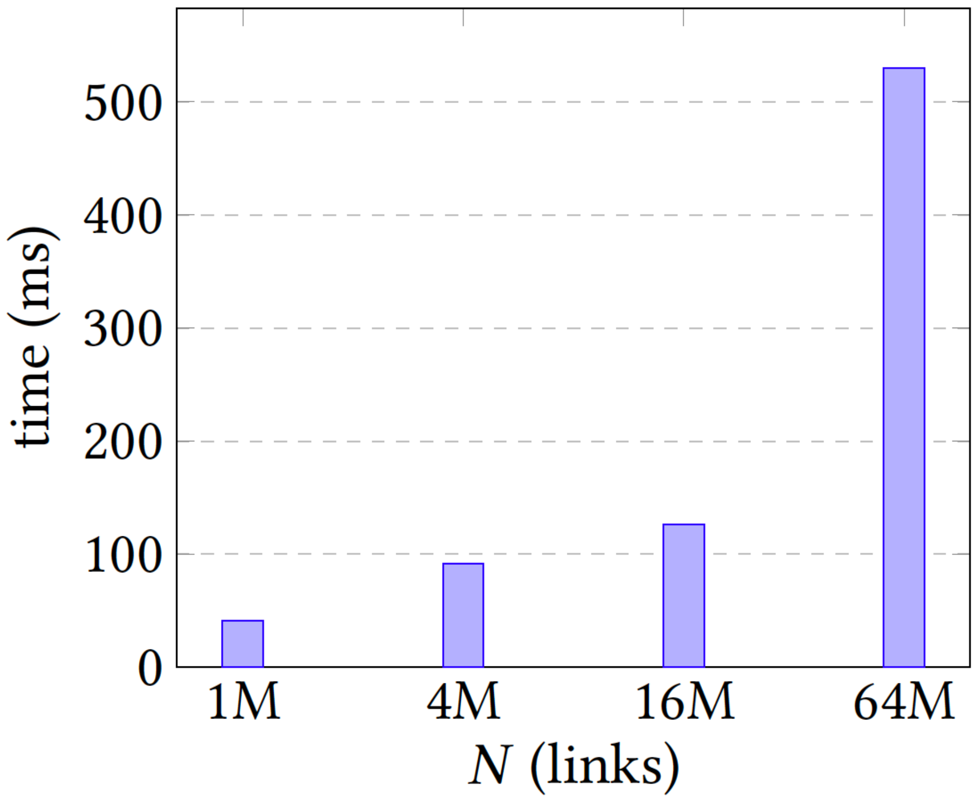
- Implementation written in Go
- Server run on AWS instance, client run on Google Pixel 6 phone
- Simulate joining a small group with 10 unknown users, each with 10 key updates





Experiments: Update

- Implementation written in Go
- Server run on AWS instance, client run on Google Pixel 6 phone

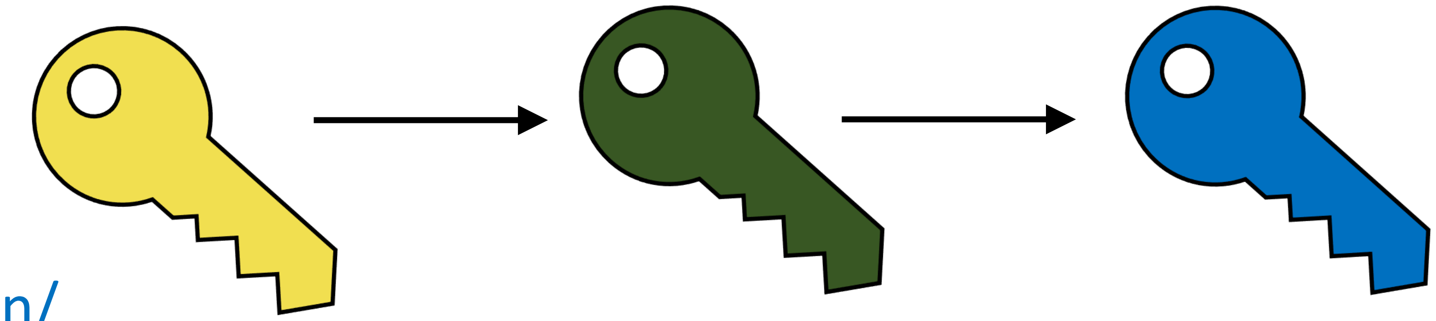


- In the graph, we measured how long it takes to add 10 random key updates for various directory sizes
- Our experiments also show that ELEKTRA can add **128 keys in about a second** to a directory containing **64M keys**
- PCSUpdate for a directory of 4M keys takes about 30 minutes

Key transparency

System	Strong privacy guarantees	Post-compromise security	Strong multi-device security with account resets	Rigorous security analysis	Efficient
CONIKS [MBBFJ Sec’15]	✗	✗	✗	✗	✓
SEEMless [CDGM CCS’19]	✓	✗	✗	✓	✓
Merkle^2 [HHKYP SP’21]	✗	✗	✗	✗	✓
Parakeet [MKS+ NDSS’23]	✓	✗	✗	✓	✓
ELEKTRA	✓	✓	✓	✓	✓

Conclusion



Email: jlen@cs.cornell.edu
https://www.cs.cornell.edu/~jlen/

- Formally model and construct ELEKTRA: the first key transparency system with strong multi-device support
- First key transparency system with post-compromise security for privacy guarantees
- Rigorous security definitions!
  - Completeness, Soundness, Privacy
- Experiments show our protocol is efficient for real-world loads