02233 – Network Security

Week 10: Private Communication (continued)

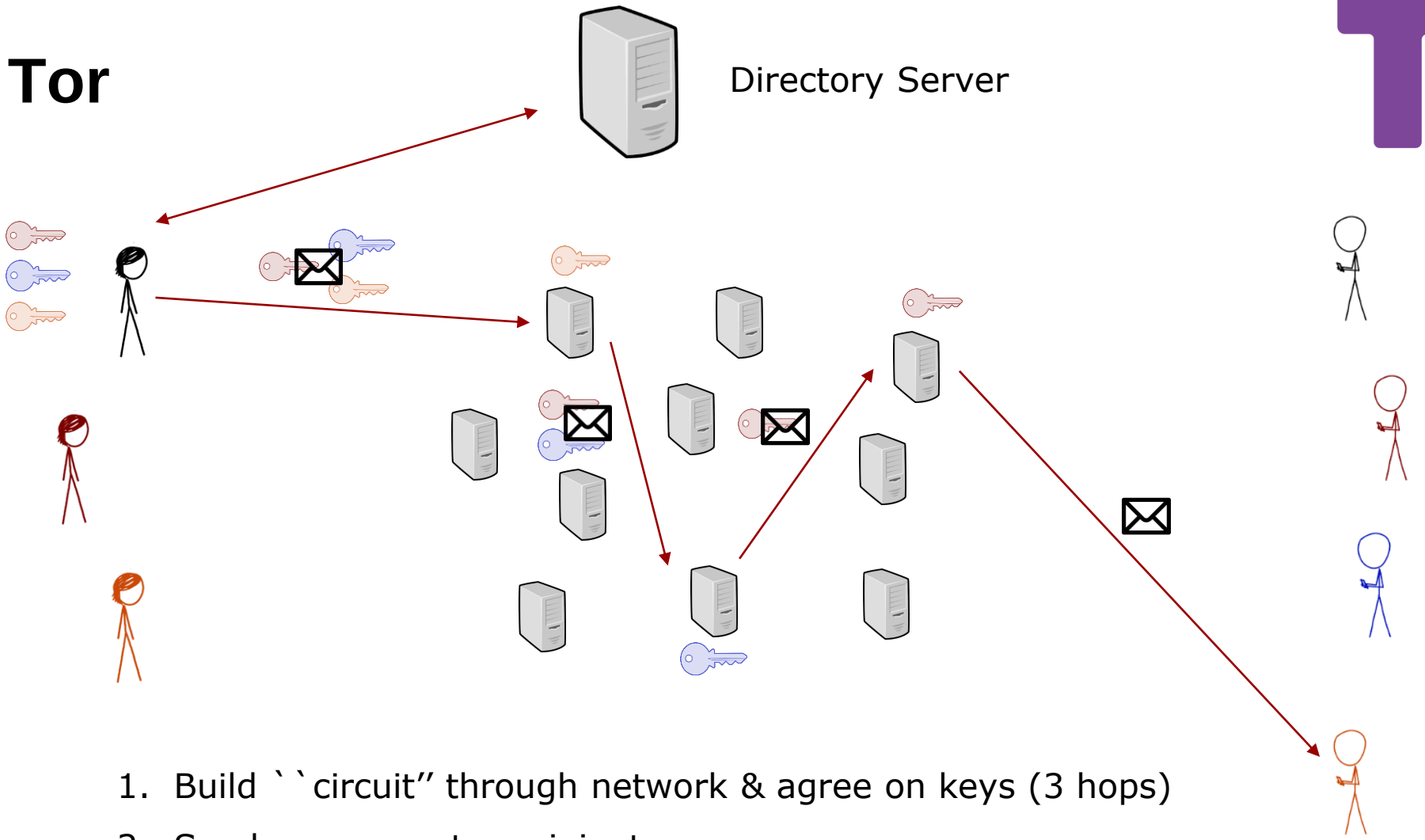
Carsten Baum
Associate Professor
cabau@dtu.dk

Schedule for today

1. A bit more Onion Routing

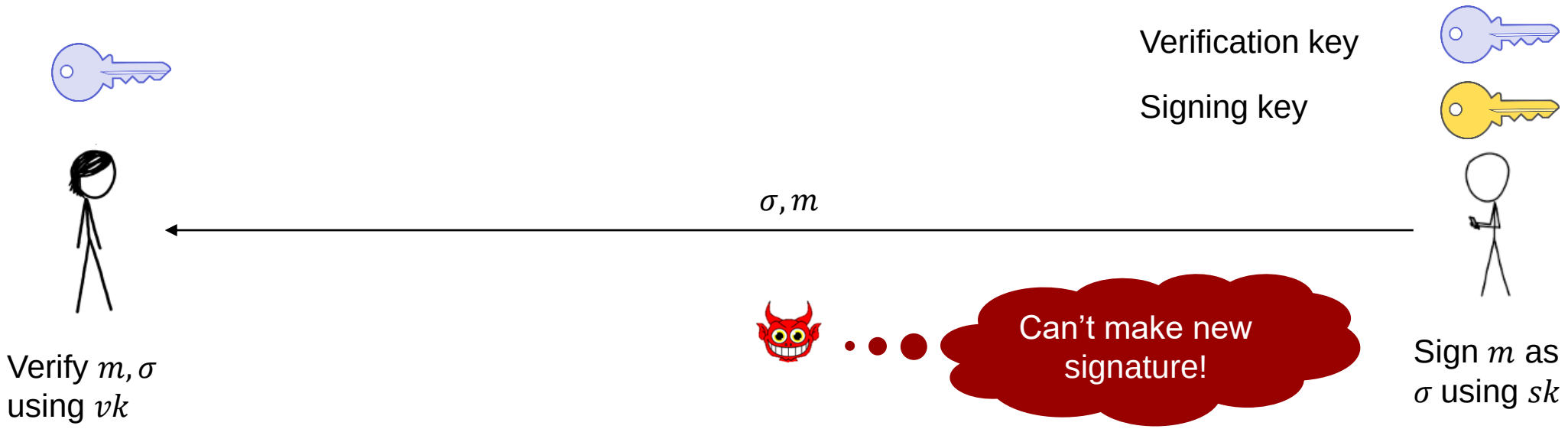
2. Private Messaging with the Signal Protocol

Tor



1. Build ``circuit'' through network & agree on keys (3 hops)
2. Send messages to recipient

Digital Signatures



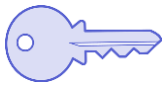
Tor Hidden Services

- Previously: anonymous access to a public server/service
  – Alice is anonymous but the server is public

- Tor also supports **hidden servers/services** (aka dark web)
  – Alice is anonymous, the server is also anonymous, Tor helps them talk!

- Alice needs ``some'' identifier for the service

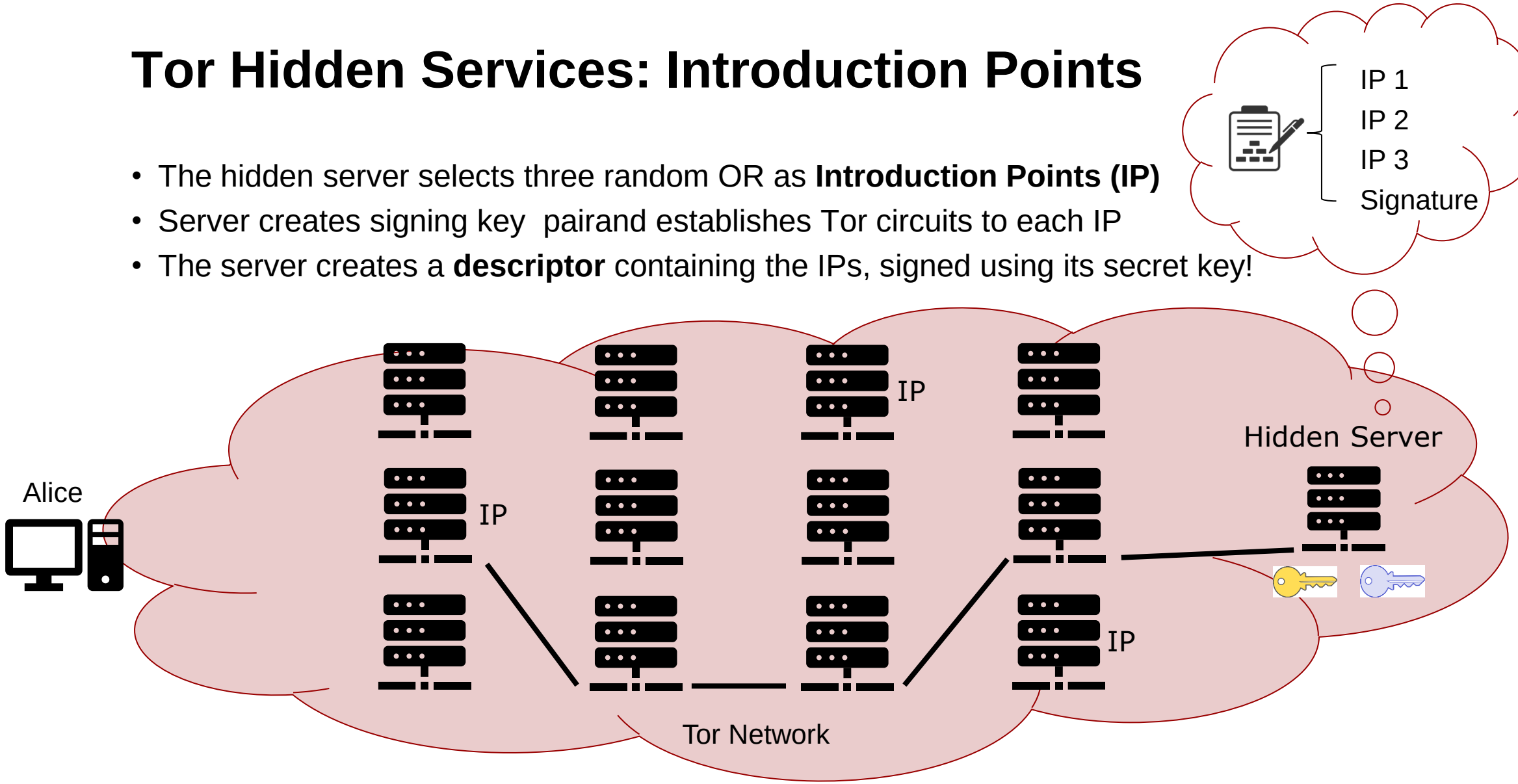
vww6ybal4bd7szmgncyruucpgfkqahzddi37ktceo3ah7ngmcopnpyyd.onion



Public key of Signature scheme,
server keeps signing key

Tor Hidden Services: Introduction Points

- The hidden server selects three random OR as **Introduction Points (IP)**
- Server creates signing key pairand establishes Tor circuits to each IP
- The server creates a **descriptor** containing the IPs, signed using its secret key!

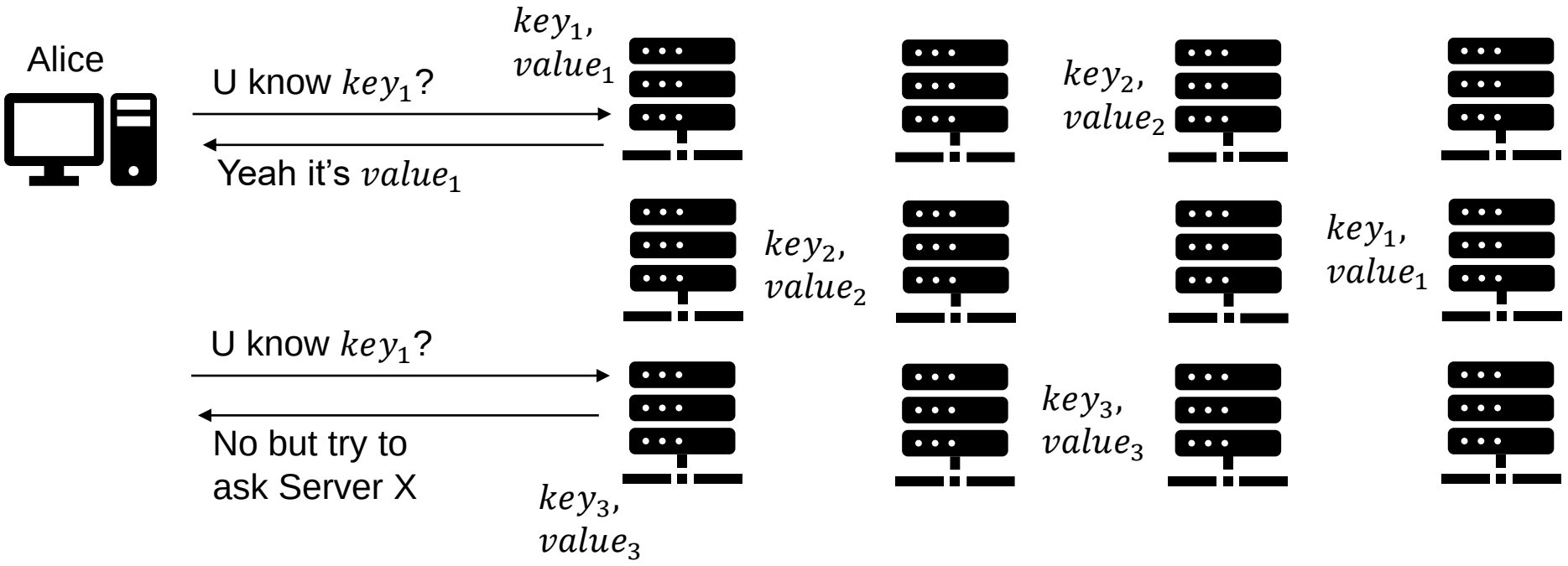


How to get descriptor to Alice: Distributed Hash Table

Distributed protocol to identify parties responsible for $(key, value)$ pairs.

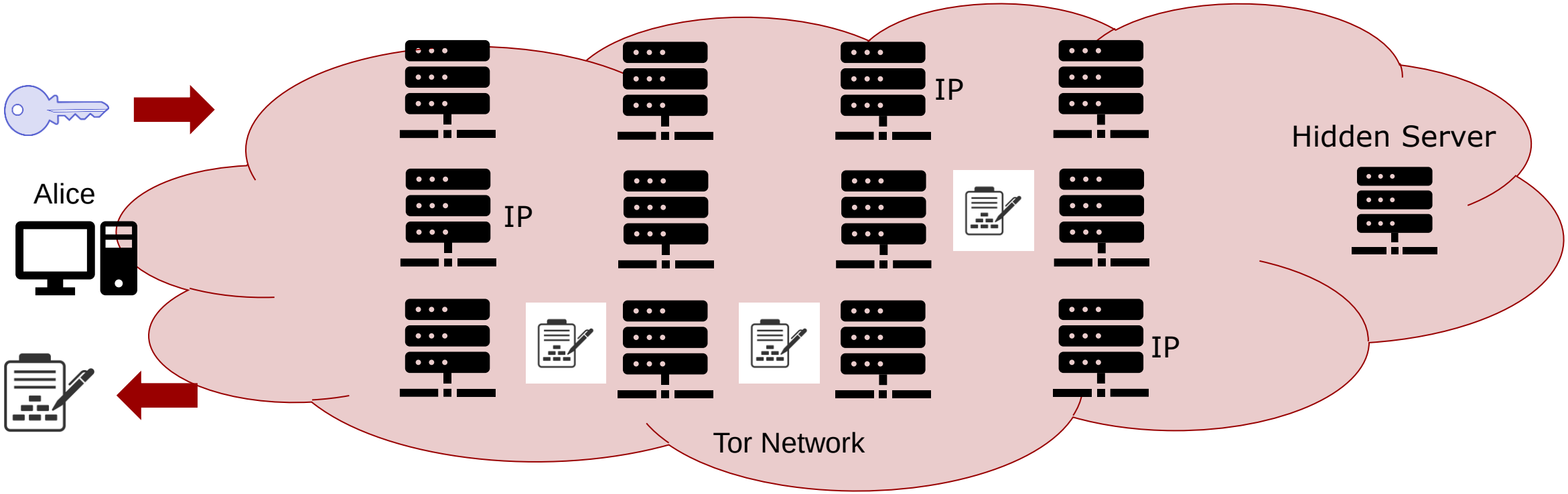
Input for lookup: key

Then: communication in network to find a server responsible for key who knows $value$



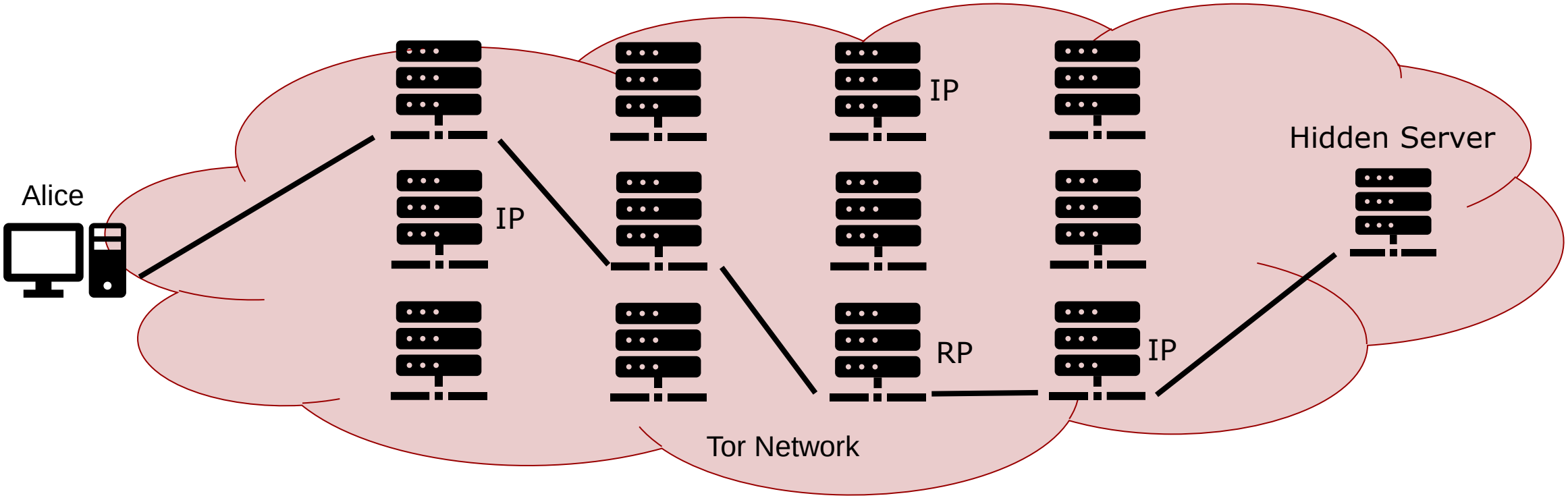
Tor Hidden Services: Finding the IPs

- Server uses DHT to identify nodes responsible for pk, then gives them descriptor
- If Alice looks for service, she identifies same nodes using pk via DHT, gets descriptor and validates signature using pk
- From descriptor, Alice can then identify IPs



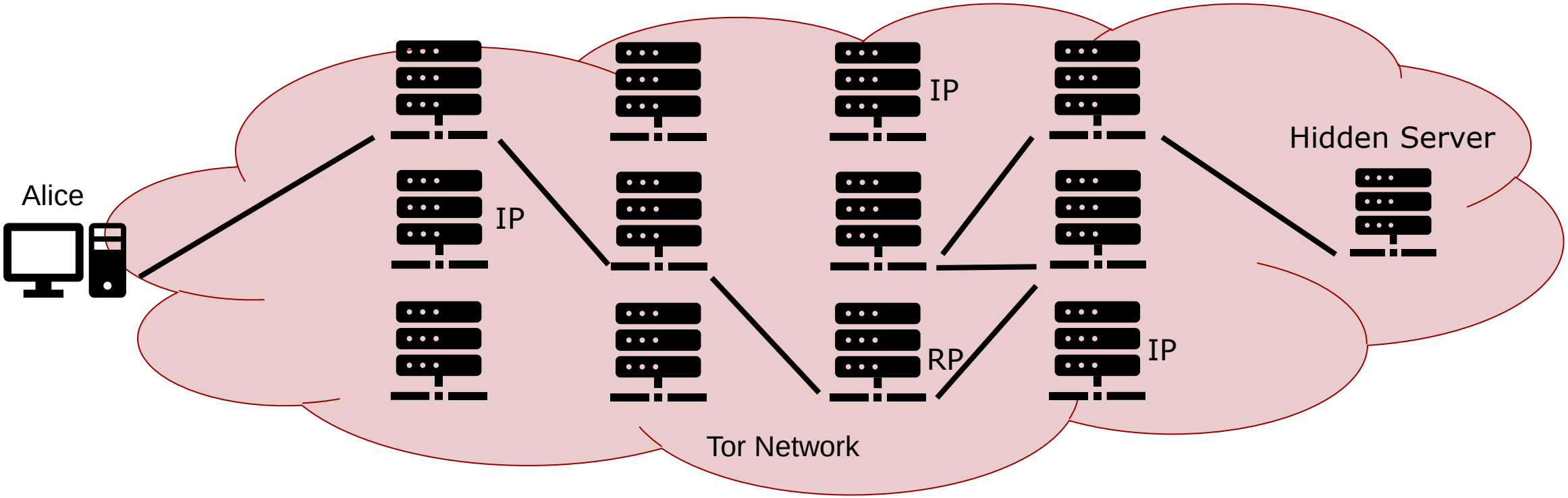
Tor Hidden Services: Rendezvous Point

- Alice makes a 3-hop Tor circuit to a random OR: **The Rendezvous Point (RP)**
- Alice generates a random id and asks a random IP to pass it to the hidden server (via the RP)
- The introduction is **via 3 Tor circuits**: Alice-RP, RP-IP, IP to Hidden Server



Tor Hidden Services: Circuit Establishments

- The Hidden Server can choose to accept or deny the random ID
- If accepted, it creates a 3-hop TOR circuit to the Rendezvous Point (RP)
- At this point the RP can match the random ID into the same **7-hop Tor circuit**

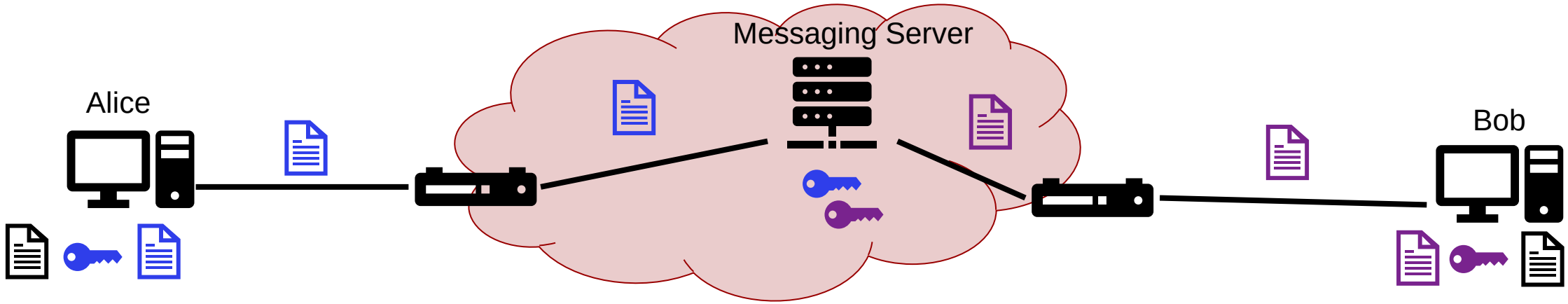


End-to-End encryption for messaging

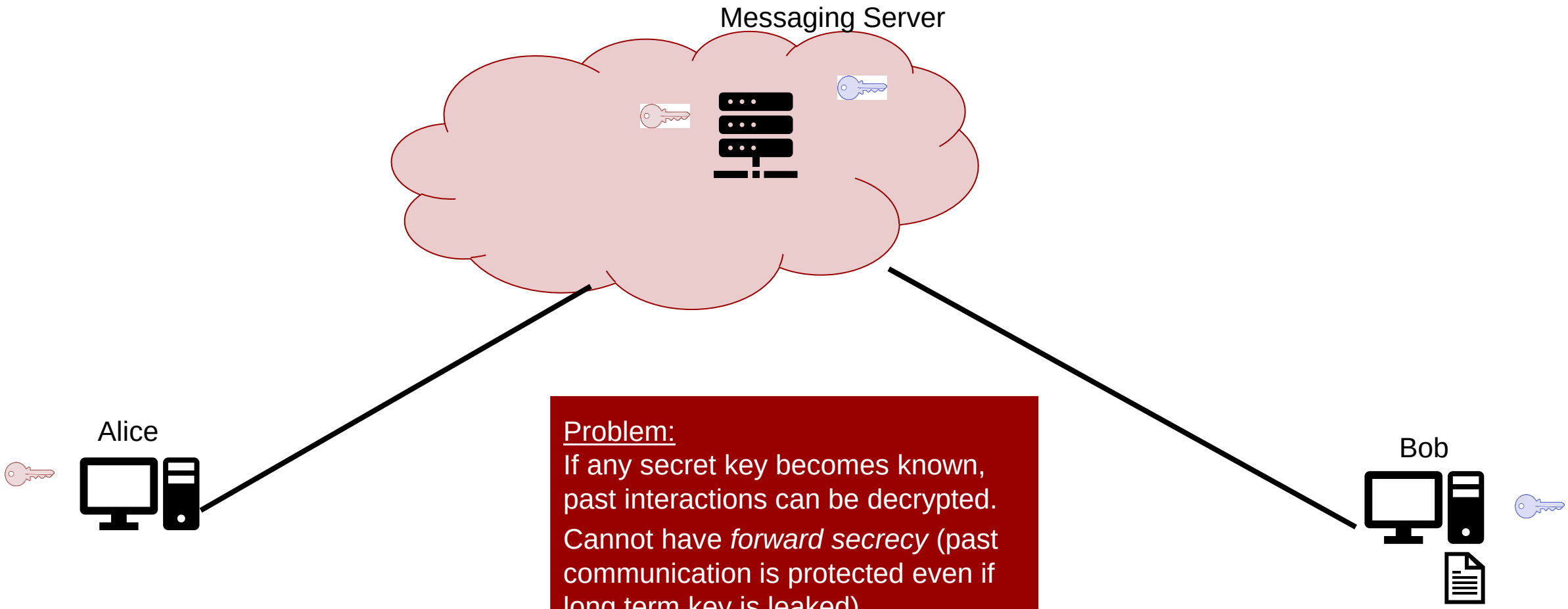
The Signal protocol

Messaging Applications

- Alice and Bob wish to communicate **asynchronously** (send messages when other is offline)
- Alice makes connection to Messaging Server over TLS
- Server stores the message waiting for Bob to come online
- When Bob is online, the server sends the message to Bob over TLS
- Not really E2E encryption, as the server can read the message!



Store public keys on server

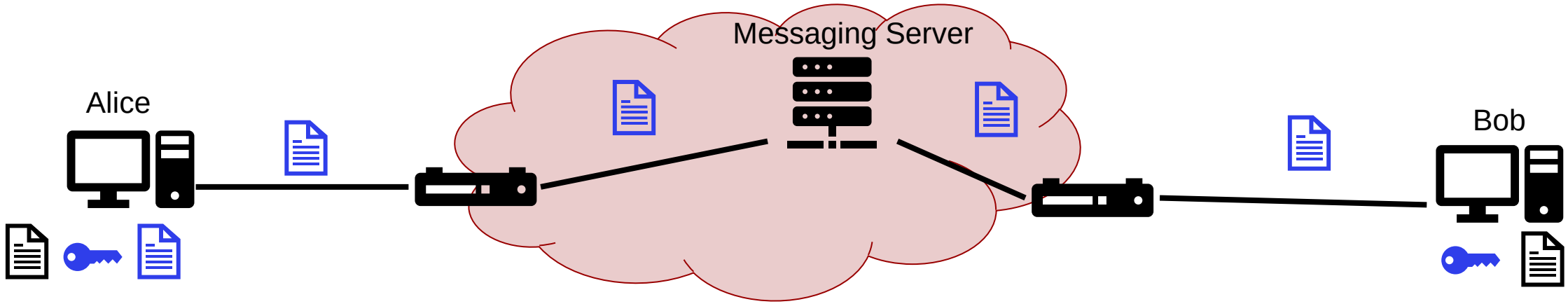


Problem:
If any secret key becomes known, past interactions can be decrypted.

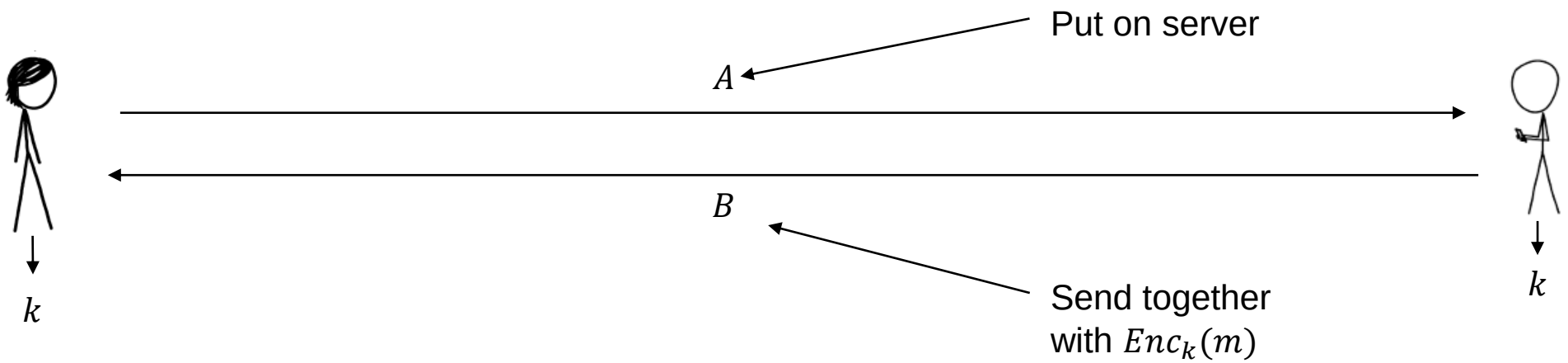
Cannot have *forward secrecy* (past communication is protected even if long term key is leaked).

Messaging Applications

- Alternative: Alice and Bob agree on key first
- Difficult if both are not online simultaneously…

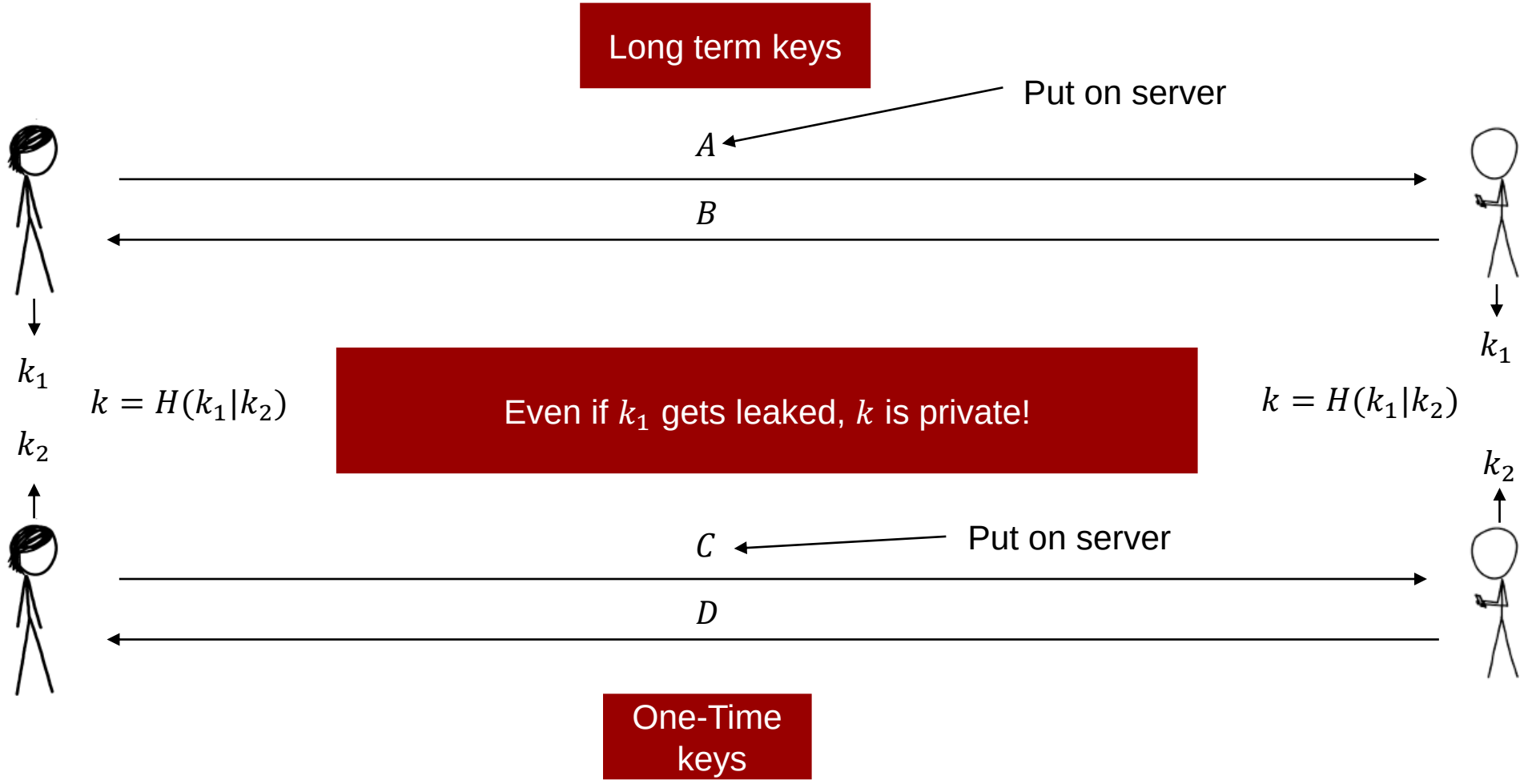


Alice and Bob can put first message on server!



Does not solve problem! What if Alice's secret becomes known?

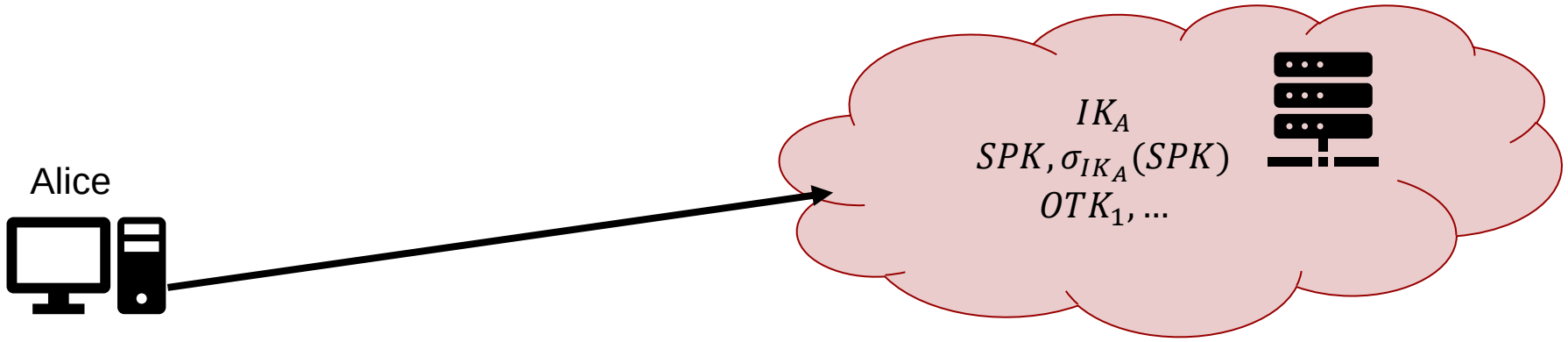
Idea:
Combine multiple key exchanges



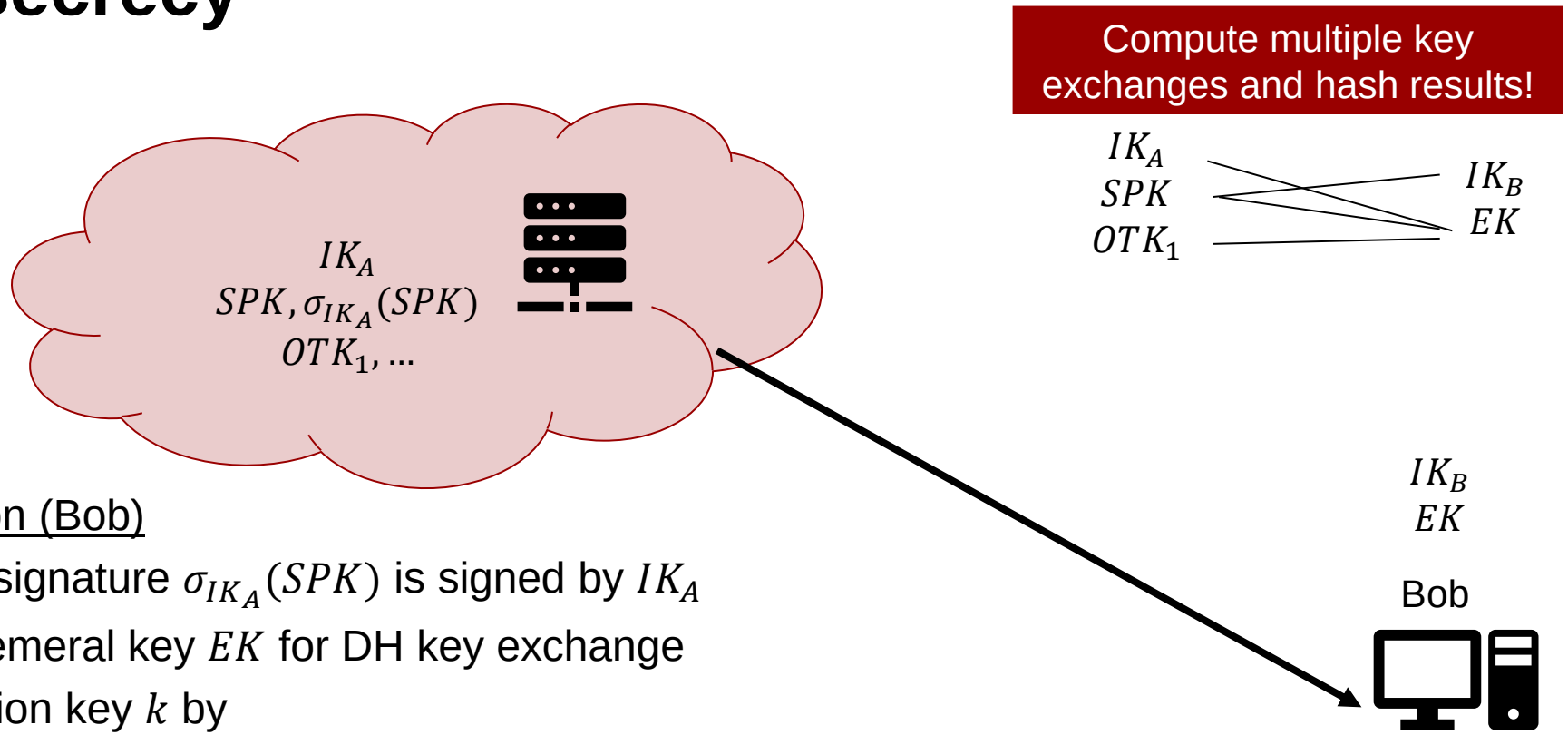
Signal: Asynchronous Key Establishment with forward secrecy

Signal builds on Diffie-Hellman Key Exchange

- Alice sends to server:
  - Identity Public Key IK_A
  - Signed Prekey $SPK, \sigma_{IK_A}(SPK)$
  - several One-Time Public Keys $OTK_1, \ldots$
- Bob wants to communicate with Alice, but let's say Alice is offline
- Bob receives from server $IK_A, SPK, \sigma_{IK_A}(SPK)$ and one One-Time key OTK_1



Signal: Asynchronous Key Establishment with forward secrecy



Key Generation (Bob)

– Check SPK signature $\sigma_{IK_A}(SPK)$ is signed by IK_A

– Create Ephemeral key EK for DH key exchange

– Create session key k by

- computing multiple instances of DH key exchange between the IK_A, IK_B, EK, SPK and OTK_i
- Hash outcomes of key exchanges to obtain session secret k

Signal: Asynchronous Key Establishment with forward secrecy

Key Generation (Bob)

– Check SPK signature $\sigma_{IK_A}(SPK)$ is signed by IK_A

– Create Ephemeral key EK for DH

– Create session key k by

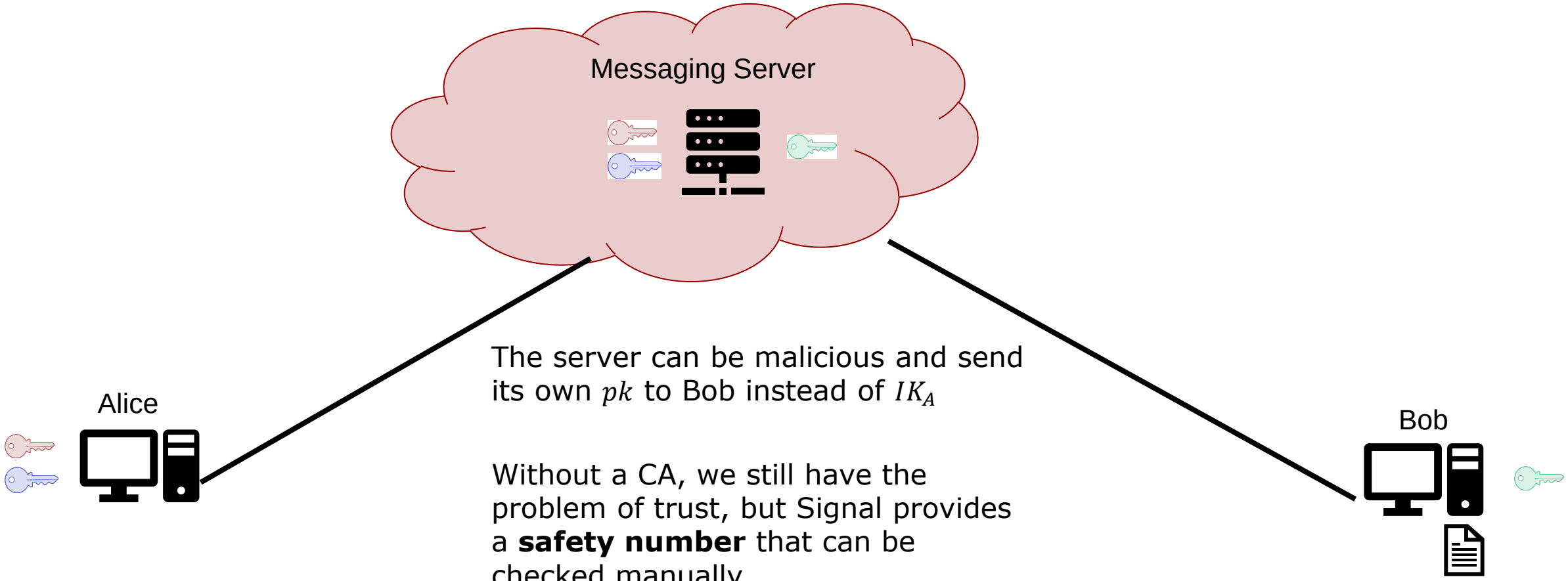
- computing multiple instances of DH key exchange between the IK_A, IK_B, EK, SPK and OTK_i
- Hash outcomes of key exchanges to obtain session secret k

– Bob uploads his IK_B to server and sends EK to Alice

If one-time key OTK_i is used only once, perfect forward secrecy.

SPK is updated regularly and can protect forward secrecy after update, if no more OTK available.

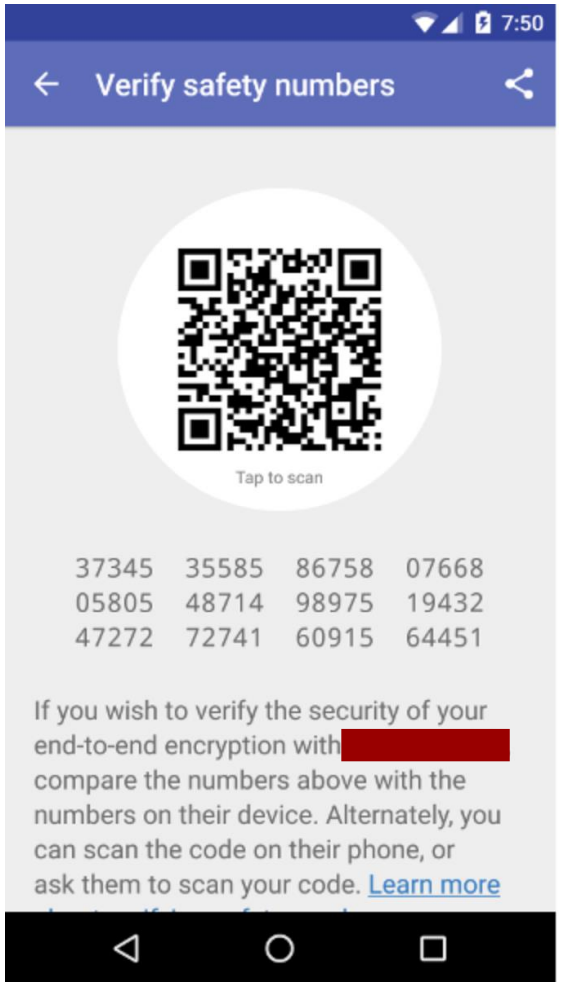
But how does Bob know those are Alice's keys?



The server can be malicious and send its own pk to Bob instead of IK_A

Without a CA, we still have the problem of trust, but Signal provides a **safety number** that can be checked manually.

Signal Protocol: Safety Number



Also in Whatsapp
View Contact -> Encryption