

Course Announcements

- Homework 8 has been posted, due Wednesday 4/1
- Lecture + recitation section videos posted on Piazza under “Resources”
- Reminder: if you want to ask a question during the virtual lecture, type it in the chat window

Lecture 16: Signal's key ratcheting

1. Digital certificates & the PKI
2. Key evolution
3. Signal's double ratchet
4. Analyzing the Signal protocol

Used in a messaging system near you!



Signal



WhatsApp



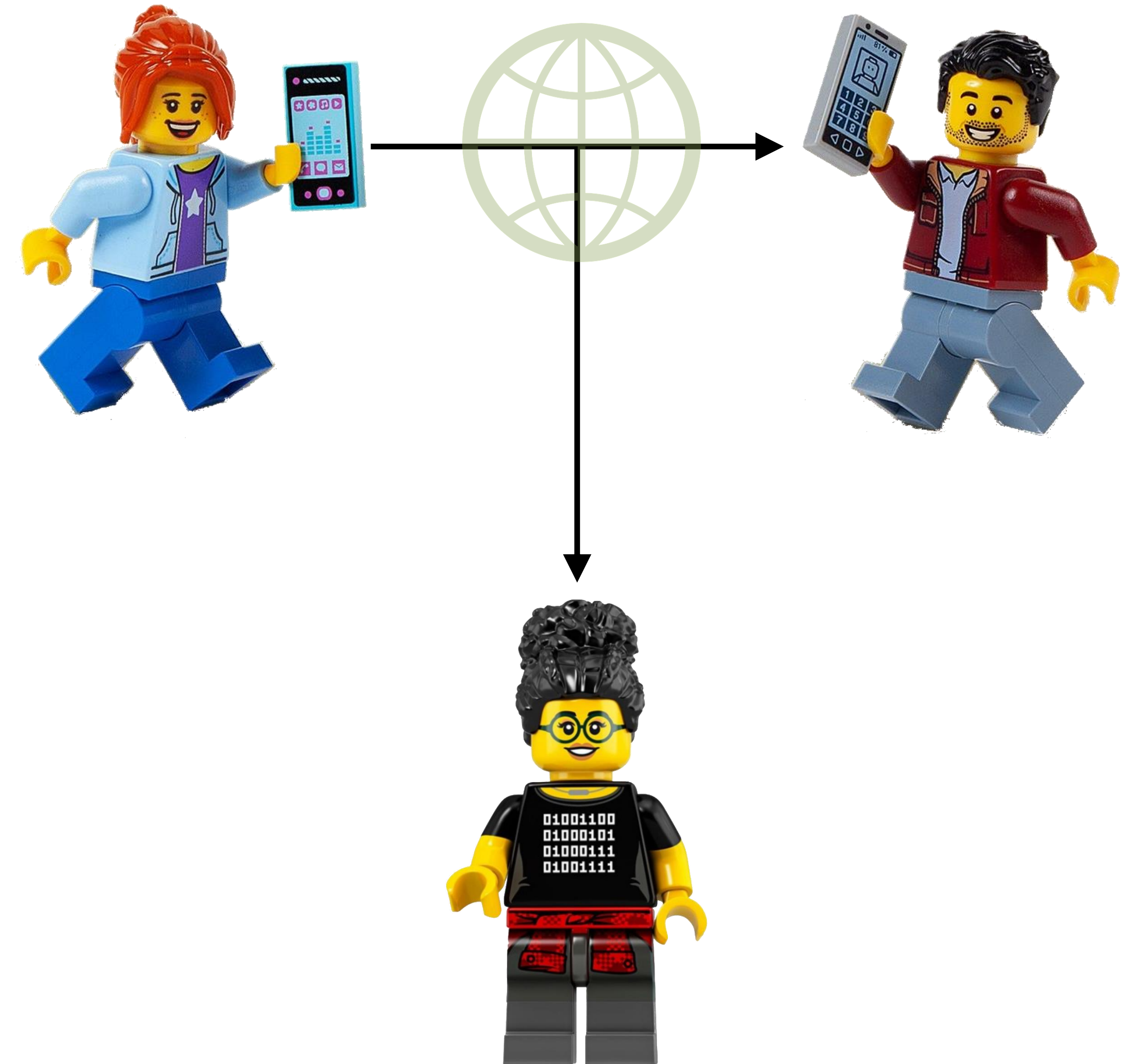
Facebook Messenger



Skype

Recall: end-to-end (e2e) data protection

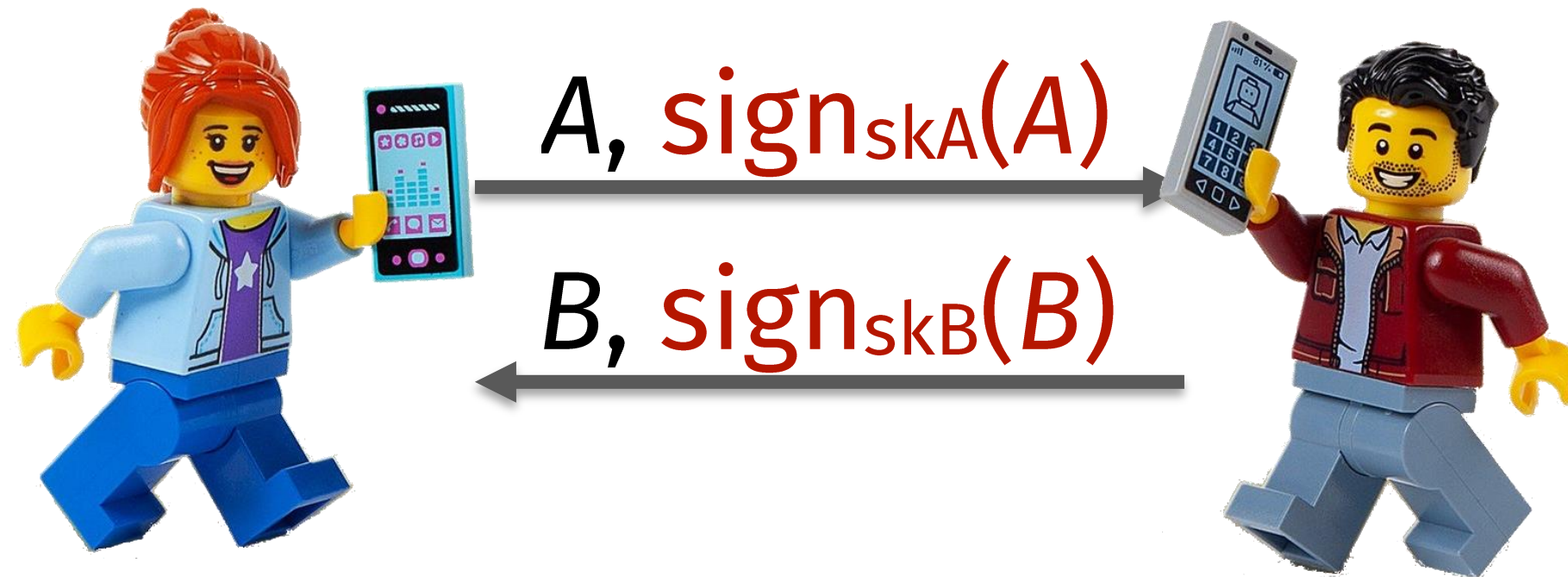
- Alice and Bob want to have a private digital conversation
- They would like to use AuthEnc
 - Provides privacy + authenticity vs. Mallory with full network control
 - Provides partial sender deniability even if Mallory coerces Bob
- Remaining issues
 - Alice and Bob don't yet have a shared (*symmetric*) key
 - Need forward + backward secrecy



Recall: Authenticated key agreement for e2e protection

Choose a randomly
Compute $A = g^a$

Choose b randomly
Compute $B = g^b$



1. Alice and Bob sign their messages during Diffie-Hellman key agreement
2. Alice and Bob verify signature of each other's messages —
3. Use shared key $A^b = B^a$ for (deniable) symmetric authenticated encryption

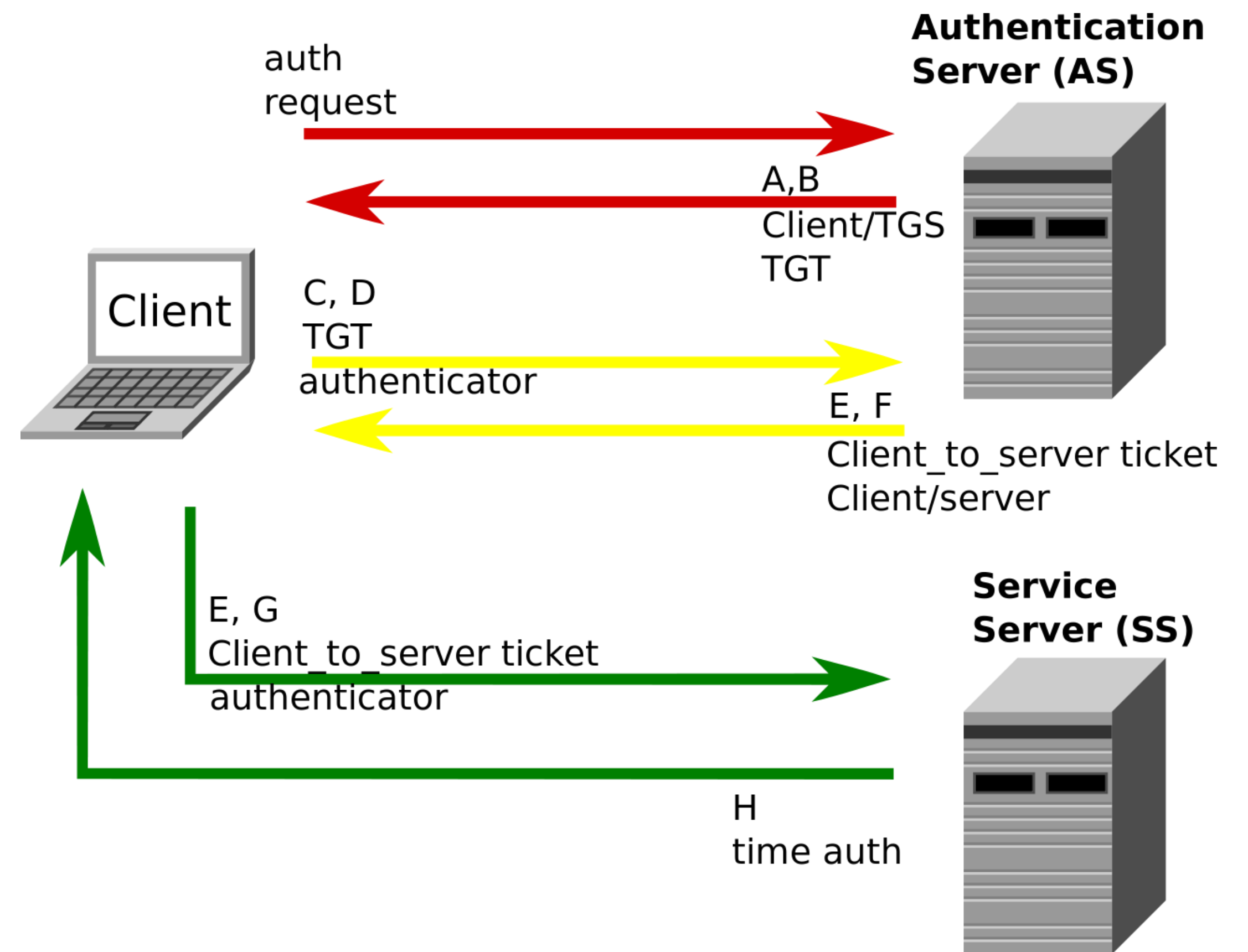
Question: how do Alice and Bob learn each other's public keys?

Question: how can we get forward + backward secrecy?

1. Digital Certificates & the Public Key Infrastructure (PKI)

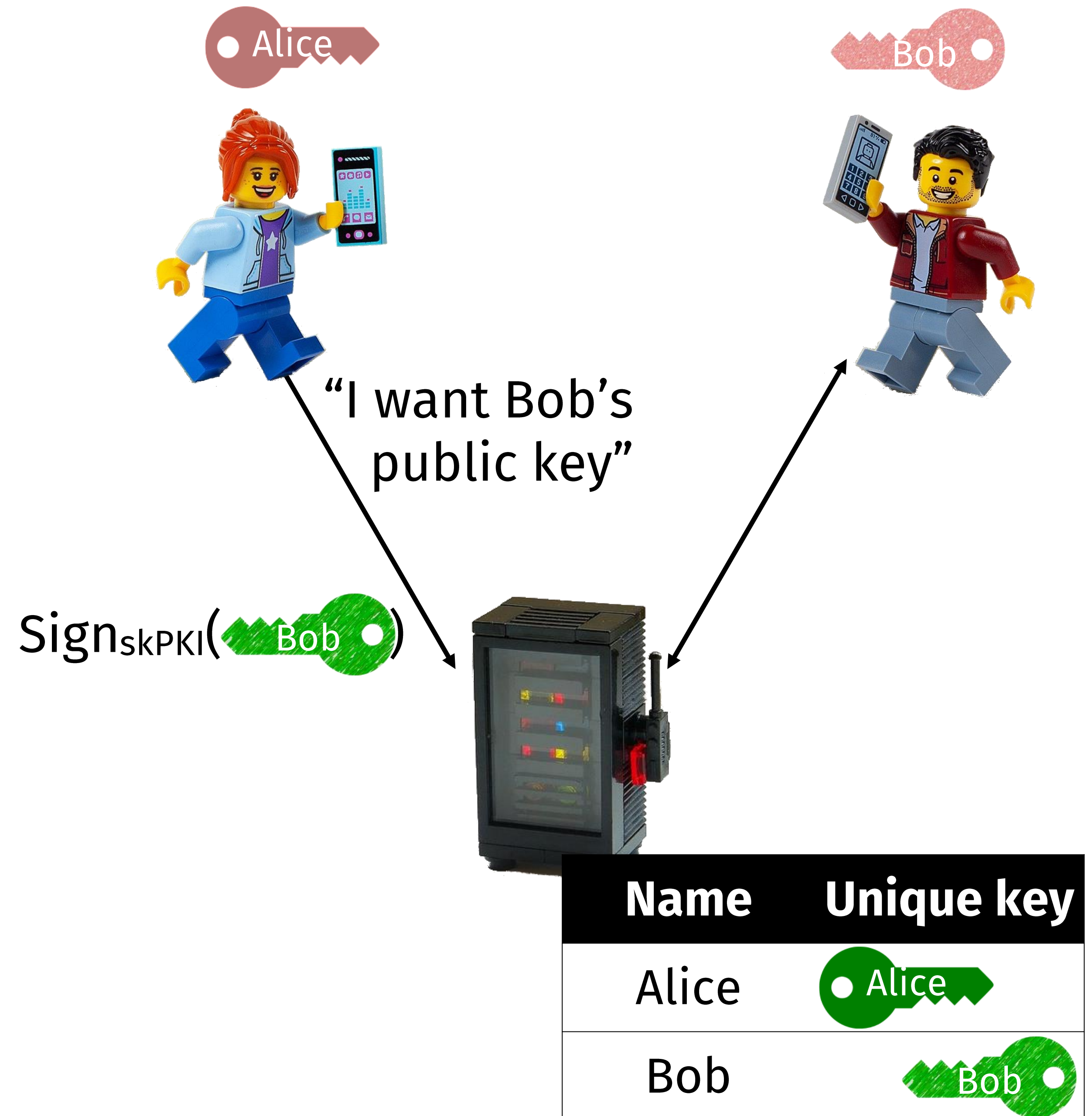
One option: ask a common friend to connect you

- Suppose that
 - Alice + Bob both trust server S
 - Alice + Bob have shared symmetric keys K_{AS} , K_{BS} with the server
- Then, server can create key K_{AB} and send it to them
 - Needham-Schoeder (1978)
 - Kerberos (late 1980s)
- Mostly used within a single enterprise, since server sees all



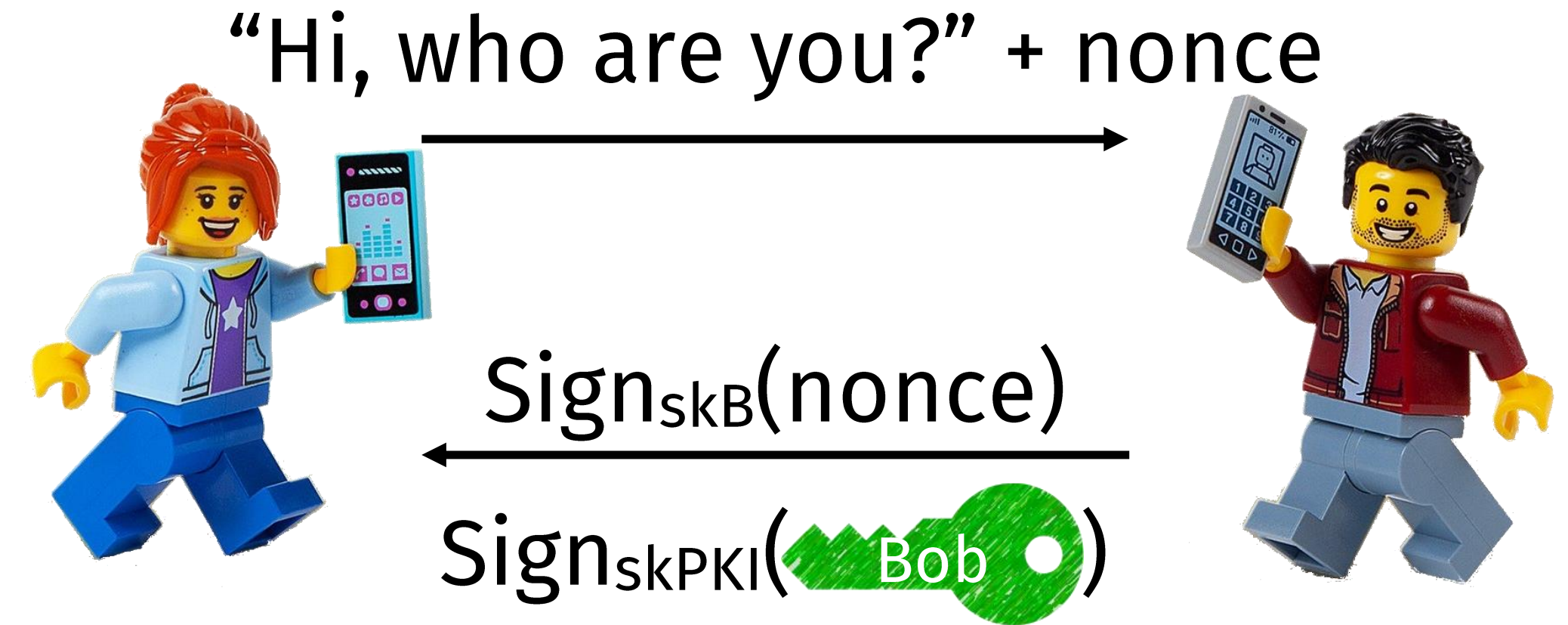
Public key infrastructure



- A *certificate authority* stores all public keys (like a phone book)
 - Server does *not* learn private keys
- Anyone can query the authority to learn someone else's key
- CA signs response *certificates* so they can be verified as legit
- Alice knows the CA's public key because it is included in her OS



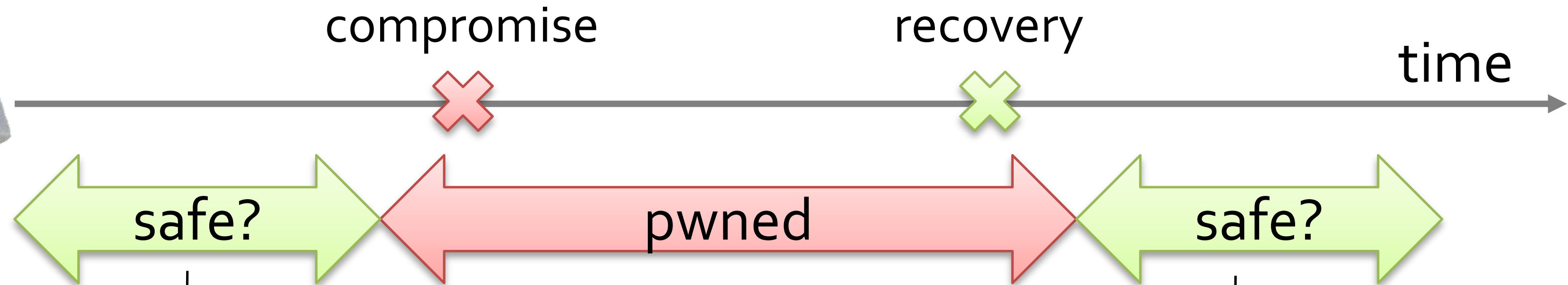
PKI improved

- Alice talks with Bob, not CA
- Bob includes a certificate that the signing key belongs to him
- (Shown: simplified version of the TLS handshake)



Name	Unique key
Alice	
Bob	

What if Bob's secret signing key is compromised?

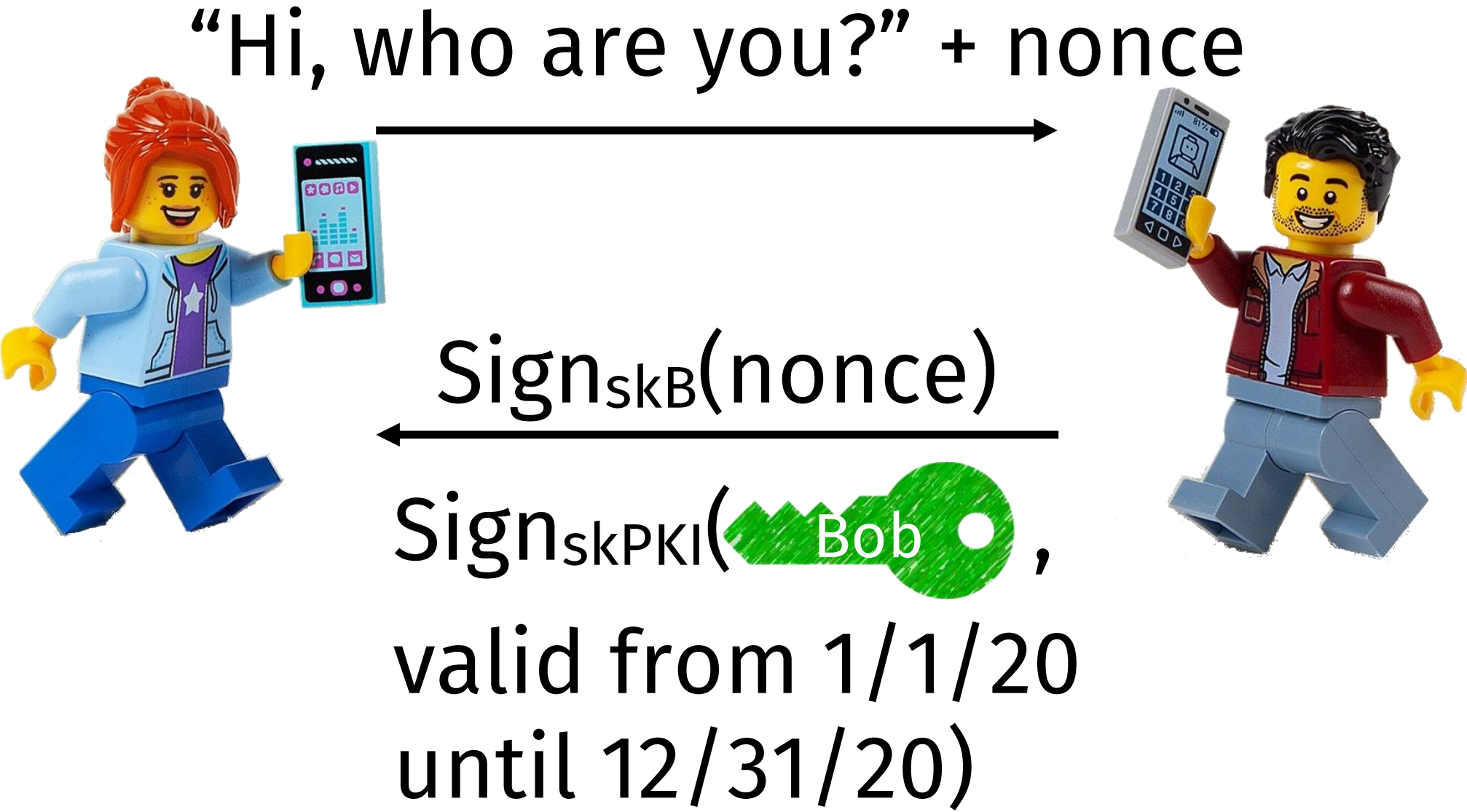


Forward (pre-compromise) secrecy
Yes! Unless Mallory has a time machine, signatures Alice verified before a breach must be valid.

Backward (post-recovery) secrecy
No. If Mallory has Bob's secret key, she can sign messages and Alice will believe they are from Bob.

Backward security technique #1: Cert expiration

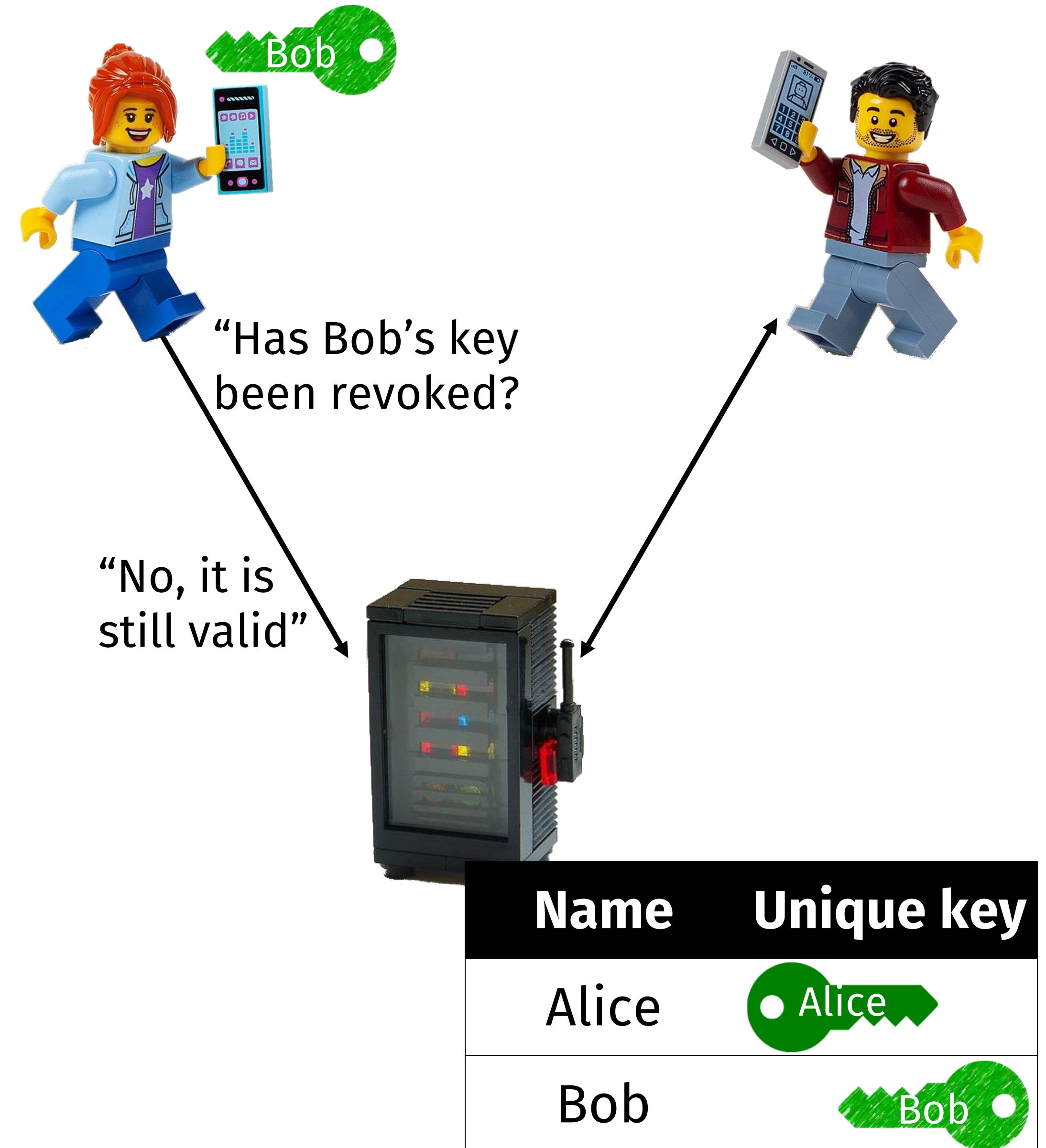
- Certificate states that Alice should only trusts Bob's key for a limited time
- Afterward, Bob must register a new public key with the CA
- (Cert expiration also helps to deal with Moore's law: keys become bigger over time)



Name	Unique key
Alice	
Bob	

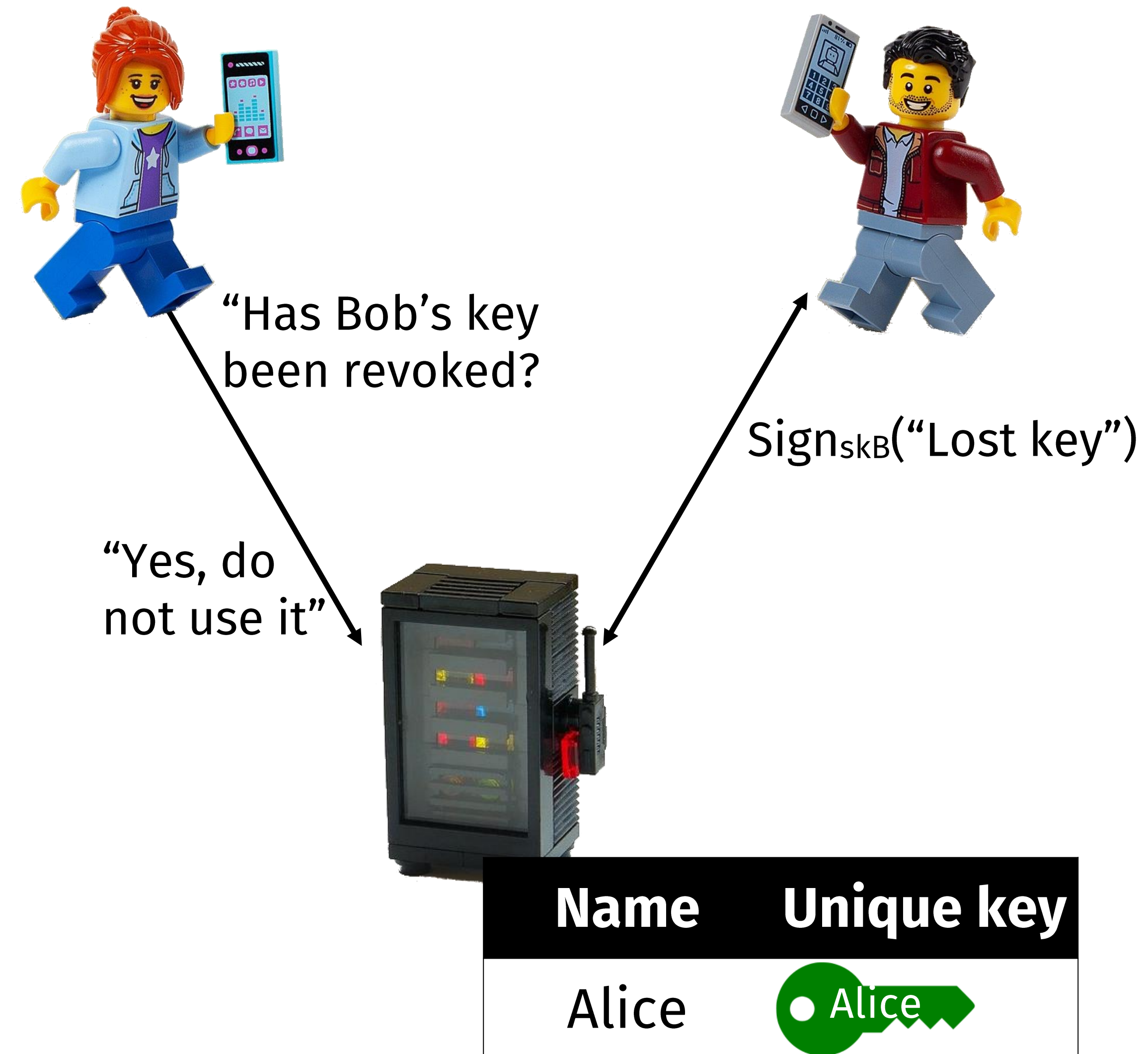
Backward security technique #2: Key revocation

- CA binds public key to a name
- If you lose control of your public key, you should tell the CA to break this binding
- Every CA maintains a certificate revocation list that anyone can query



Backward security technique #2: Key revocation

- CA binds public key to a name
- If you lose control of your public key, you should tell the CA to break this binding
- Every CA maintains a certificate revocation list that anyone can query



2. Key Evolution

Symmetric key evolution

Question: Once Alice and Bob negotiate a shared symmetric key K_{AB} for authenticated encryption, must they re-execute another (expensive) key negotiation protocol each time they want to update the key?

Basically, seek Authenticated Encryption with a key update mechanism

- KeyGen: randomly choose key K of length λ , e.g. uniform in $\{0,1\}^\lambda$
- AuthEnc $_K$ (private P , authenticated A , nonce N) \rightarrow ciphertext C
- AuthDec $_K$ (C , A , N) $\rightarrow P$ or “error”
- KeyUpdate(K) $\rightarrow K'$ where Alice + Bob agree to use K' from now onward, and cannot compute K from K'

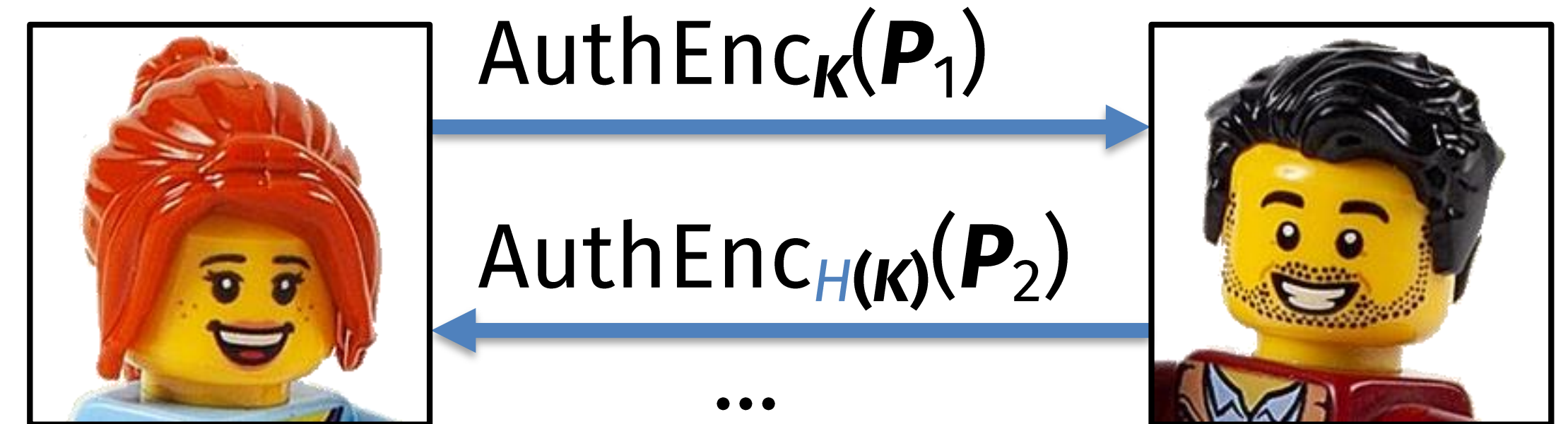
Symmetric key evolution via hash functions

Idea: Once we have a single shared key K_{AB} , expand using a chain of hash functions

$$K_{AB} \rightarrow H(K_{AB}) \rightarrow H(H(K_{AB})) \rightarrow H(H(H(K_{AB}))) \rightarrow \dots$$

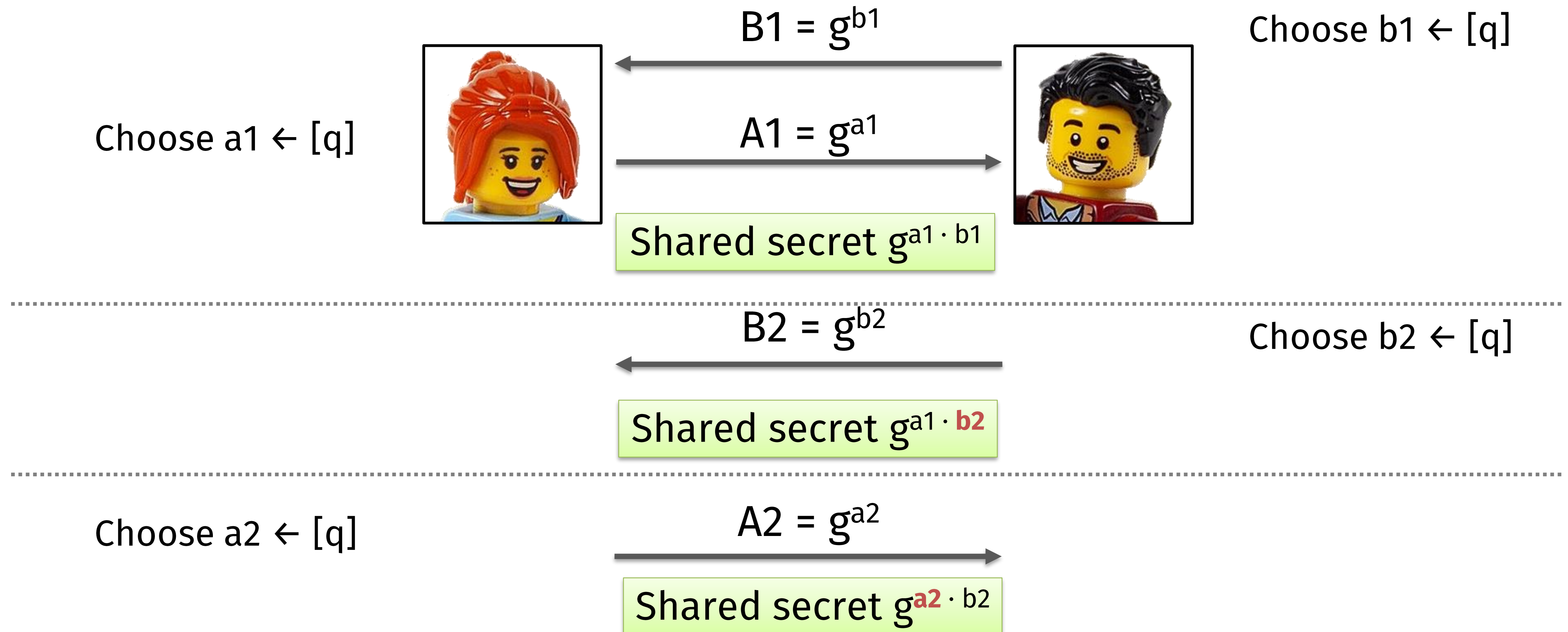
Algorithm:

- Alice + Bob agree on key K_{AB} to use for auth enc
- After some time has passed, they can evolve their key by updating $K \leftarrow H(K)$
 - Here, “time” can denote actual wall-clock time or a message counter
 - Alice + Bob must stay in sync, or else the chain breaks & they must redo key agreement
- Crucially, they ensure that old values of K are deleted from their system! Evolution relies on the fact that Mallory cannot steal something that isn't around to be stolen



Public key evolution

2 rounds of Diffie-Hellman create a shared secret, and 1 round can update it!



3. Signal's Double Ratchet

Double ratchet rules

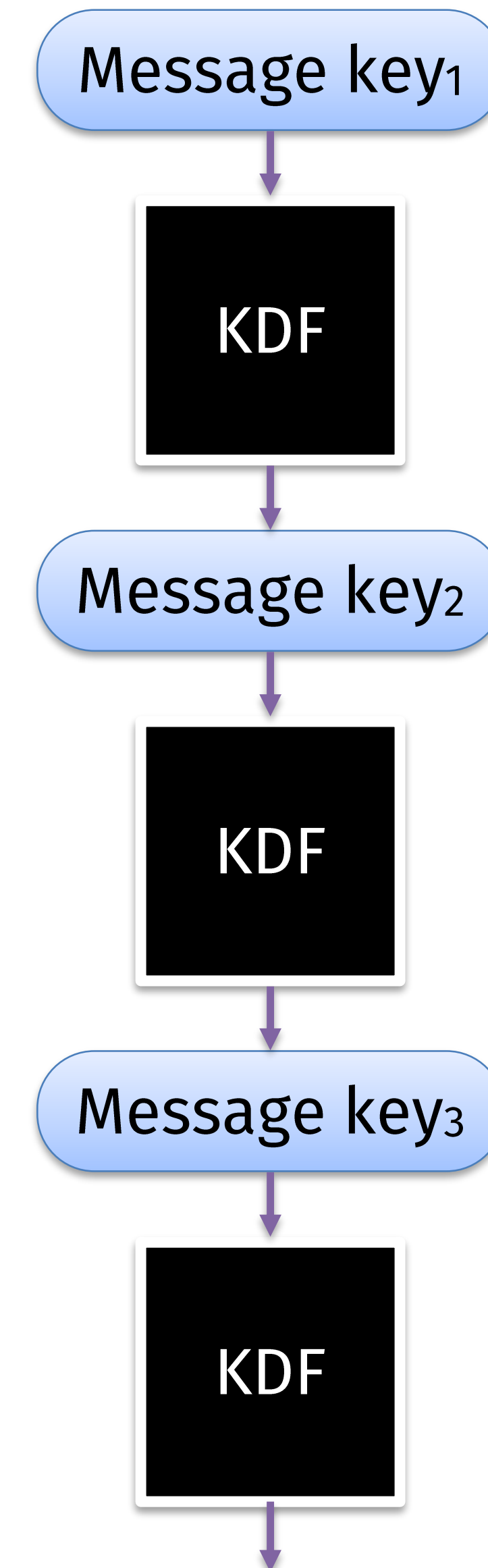
Alice and Bob maintain two sets of keys: one for Alice-to-Bob messages, and another for Bob-to-Alice messages

1. When a message is sent or received, a *symmetric ratchet* KDF step is applied to the sending or receiving chain to derive a new message key
2. When alternating the direction of communication, a *public ratchet* step updates the chain keys that are used in the symmetric ratchet

Signal messaging protocol (simplified)

1. Key evolution

- Each key encrypts 1 msg, then evolved + deleted
- Keys are forward secure but not backward secure



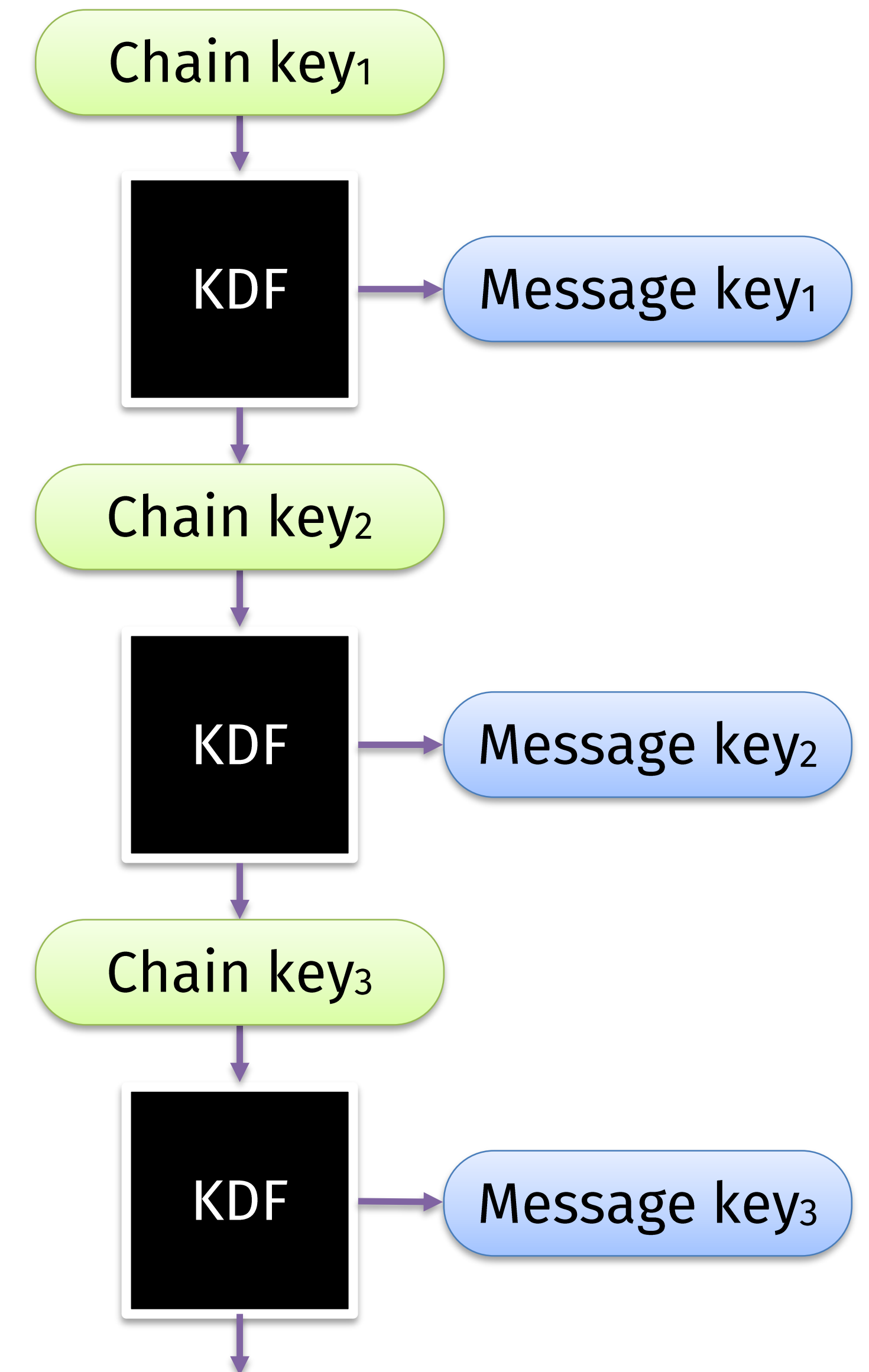
Signal messaging protocol (simplified)

1. Key evolution

- Each key encrypts 1 msg, then evolved + deleted
- Keys are forward secure but not backward secure

2. Key derivation

- Message keys now forward + backward secure
- “Message keys aren't used to derive other keys... useful for handling lost/out-of-order messages”



Signal messaging protocol (simplified)

1. Key evolution

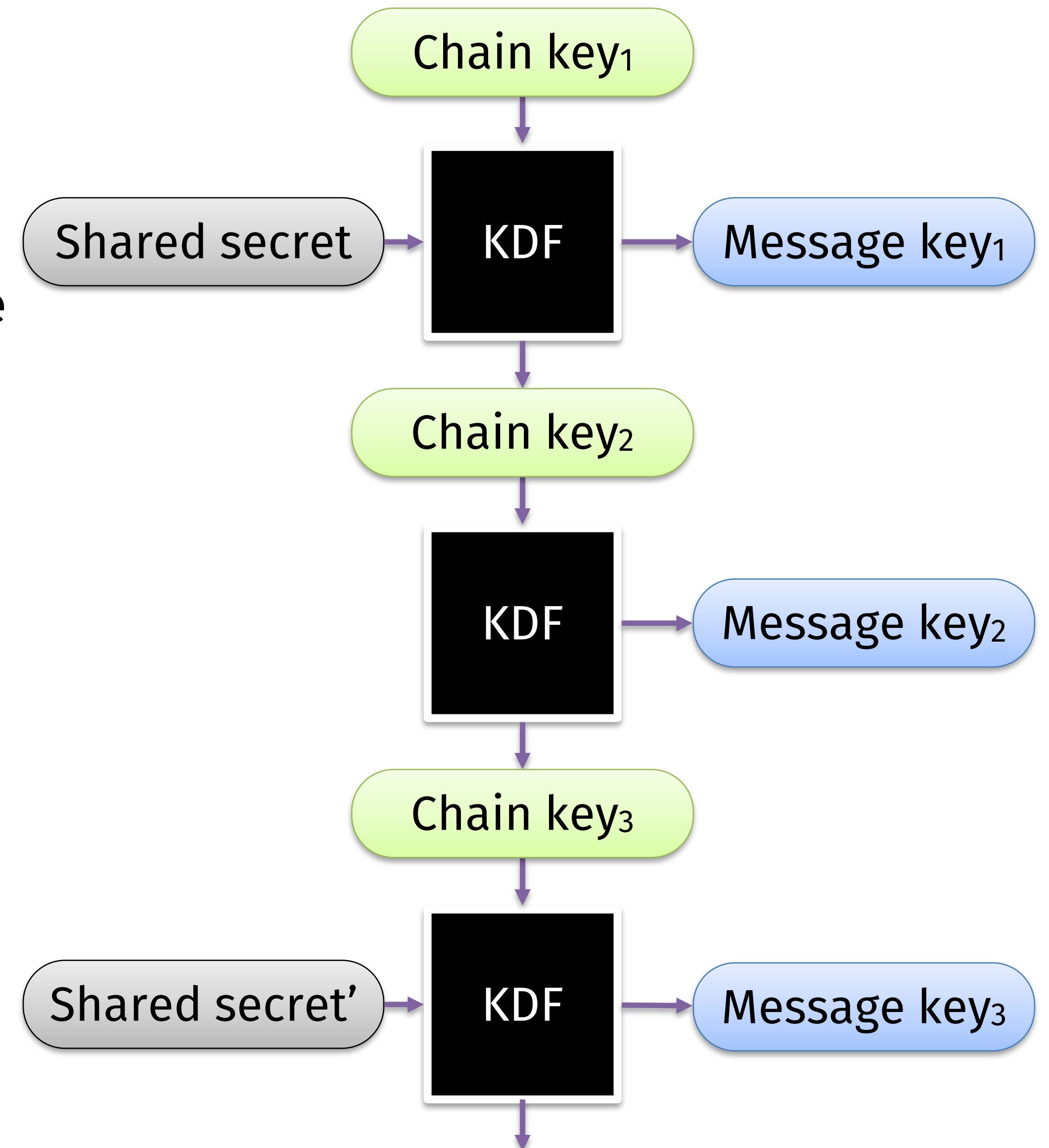
- Each key encrypts 1 msg, then evolved + deleted
- Keys are forward secure but not backward secure

2. Key derivation

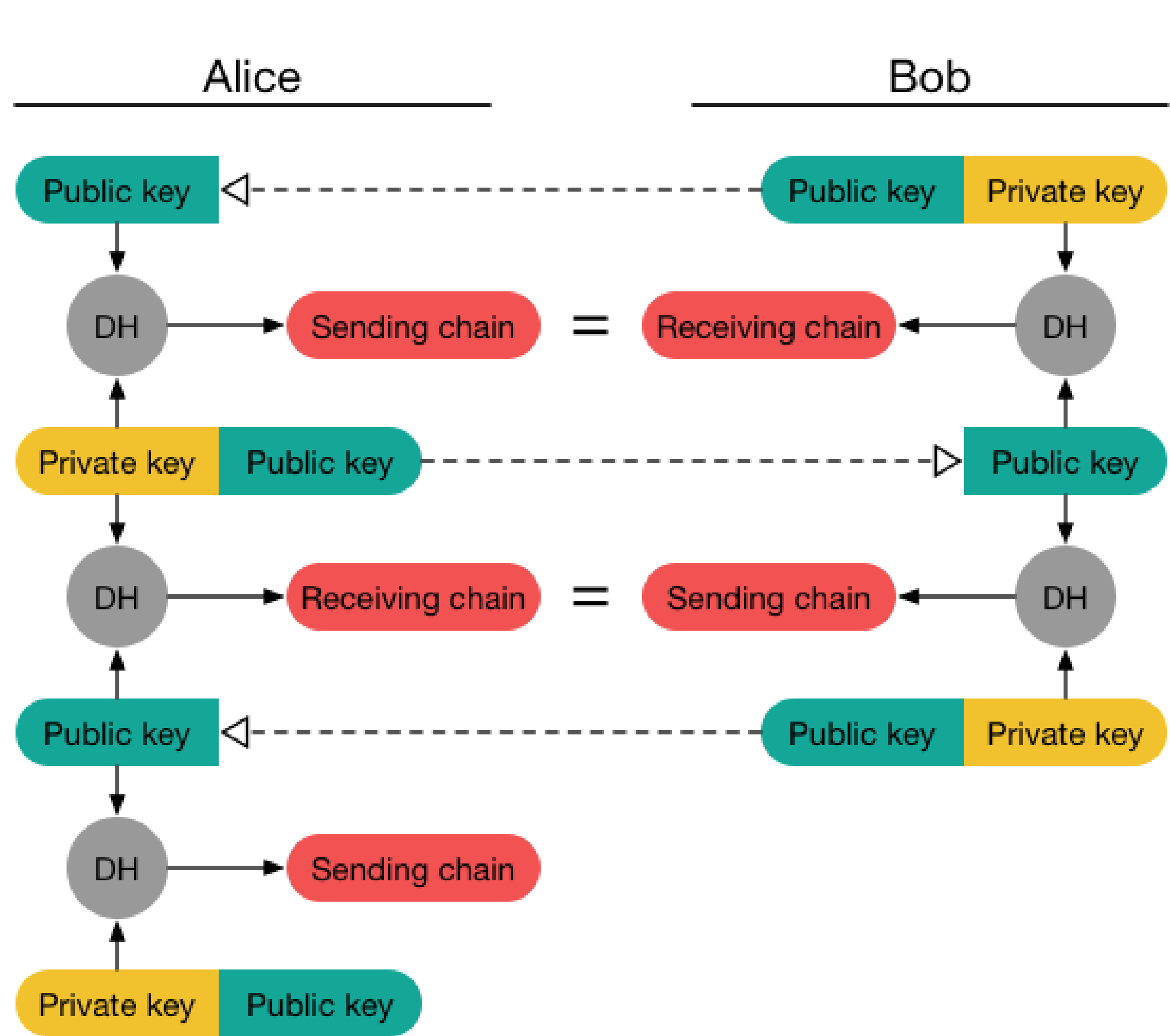
- Message keys now forward + backward secure
- “Message keys aren't used to derive other keys... useful for handling lost/out-of-order messages”

3. Key ratcheting

- Periodically build new D-H shared secrets
- If adv doesn't know shared secret, then recover from losing chain key (backward secrecy)



Public ratchet seeds symmetric ratchets (one per direction)



How do we start this process?
How does Alice learn Bob's first D-H message if Bob is offline?

4. Analyzing the Signal Protocol

Putting everything together

Deniable

Choose $a \leftarrow [q]$
Compute $A = g^a$

Choose $b \leftarrow [q]$
Compute $B = g^b$

Fwd/back secure

Ephemeral
secret



Output B^a



$K = \text{KDF}(B^a)$



Output A^b



$K = \text{KDF}(A^b)$

Evolve
public key

AuthEnc is
deniable



$\text{AuthEnc}_K(P_1)$

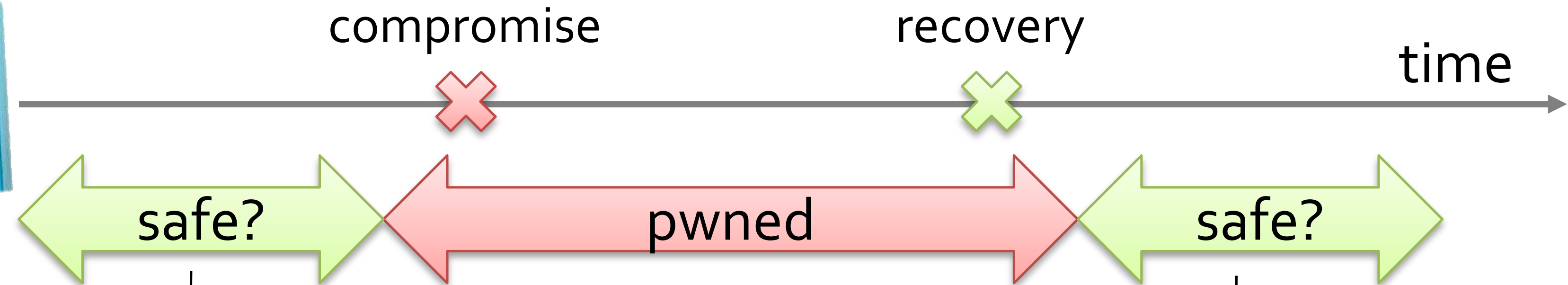
$\text{AuthEnc}_K(P_2)$

...



Evolve
symm key

Why Signal provides forward and backward secrecy



Forward (pre-compromise) secrecy

“The parties derive new keys for every Double Ratchet message so that earlier keys cannot be calculated from later ones.”

Backward (post-recovery) secrecy

“The parties also send Diffie-Hellman public values attached to their messages. The results of Diffie-Hellman calculations are mixed into the derived keys so that later keys cannot be calculated from earlier ones.”

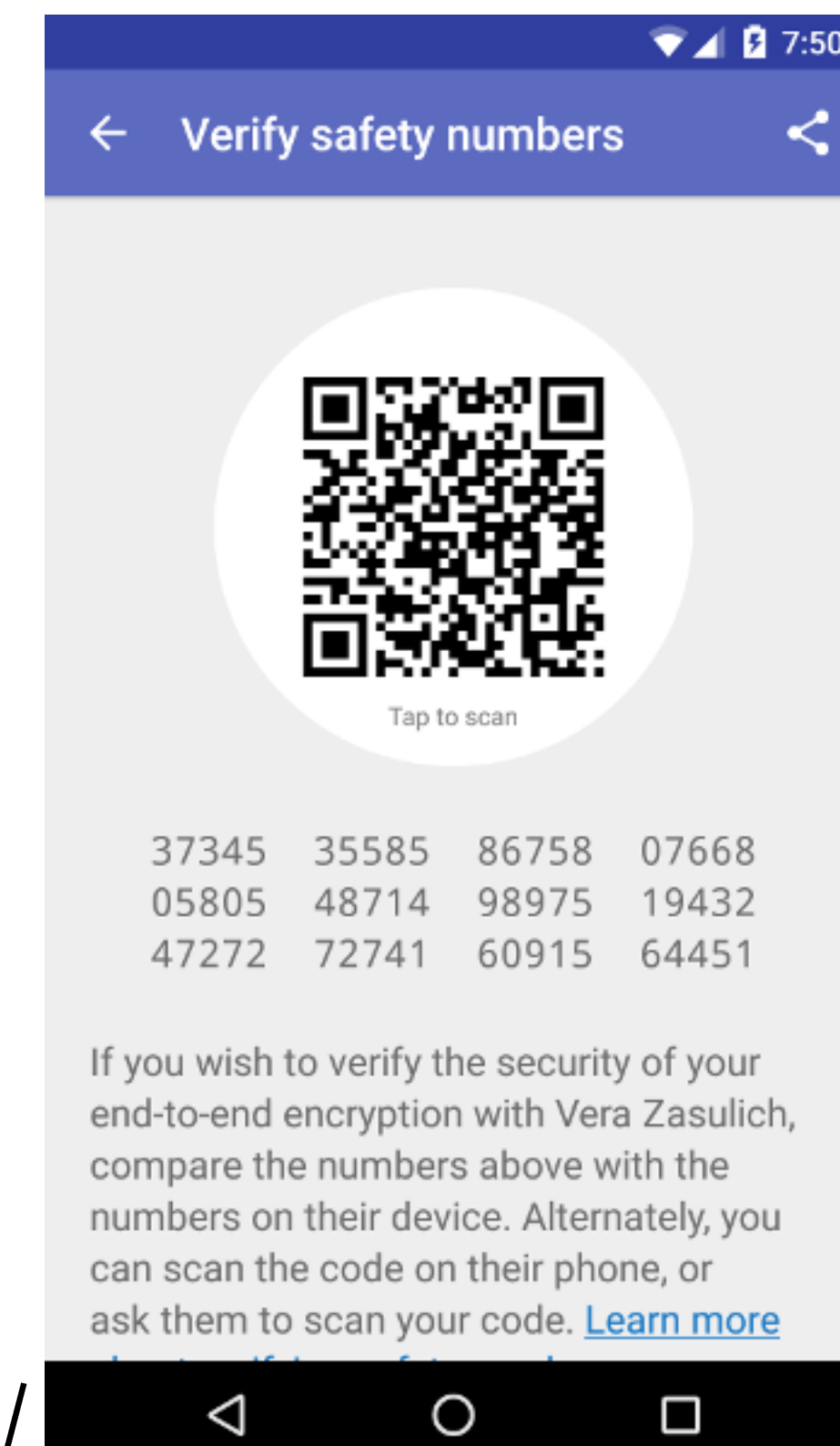
Ephemeral utopia

No long-term keys \Rightarrow great forward secrecy

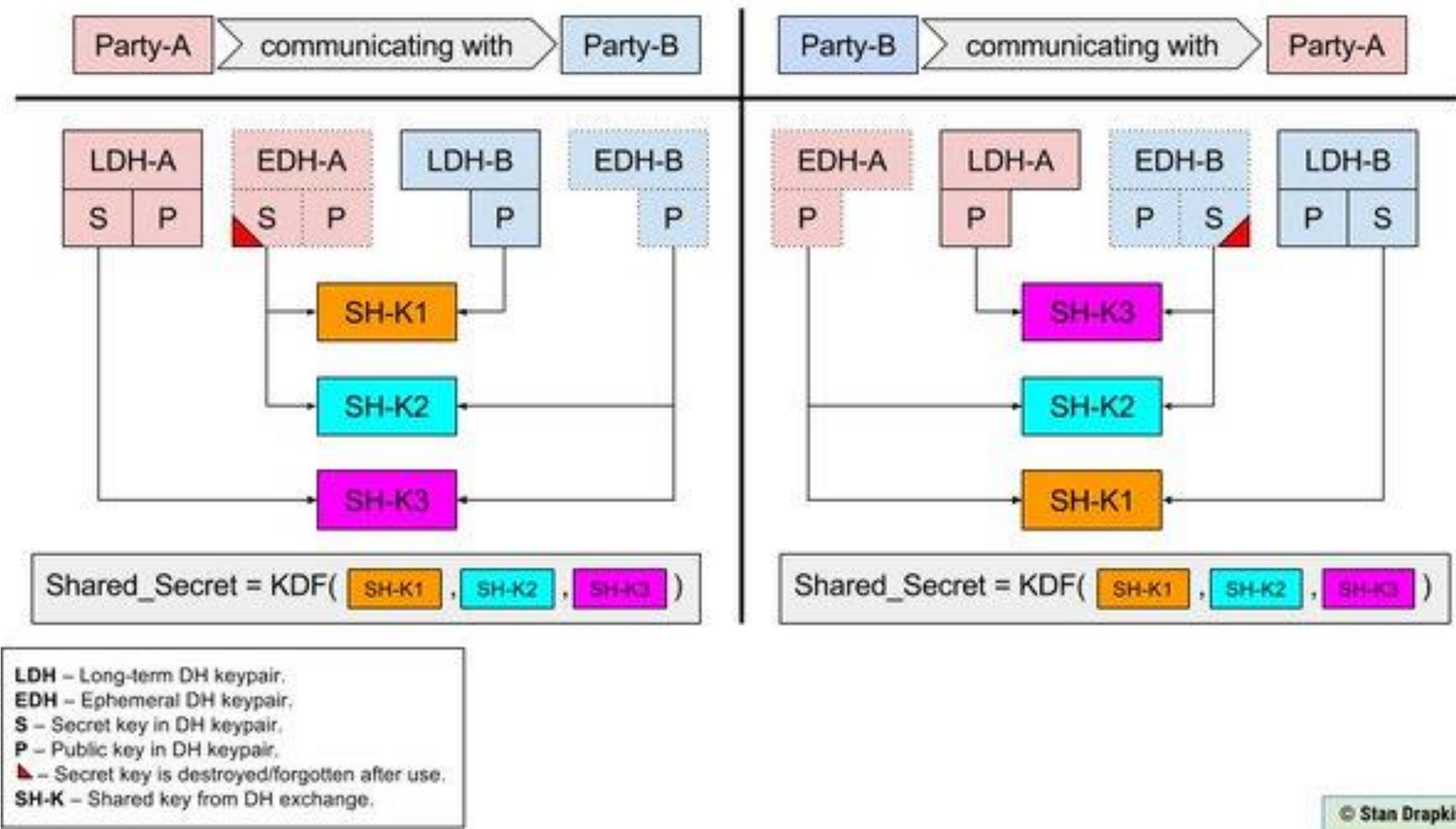
- Message key used to AuthEnc a message is used once and tossed
- Chain key used to construct msg key is refreshed in each public ratchet
- Diffie-Hellman key pairs chosen ephemerally in each public ratchet

Wait... actually, is this a utopia or a dystopia?

- If you don't have *any* long-term state, then who are you?!
- Resolution: Also have a long-term key, Signal maintains a PKI



Solution: a more involved *Triple-DH protocol*



SECURE MESSAGING APPS COMPARISON

BECAUSE PRIVACY MATTERS

App name	Allo	iMessage	Messenger	Signal	Skype	Telegram	Threema	Viber	Whatsapp	Wickr	Wire
TL;DR: Does the app secure my messages and attachments?	No	No	No	Yes	No	No	Yes	No	No	No	Yes

Source: <https://www.securemessagingapps.com>

Thinking About What You Need In A Secure Messenger

BY GENNIE GEBHART | MARCH 28, 2018



Source: <https://www.eff.org/deeplinks/2018/03/thinking-about-what-you-need-secure-messenger>

**Next week:
protecting data in use**

Course roadmap

**Elegant
protocols**

**Utilitarian
tools**

