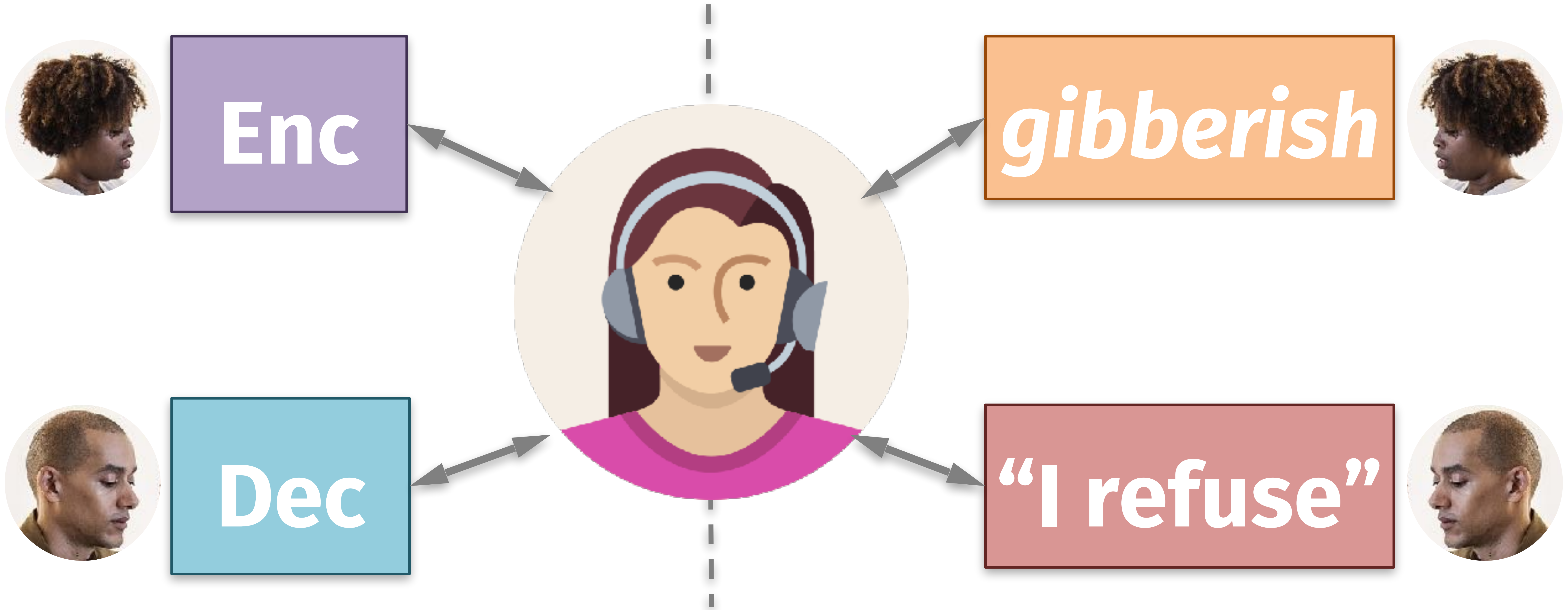
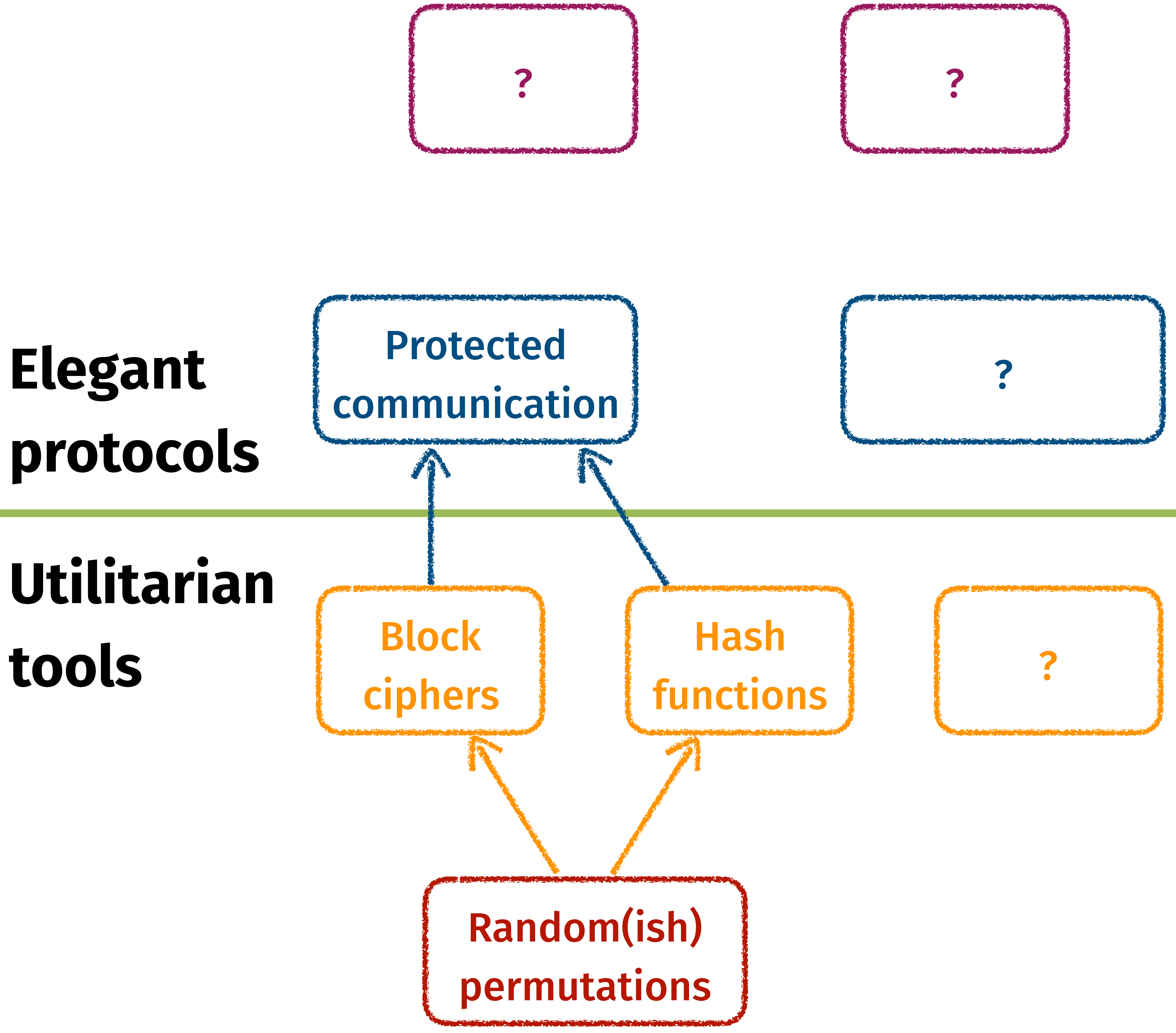
Lecture 14: Timely key exchange

- Lab 6 due Monday 3/25 at 11pm
- Guest lectures on cryptography + law starting 1 week from today

Recall: Authenticated encryption



Course roadmap



Confidentiality

Integrity

Availability

Confidentiality

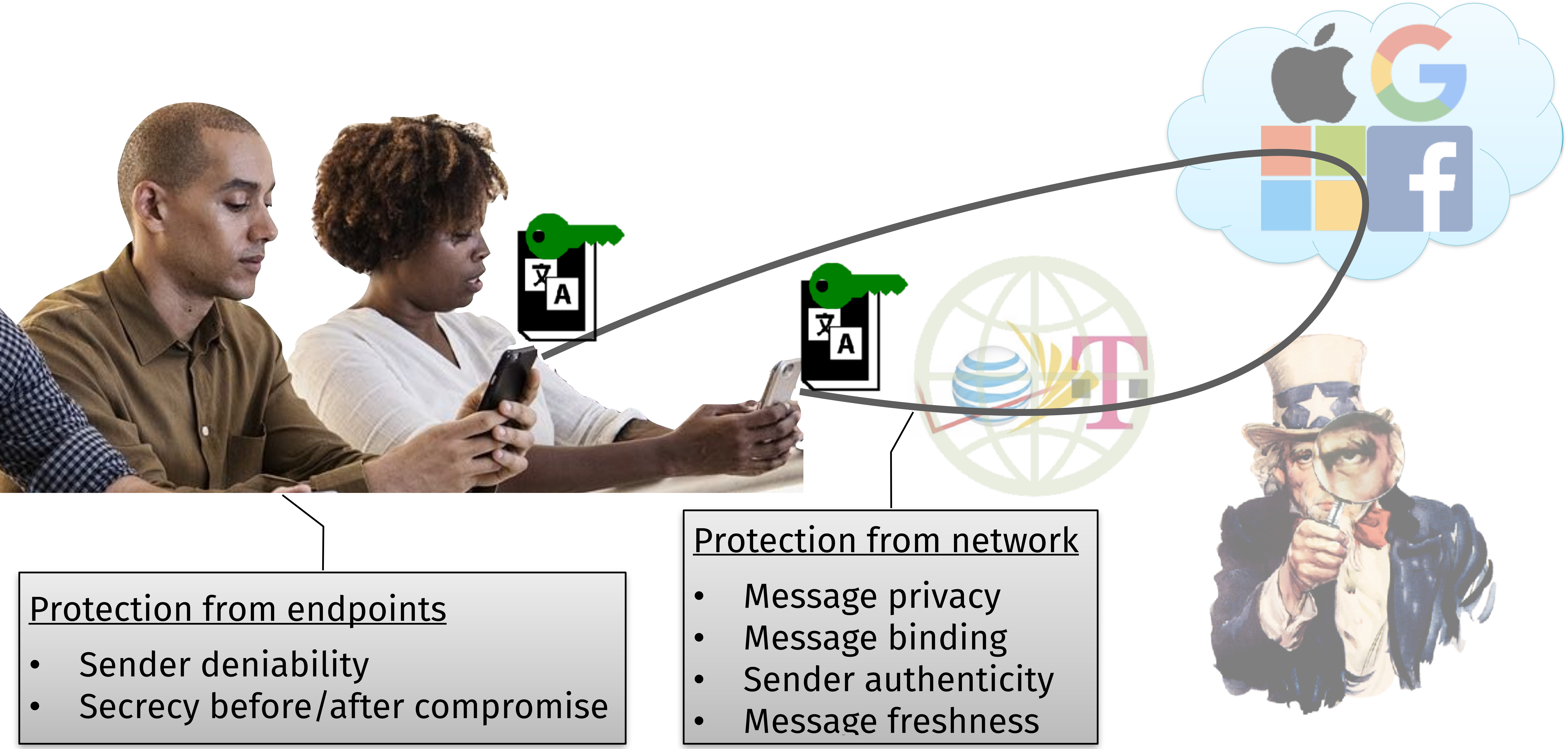
- Private ✓
- Deniable ?
- Withstand device compromise ✗

Integrity

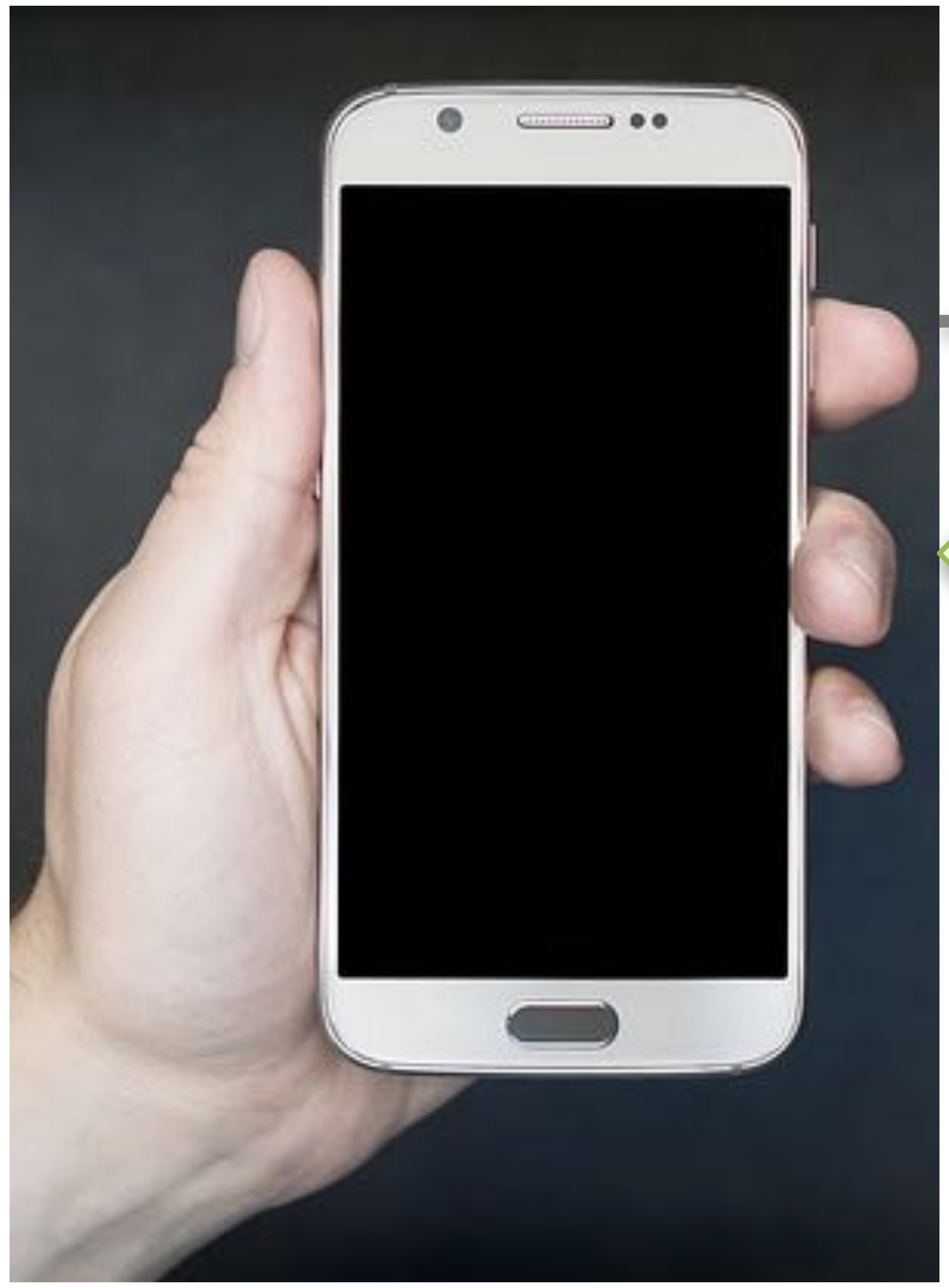
- Authenticated ✓
- Binding / non-malleable ✓
- Fresh ✗

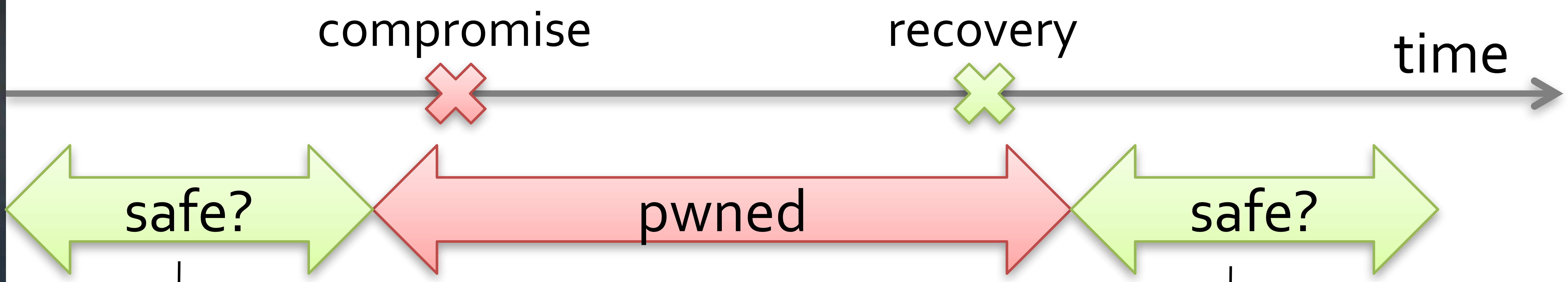
Availability

End-to-end crypto over the Internet



Forward and backward secrecy

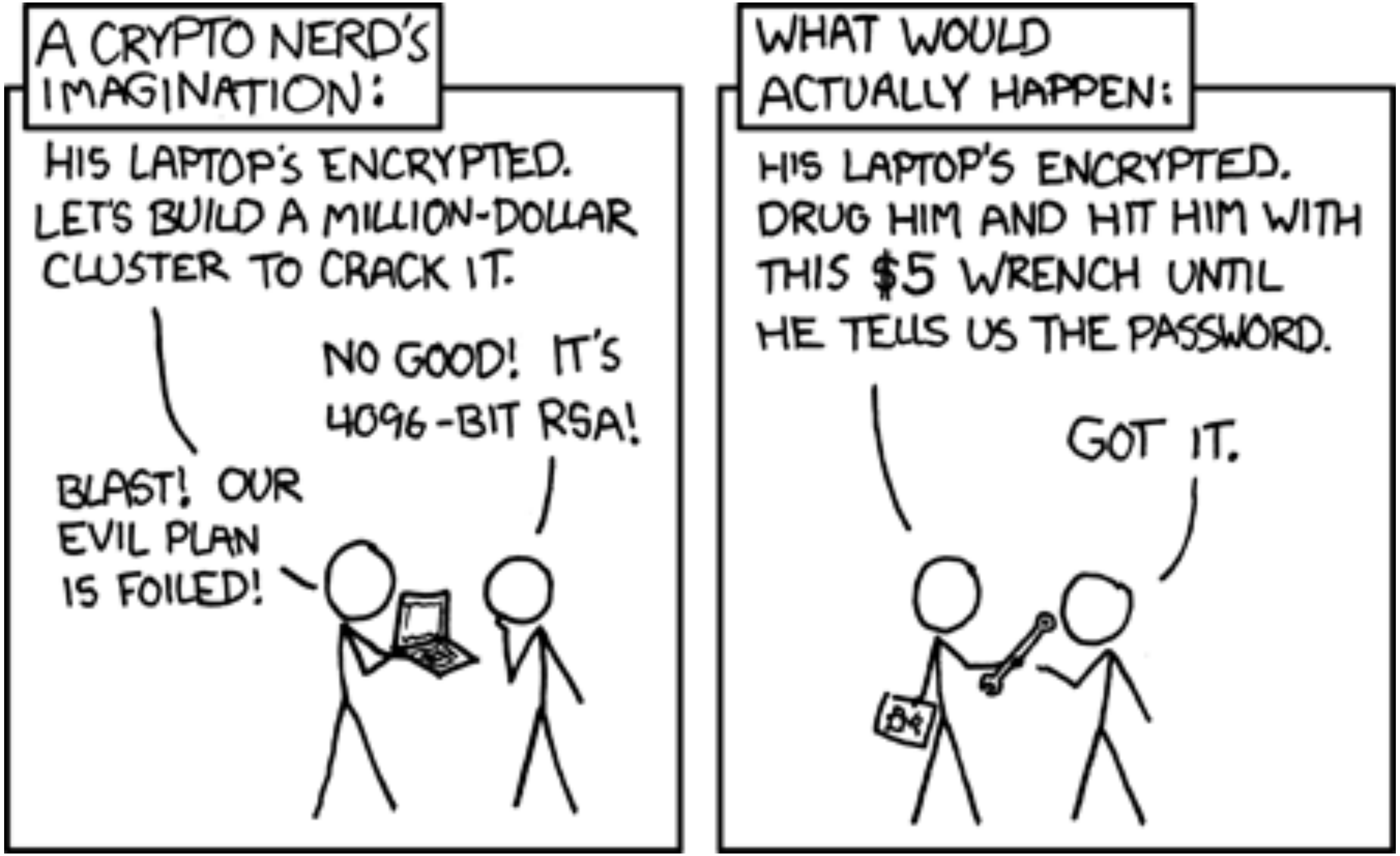




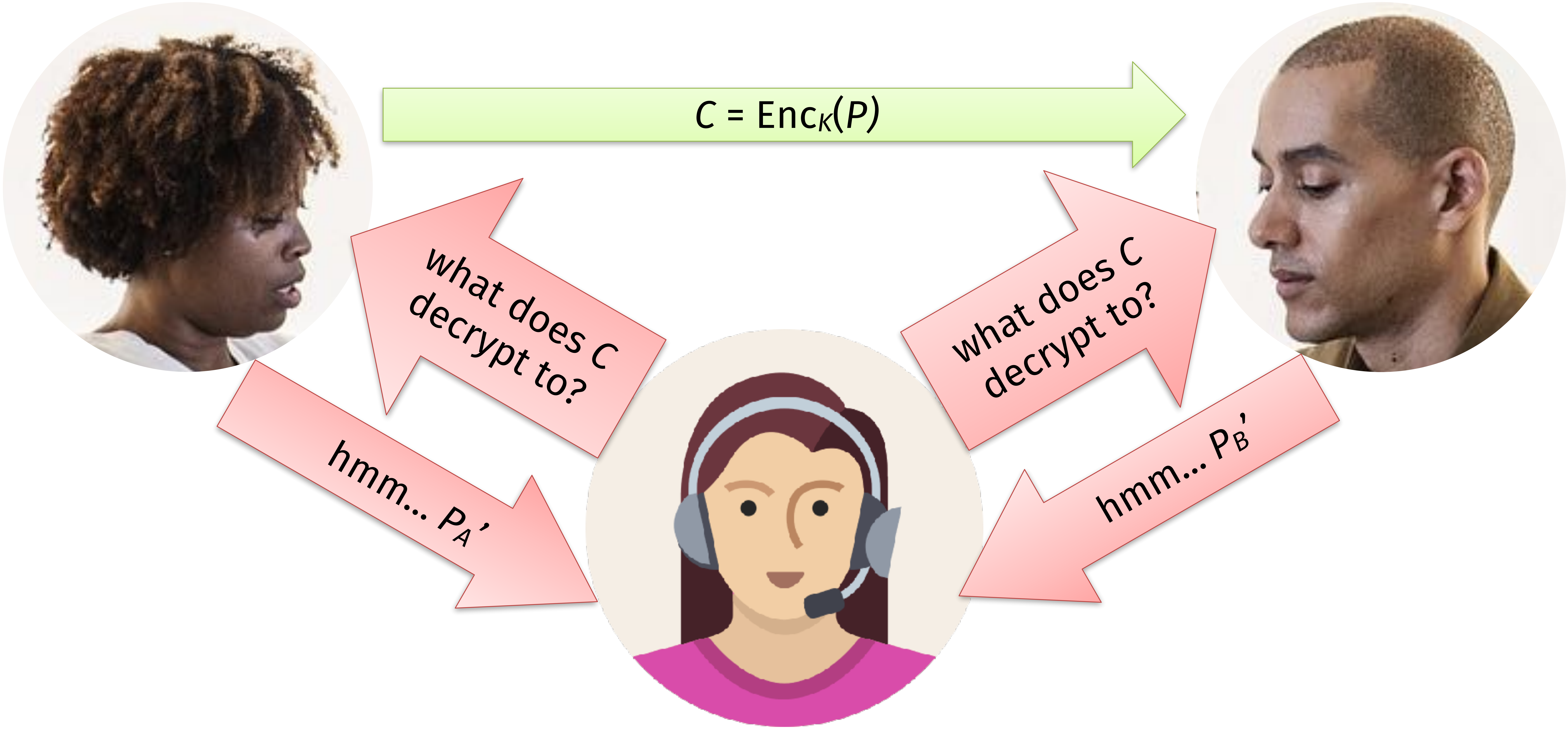
Forward (aka pre-compromise) secrecy:
past messages remain confidential even if secret state is exposed later

Backward (aka post-recovery) secrecy:
future messages remain confidential even if secret state is compromised and then the device recovers

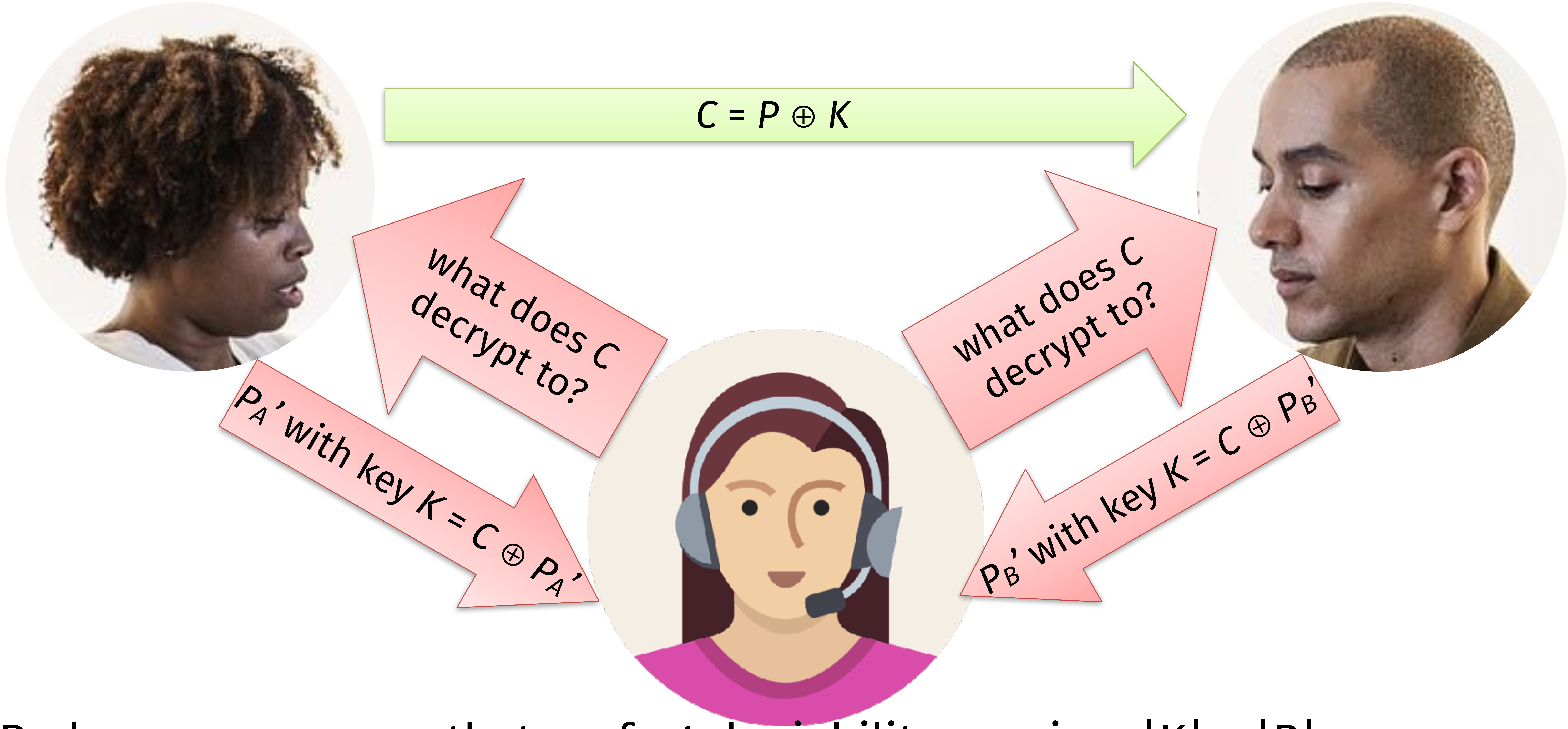
Non-repudiable crypto (xkcd.com/538)



Deniable crypto = can pretend you said something else

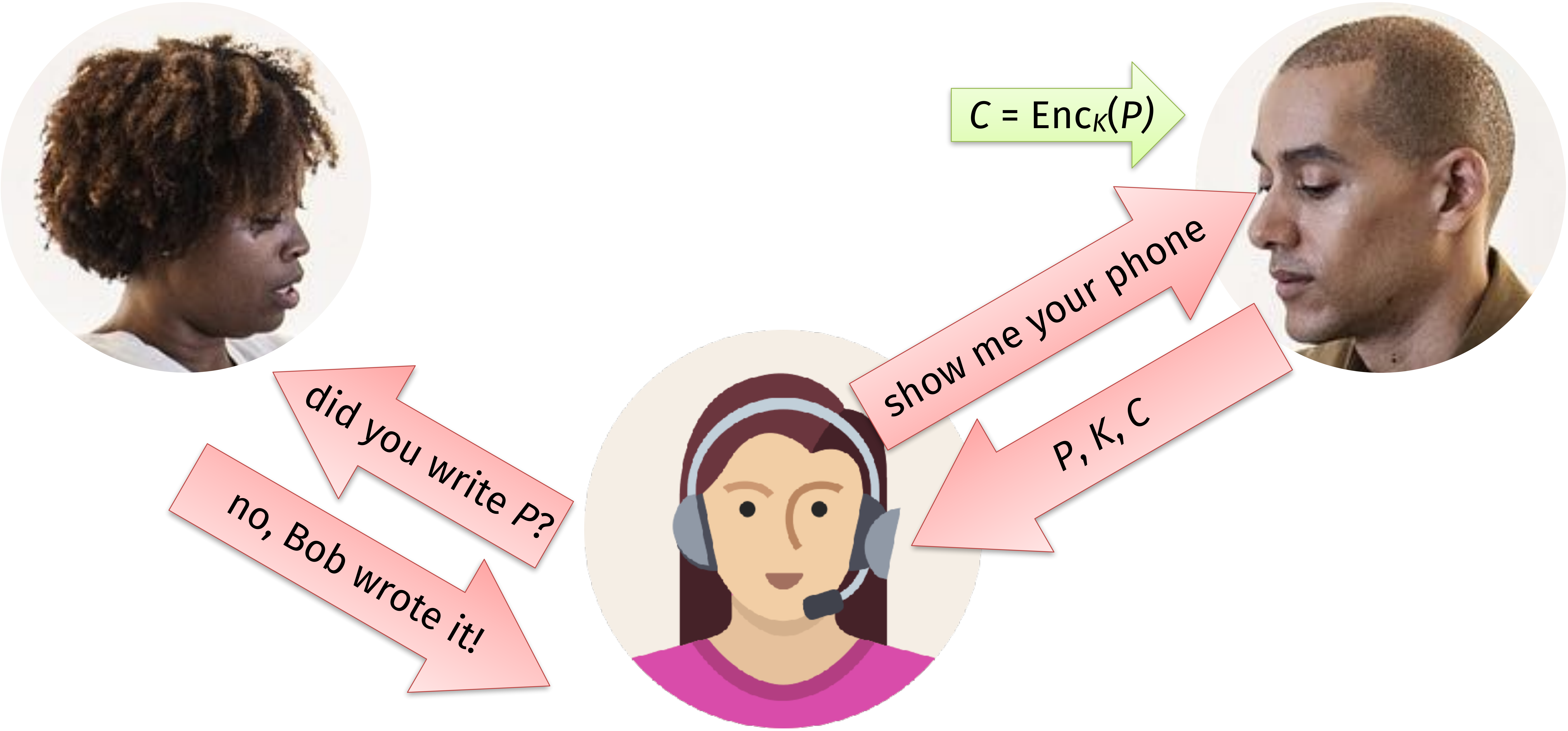


One time pad → perfect deniability

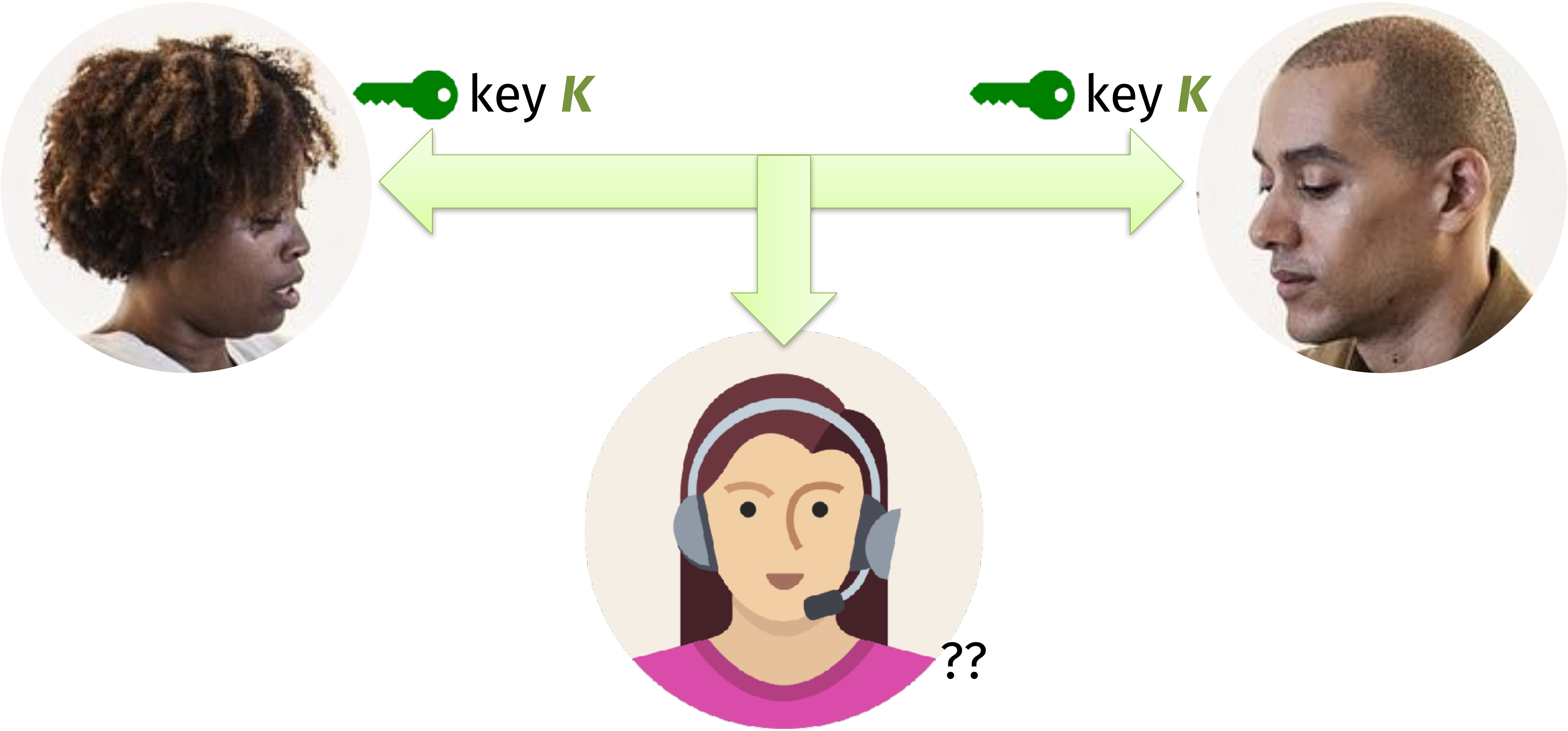


Bad news: can prove that perfect deniability requires $|K| \geq |P|$

Auth encryption → partial sender deniability



Part 3: Generate, exchange, evolve, and delete keys



Core ideas of Part 3

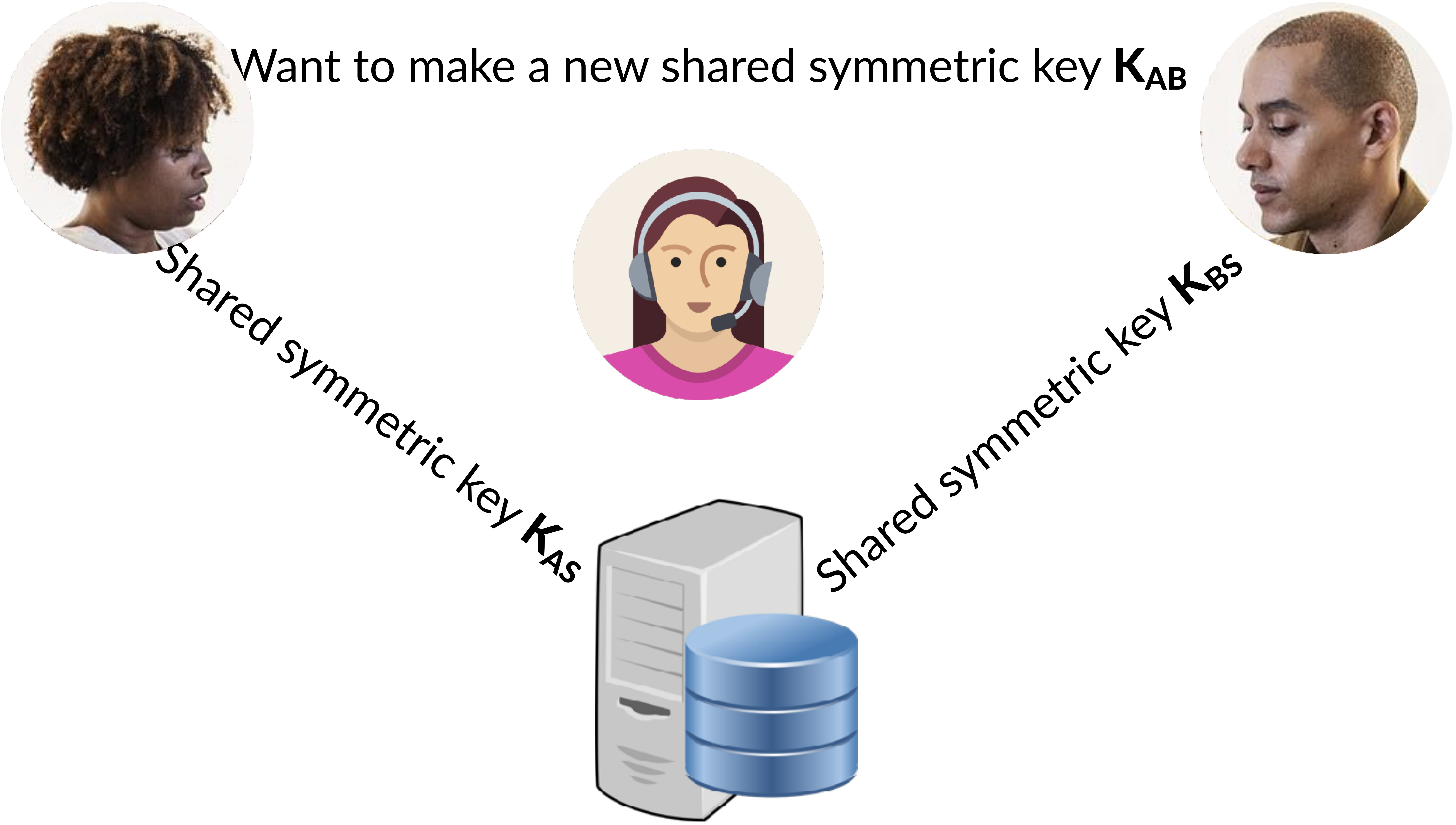
- Key exchange
  - Alice and Bob want to generate a shared key without ever having met before
  - Often, need the help of a (partially) trusted entity to mediate this connection
- Key evolution (aka ratcheting)
  - Use each key to protect just 1 message, then *delete it*!
  - Protect message privacy + integrity against device compromise in past + future
  - Generate a new key for the next message

Key management = initial exchange + subsequent evolution

Server trust?	Crypto used	Method
Full	Symmetric	Needham-Schroeder $\Rightarrow$ Kerberos system
Partial	Asymmetric	(Authenticated) Diffie-Hellman key exchange
None	Symmetric	Key evolution, starting from an initial shared symmetric key

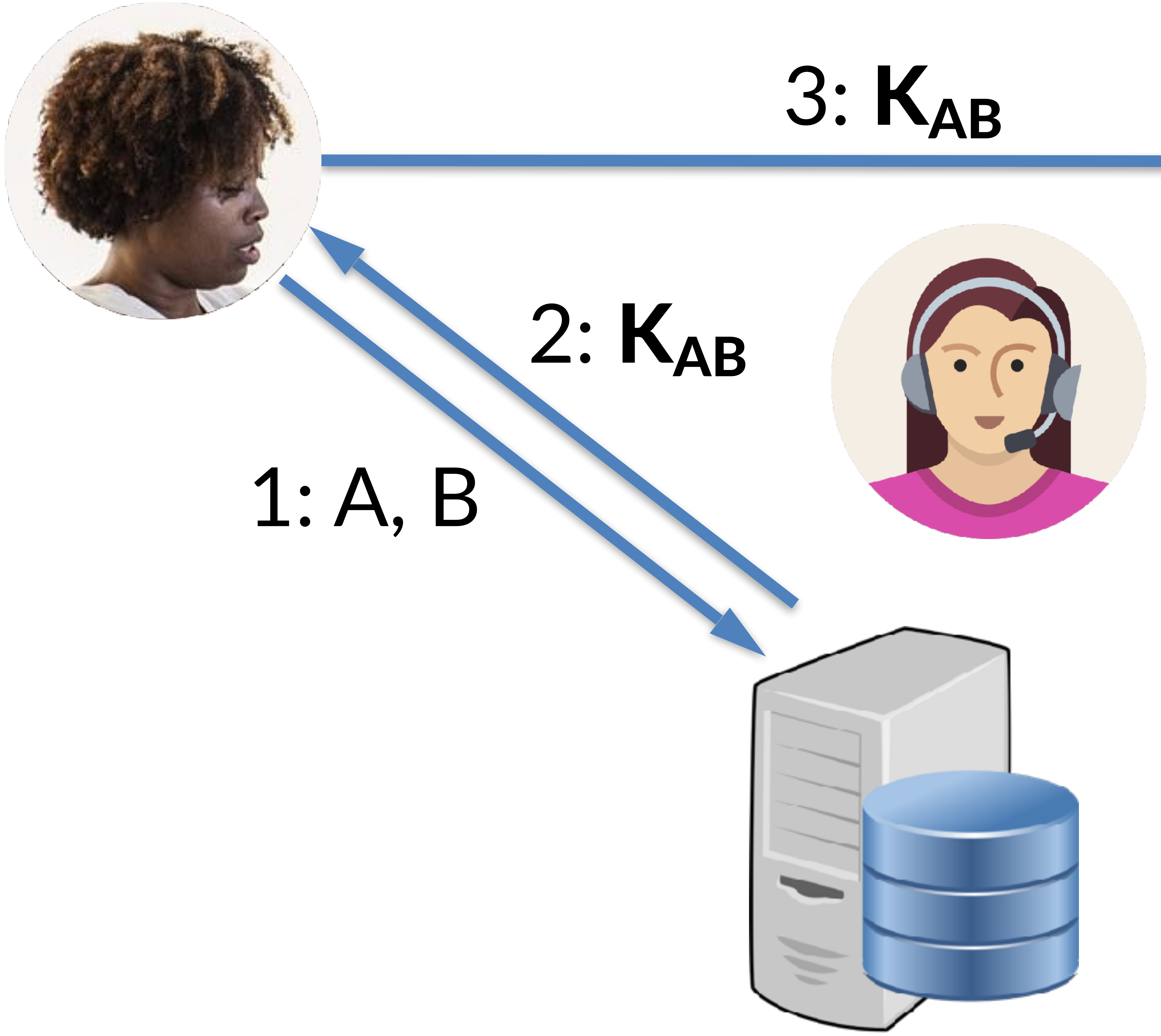
Needham–Schroeder protocol for key transport

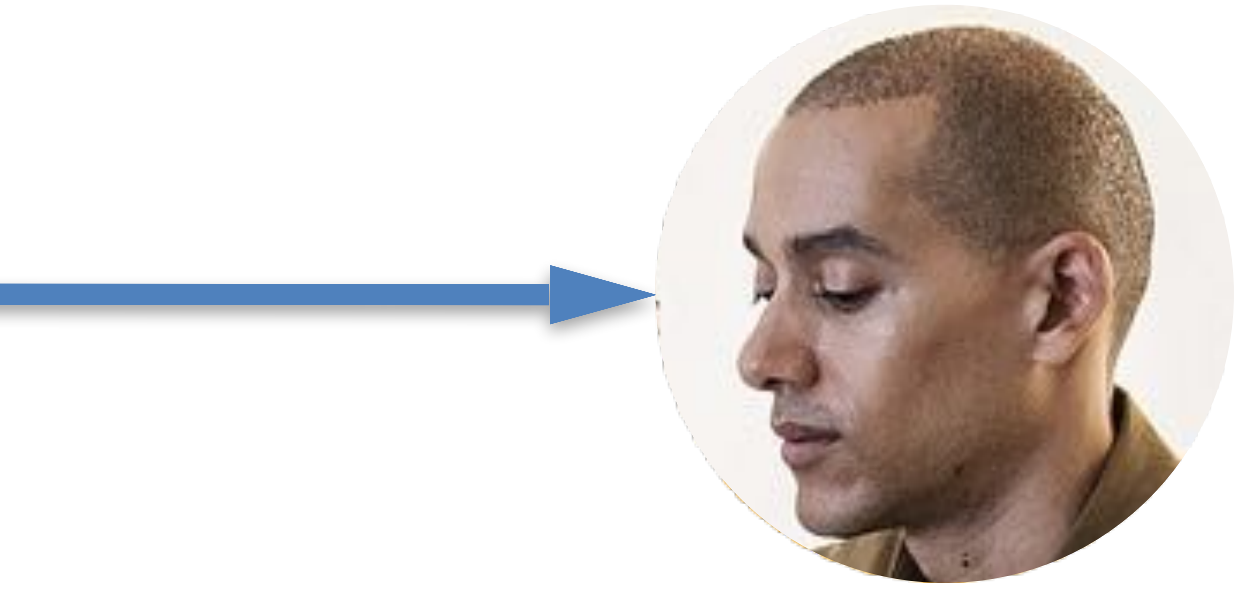
Objective: with the help of a trusted server, Alice + Bob agree on a shared key



Needham–Schroeder: take 1

Basic idea: Let the trusted server choose K_{AB}





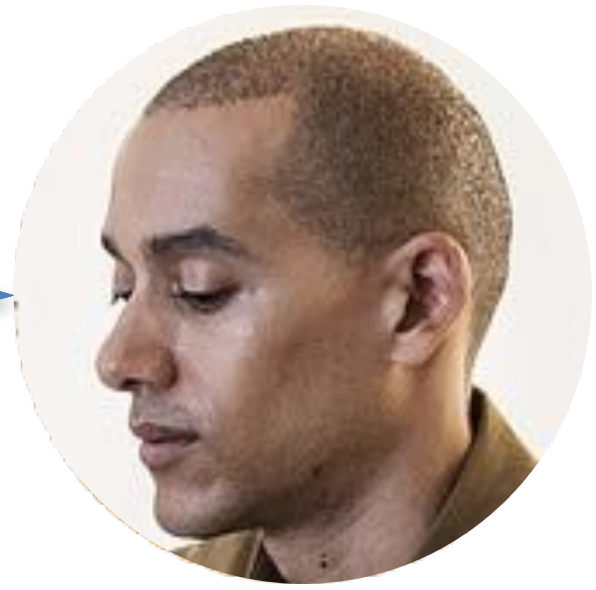
Problems?

- Mallory can read the key
- Alice doesn't know if K_{AB} came from trusted server
- Bob doesn't know this either (even if Alice did)

Needham–Schroeder: take 2

New idea: Authenticate messages from the trusted server

3: A, $\{\mathbf{K}_{AB}\}_{BS}$



2: $\{\mathbf{K}_{AB}\}_{AS}$, $\{\mathbf{K}_{AB}\}_{BS}$

Notation: Protect message contents using this key

1: A, B

Two Problems

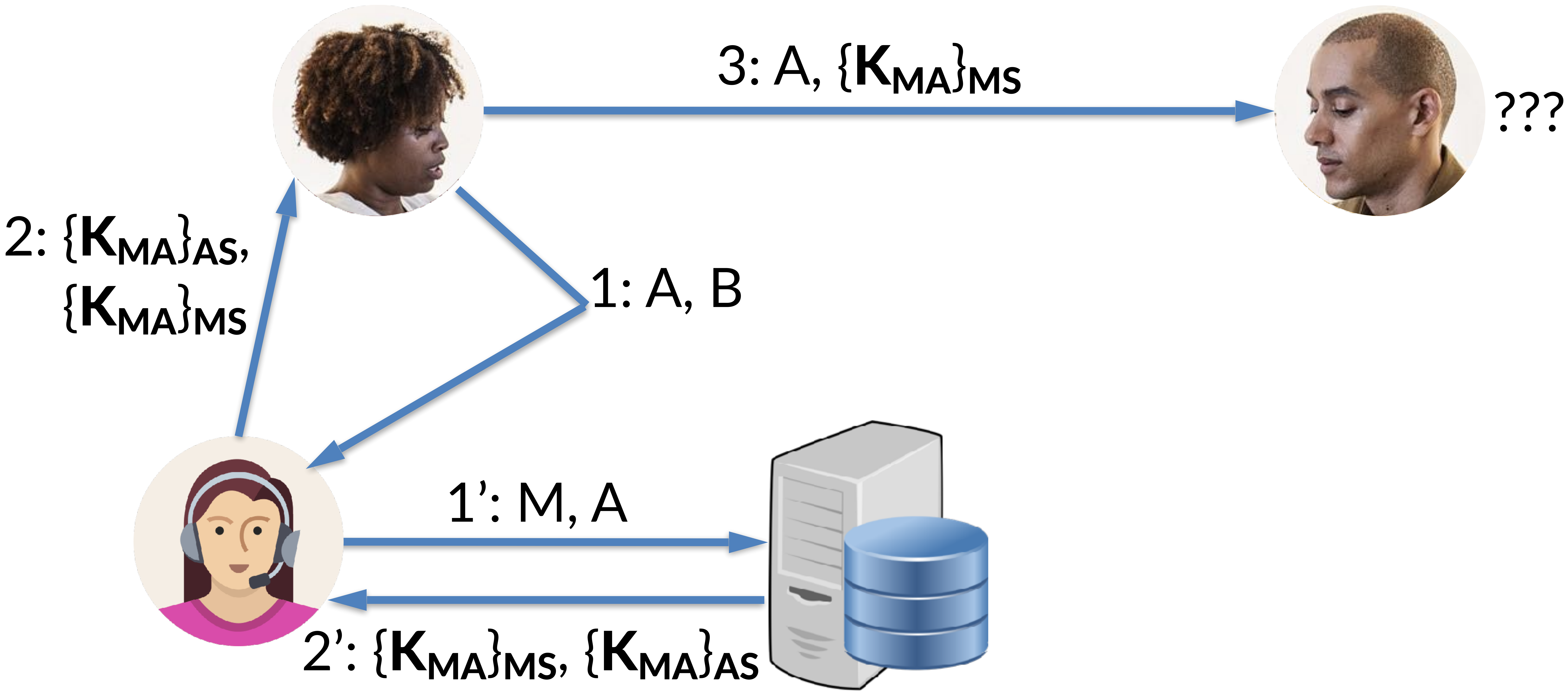
- Bob doesn’t know whether he’s really talking to Alice
- Server is too willing to create messages intended for others

Problem 1 in more detail

Issue: Mallory can con Bob into producing messages that weren’t intended for Alice, and then forward them along anyway

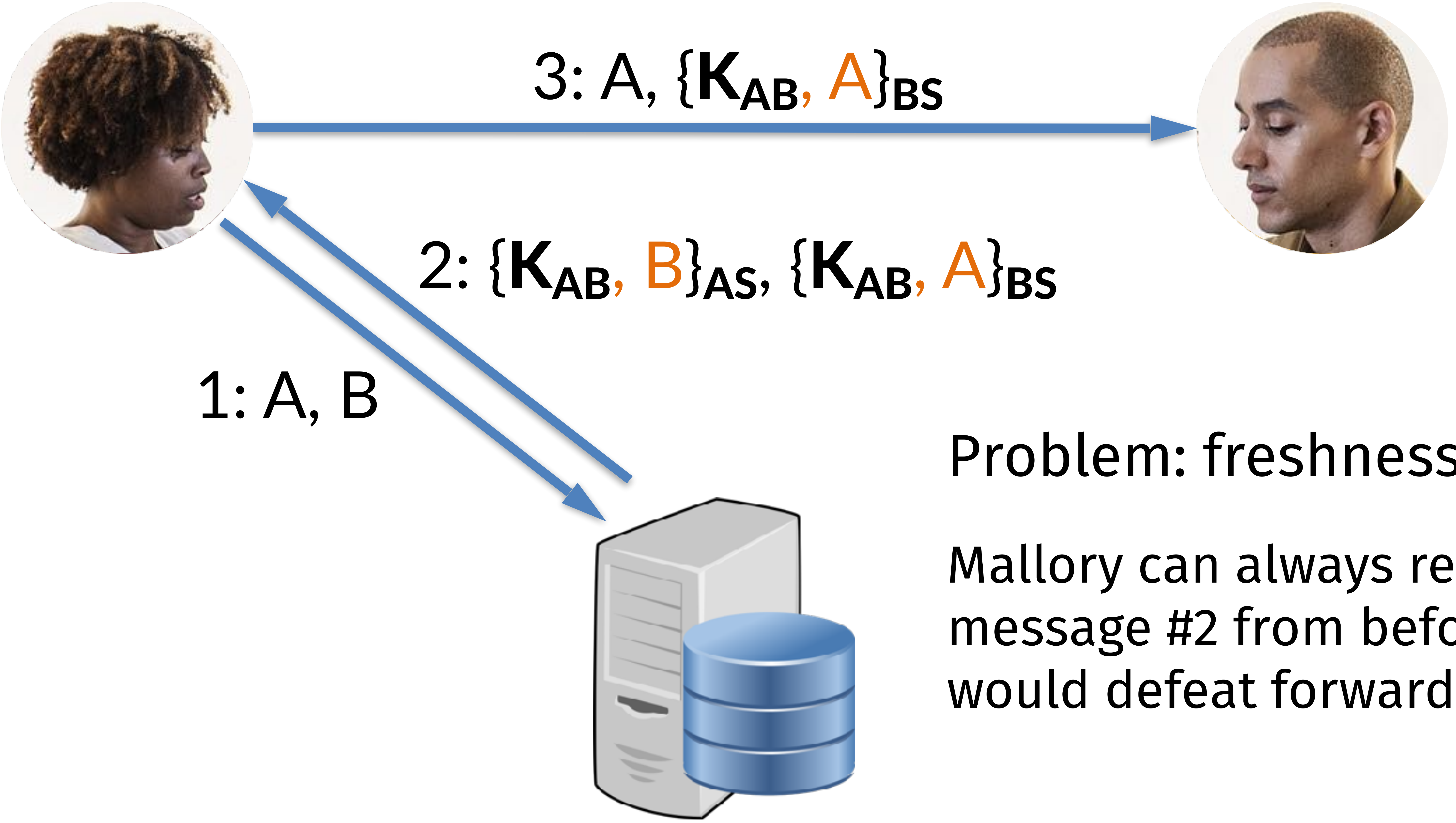
Problem 2 in more detail

Because the server signs messages intended for *other* parties, Mallory can use this capability to emulate the actions of the trusted server!



Needham–Schroeder: take 3

Both problems were due to the fact that people misinterpreted some of the server's responses as intended for other participants. So, have server be explicit.



Problem: freshness

Mallory can always repeat message #2 from before. This would defeat forward secrecy.

Needham–Schroeder: take 4

Use nonces to ensure uniqueness messages 2 and 3 without maintaining state

$3: A, \{\mathbf{K_{AB}}, A\}_{\mathbf{BS}}$

$2: \{ N_A, B, \mathbf{K_{AB}}, \{\mathbf{K_{AB}}, A\}_{\mathbf{BS}} \}_{\mathbf{AS}}$

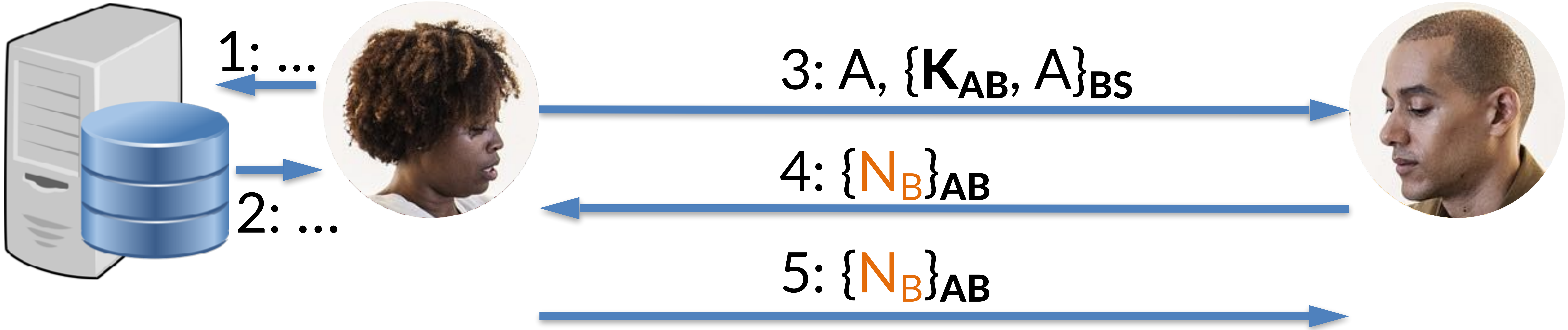
$1: A, B, N_A$

Two Problems

- Bob doesn't know key is fresh
- Alice doesn't know if Bob received the key

Needham–Schroeder: take 5

Improvement: Have Alice and Bob immediately use their newly-received keys to make sure that they both have them and that they agree upon their freshness

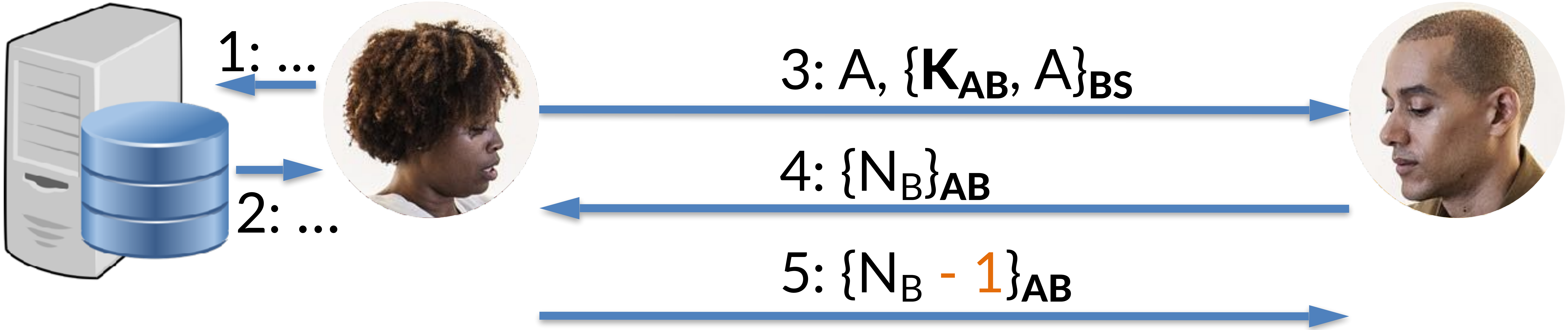


Problem?

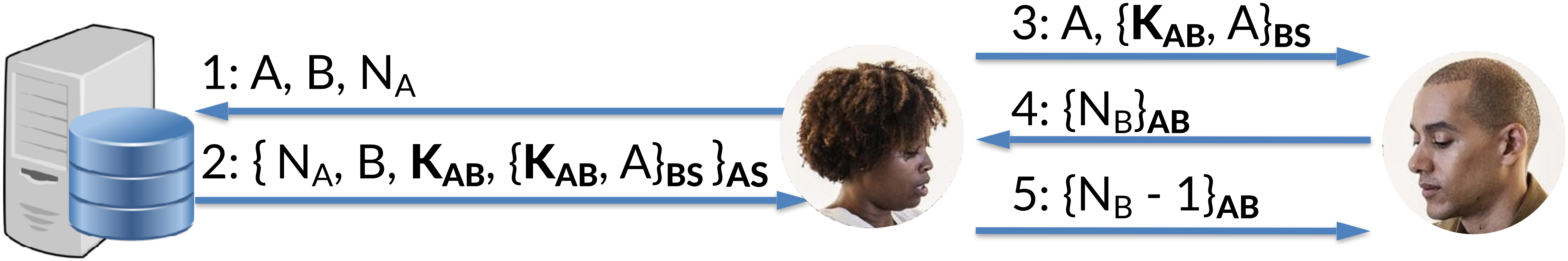
- Anybody can produce message 5, since it equals message 4
- We need Alice to do something that *depends* on the nonce but doesn't *equal* it

Needham–Schroeder: take 6

Key idea: Alice sends a simple *function* of the nonce N_B



Full Needham–Schroeder protocol (1978)



Whew, we finally reproduced the full protocol!

It has no remaining problems… right?

Denning and Sacco (1981)

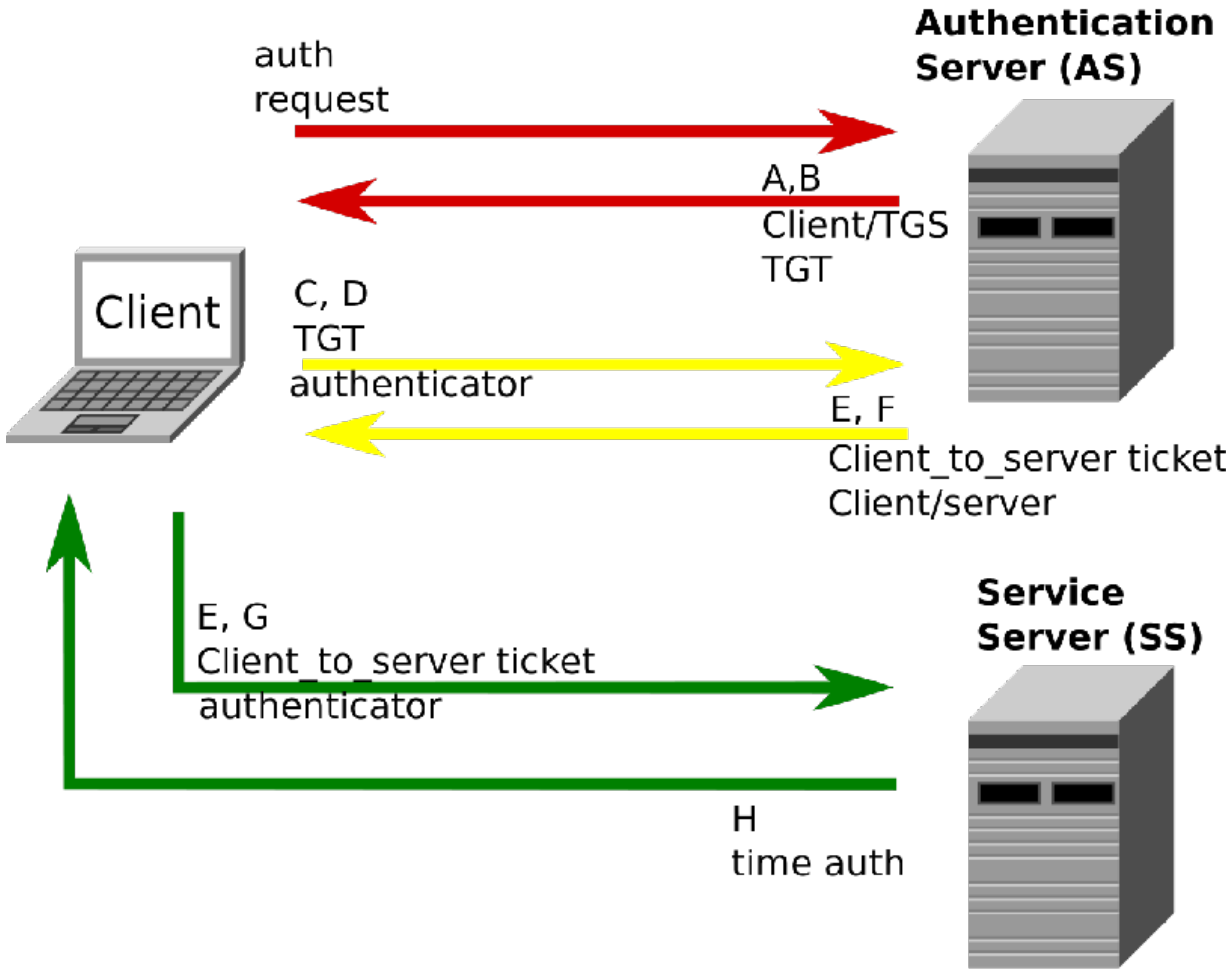
- Problem: Bob's involvement begins in step 4; he has no idea if the first three steps of the protocol occurred recently before then

- If Mallory has compromised key K_{AB} when it was used in the past, she can replay message #3 to start a "new" Needham-Schoeder instance with Bob that he will think is fresh when it isn't

Denning and Sacco's fix

Key idea: Get Bob involved early in the protocol so he also knows it is fresh

Needham-Schroeder → Kerberos

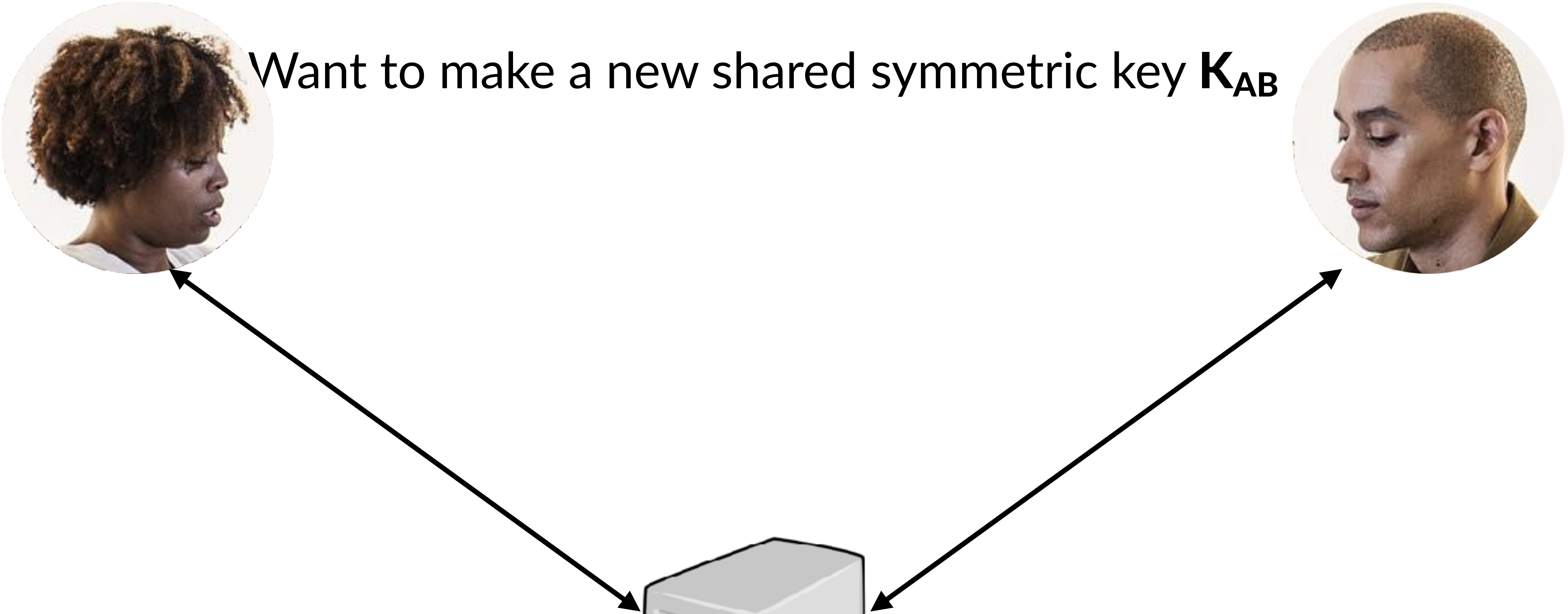


Requirements

1. Users only enter passwords once, at the beginning of each session
2. The network itself is untrusted: passwords and authentication tokens need protection in transit
3. A service must be able to prove that the person *using* a ticket == the person to whom it was *issued*
4. Clients must authenticate services before sending sensitive info to them (*mutual authentication*)

Next time: Public key infrastructure

Lower trust in the server, at the expense of using more expensive cryptography



Name	Unique key
Alice	
Bob	