

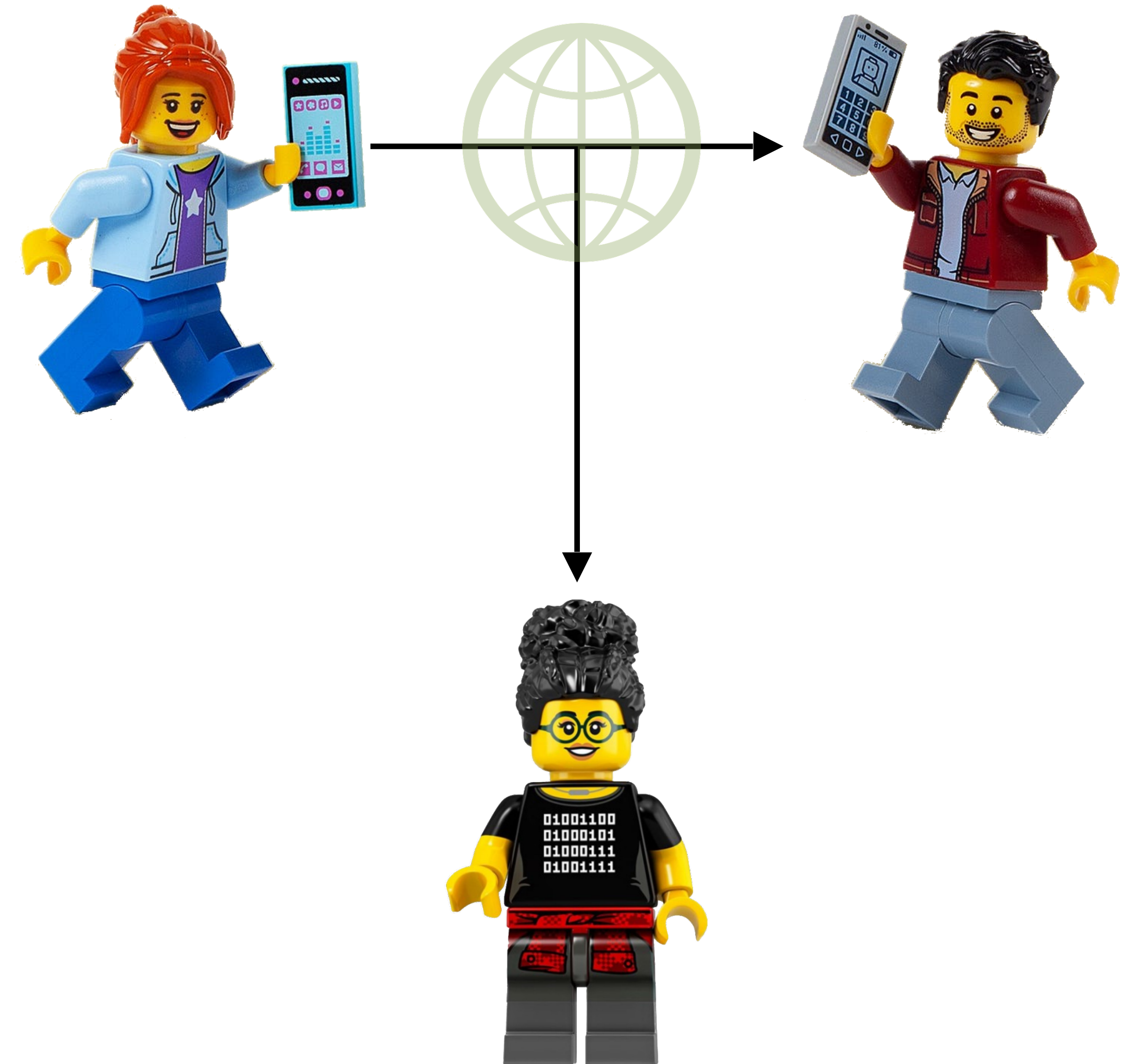
Lecture 18: Signal's double ratchet (part 2)

Announcements

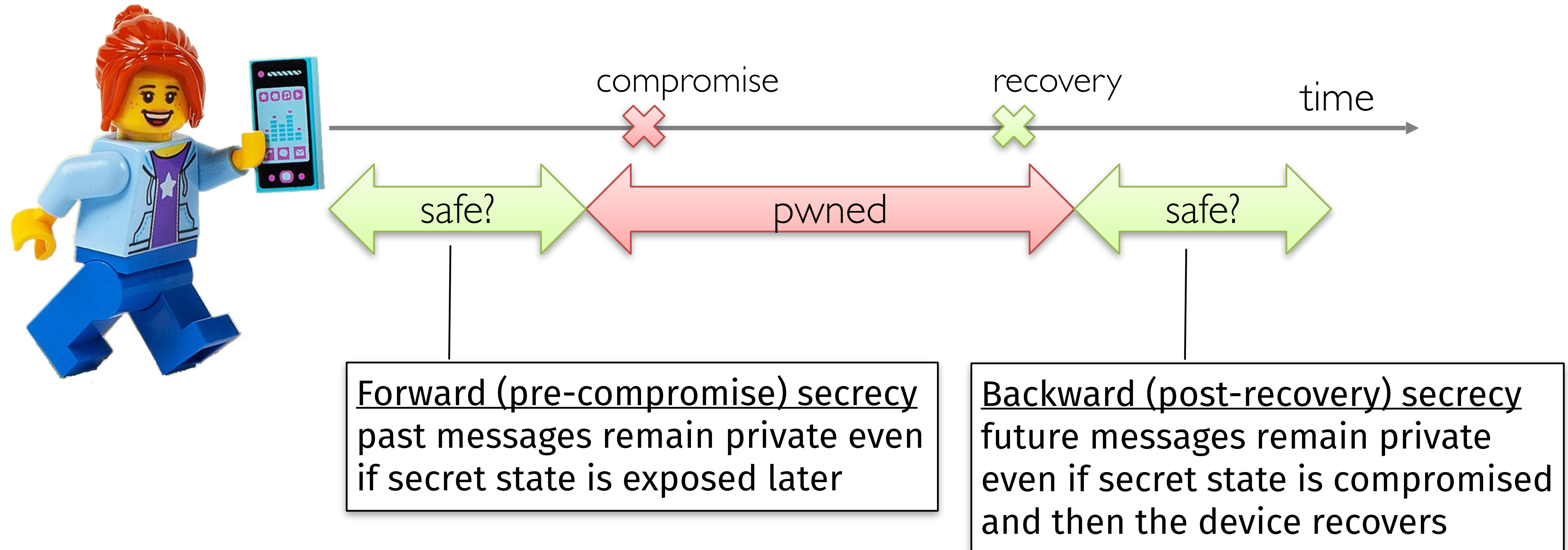
- Weekly reading: Boneh-Shoup, parts of chapter 9 on authenticated encryption
- HW 7 is due Friday 4/4
- Test 3 is next Monday 4/7 in IEC B10 during discussion lab (9:05-9:55 / 10:10-11)

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
 - Need to change keys frequently
 - Need forward + backward secrecy
 - Need deniability



Recall: Forward and backward secrecy



18.1 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 $\{0,1\}^\lambda$
- $\text{AuthEnc}_K(\text{private } P, \text{authenticated } A, \text{nonce } N) \rightarrow \text{ciphertext } C$
- $\text{AuthDec}_K(C, A, N) \rightarrow P$ or “error”
- $\text{KeyUpdate}(K) \rightarrow K'$ where Alice + Bob agree to use K' from now onward, and cannot compute K from K'

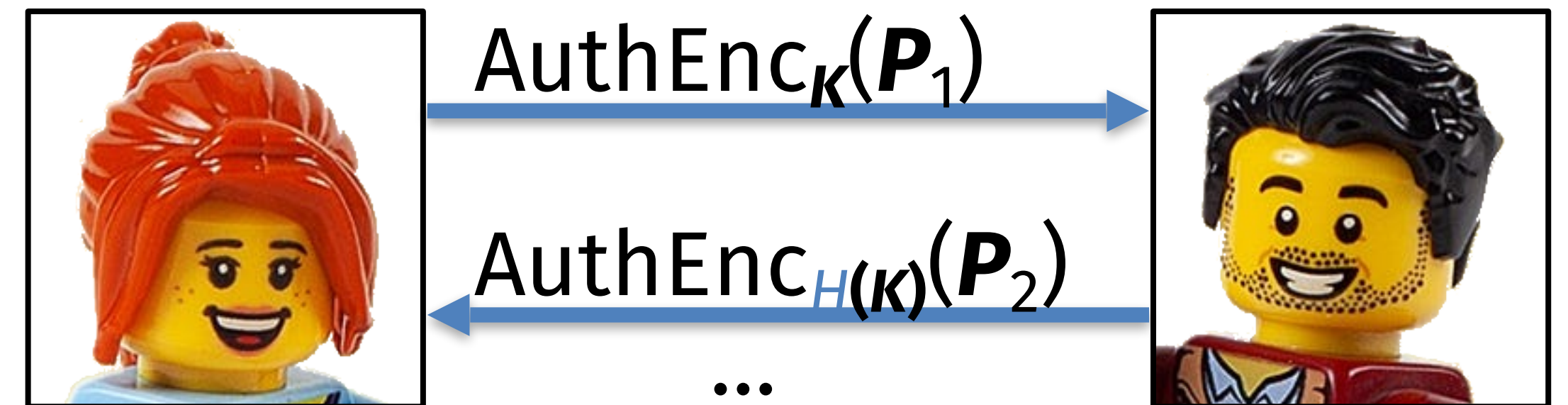
Symmetric key evolution via hash functions

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

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

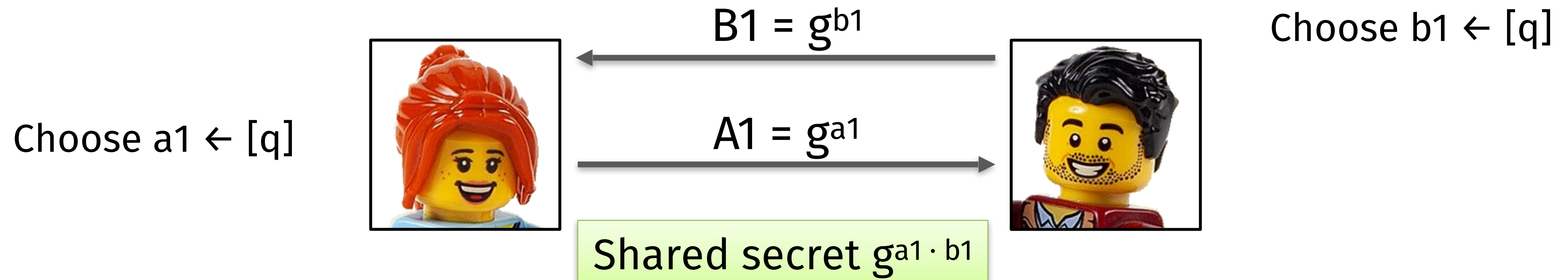
Algorithm:

- 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!



18.2 Signal's Double Ratchet

Double ratchet rules

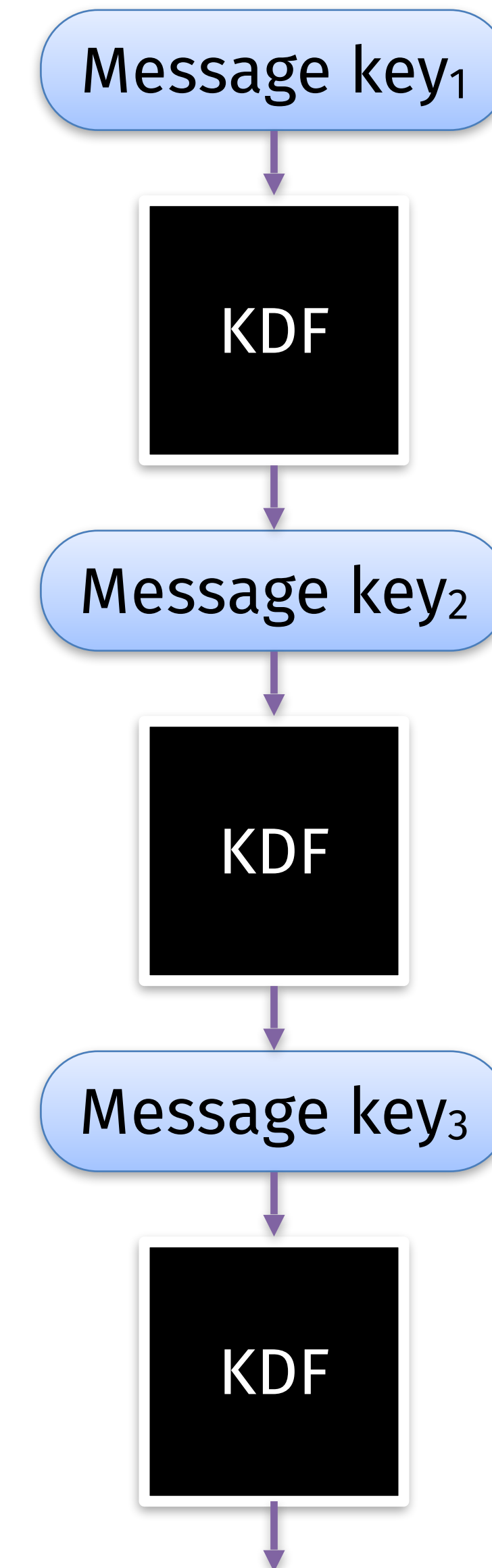
Alice and Bob maintain two sets of keys: one for Alice-to-Bob messages, and another for Bob-to-Alice messages

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



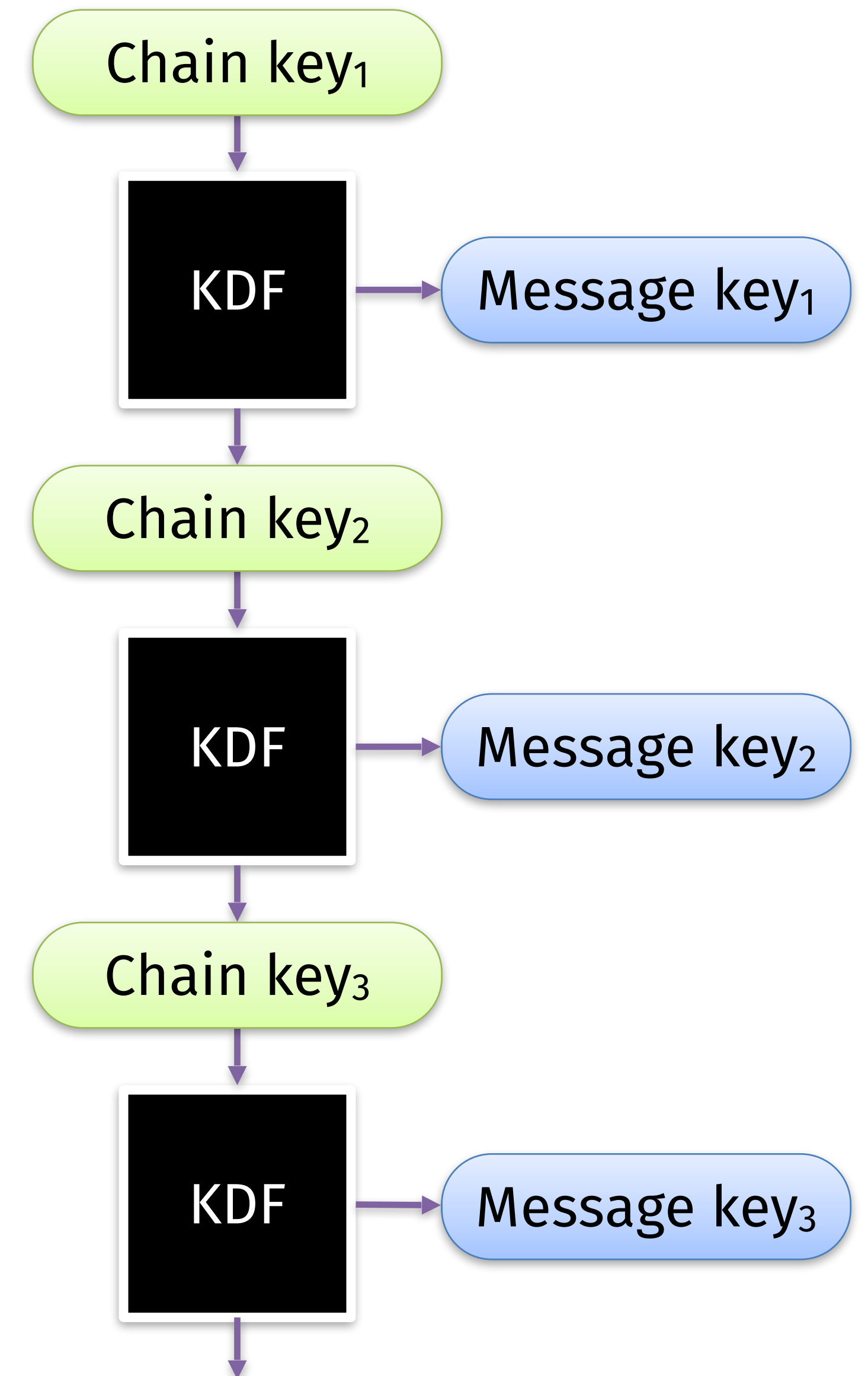
Signal messaging protocol (simplified)

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

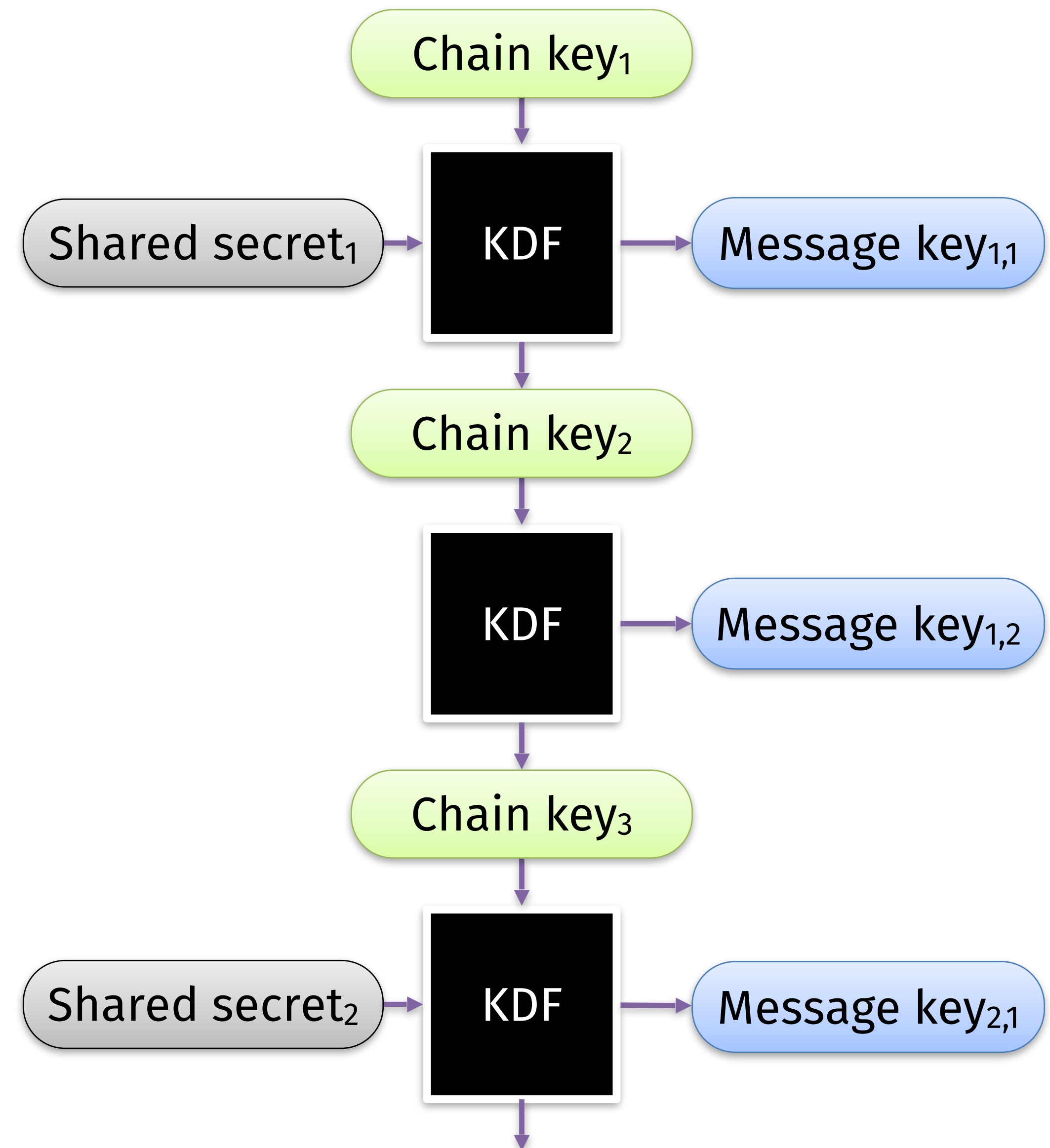
- Each key encrypts 1 msg, then evolved + deleted
- Keys are forward secure but not backward secure

2. Key derivation

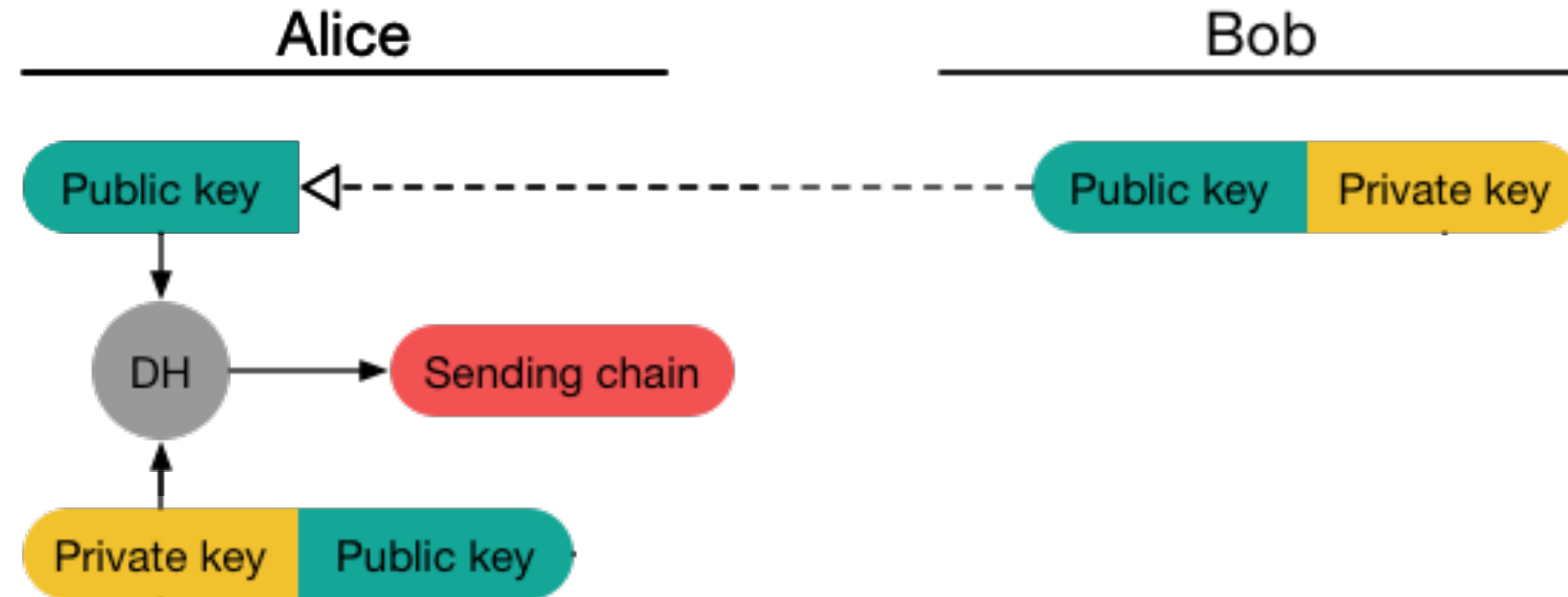
- Message keys now forward + backward secure
- “Message keys aren't used to derive other keys... useful for handling lost/out-of-order messages”

3. Key ratcheting

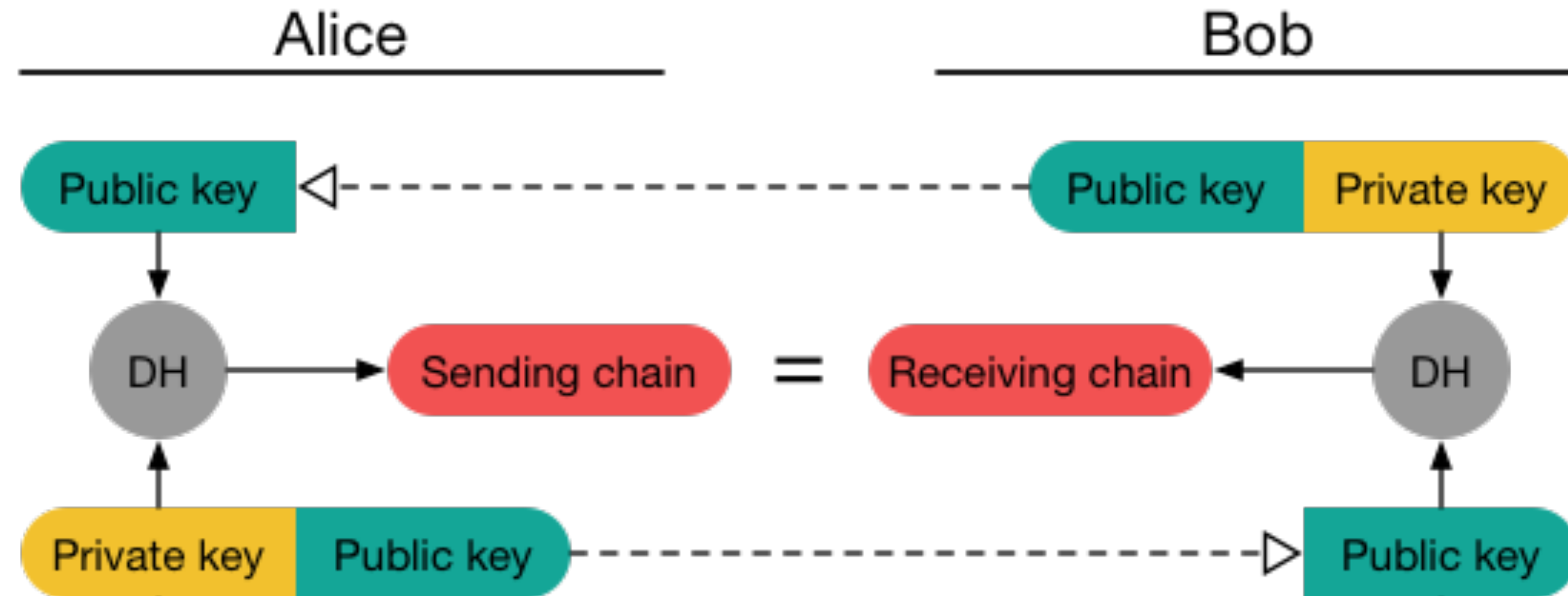
- Periodically build new D-H shared secrets
- If adv doesn't know shared secret, then recover from losing chain key (backward secrecy)



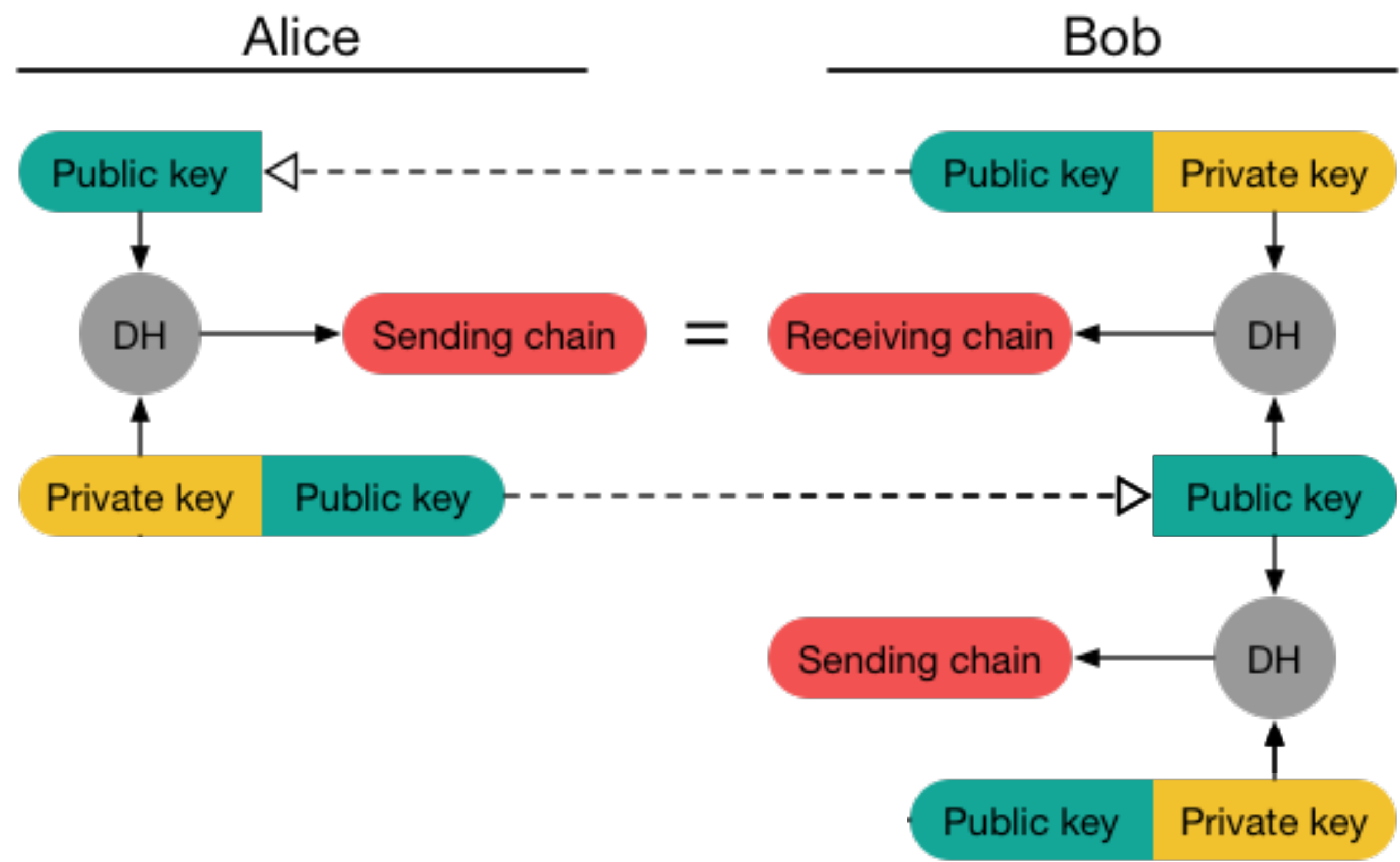
Public ratchet seeds symmetric ratchets (one per direction)



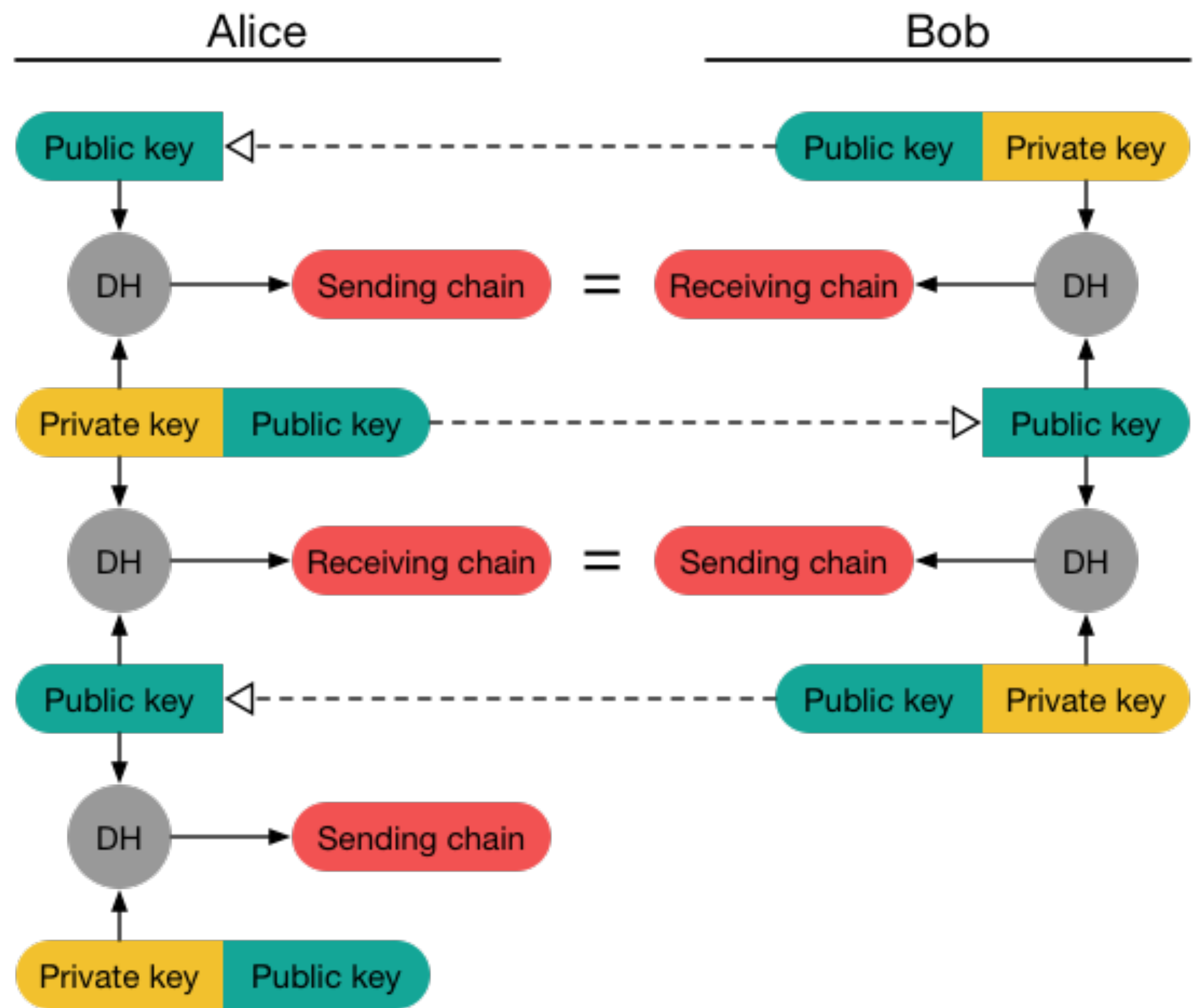
Public ratchet seeds symmetric ratchets (one per direction)



Public ratchet seeds symmetric ratchets (one per direction)

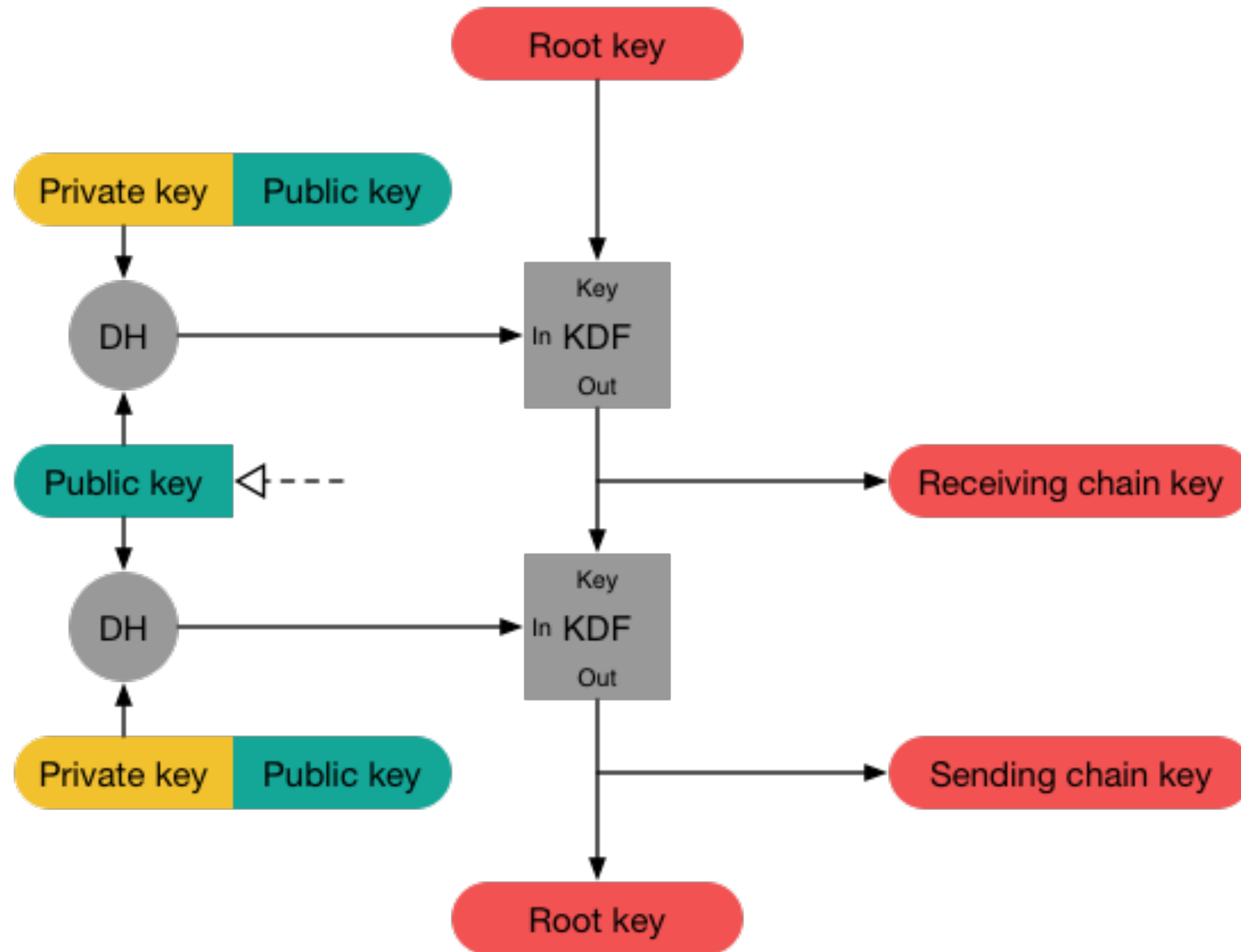


Public ratchet seeds symmetric ratchets (one per direction)

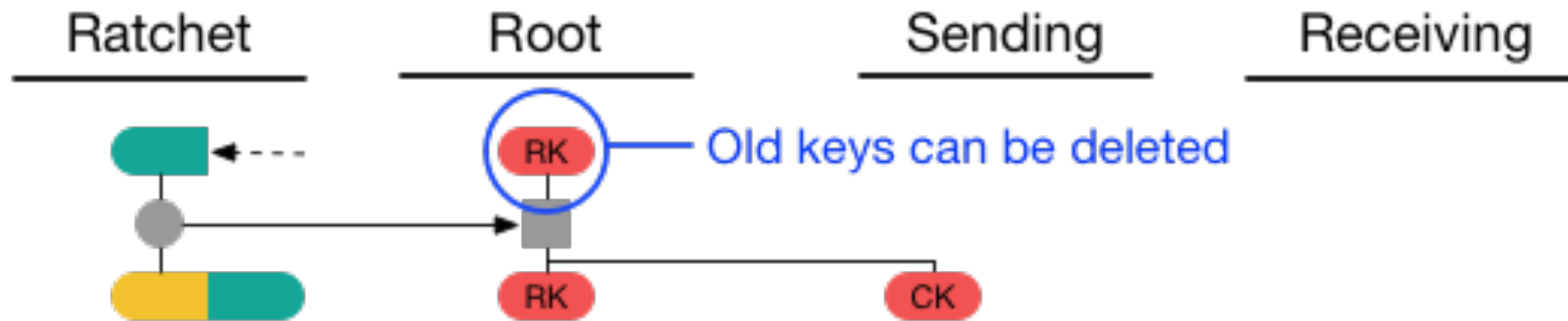


18.3 Adding Root Keys

Add a third chain to improve post-compromise secrecy



Alice's full state



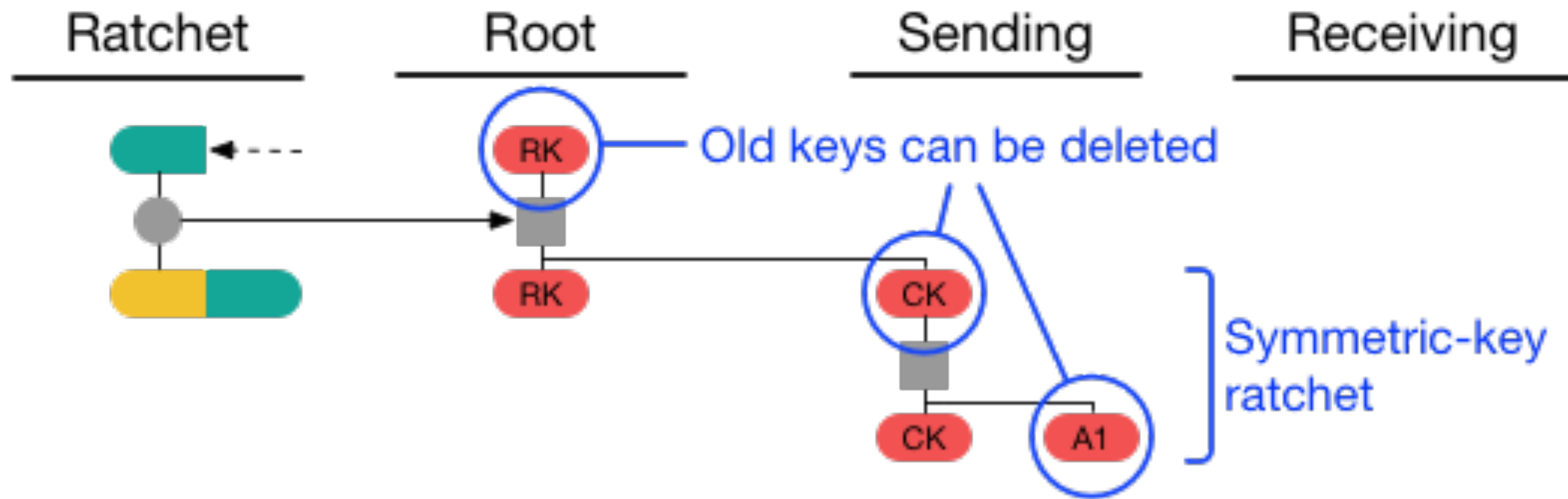
Alice has been initialized with

1. Bob's ratchet public key
2. A shared secret which is the initial root key

From this info, Alice derives

1. A new root key, whose “constant” *depends* on the public ratchet
2. The first sending key chain

Alice's full state



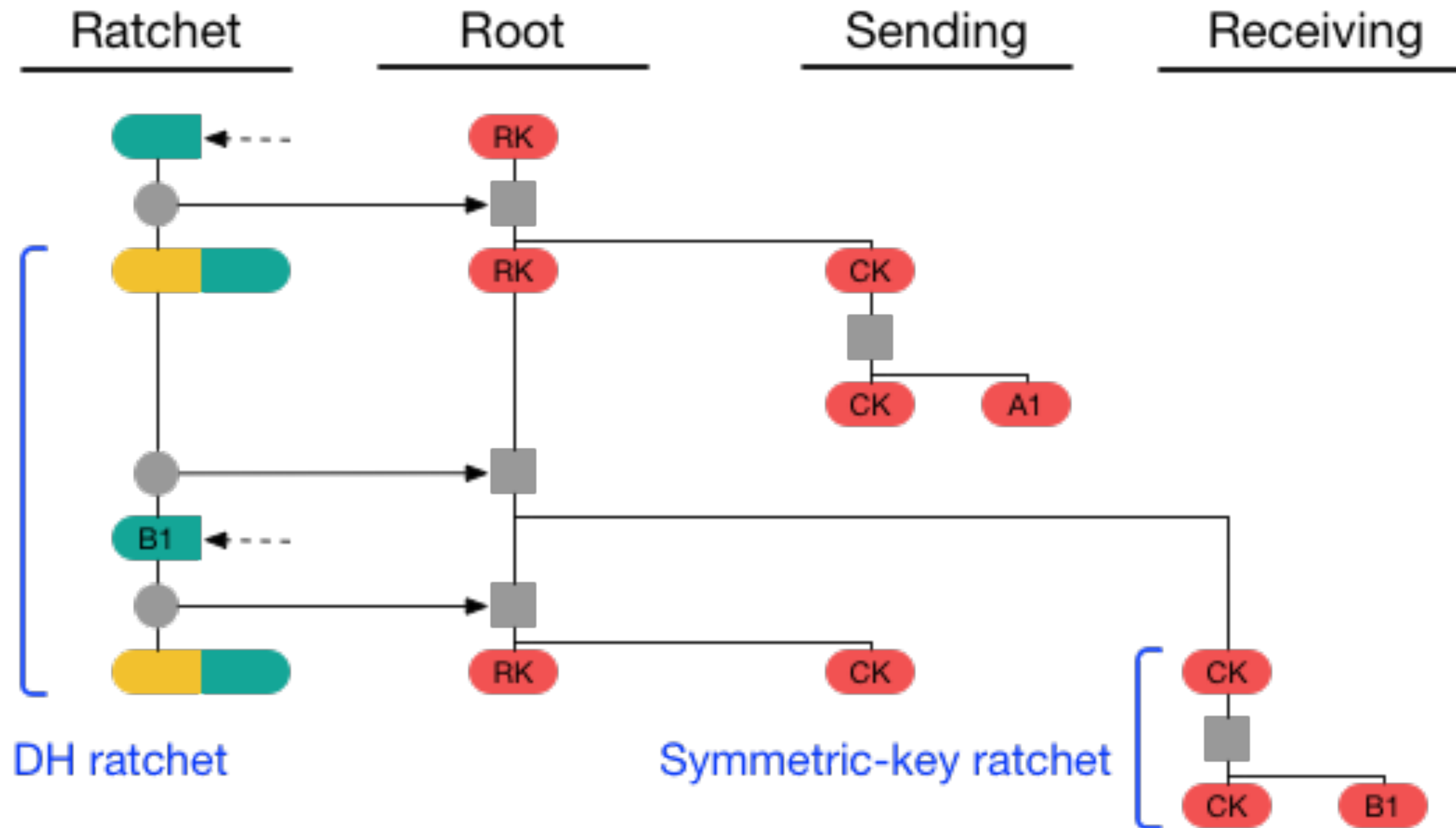
Alice sends her first message by

1. Ratcheting the sending key
2. Using the resulting *single use* message key to protect her new message


Engineering question:

Note that Bob must send  before Alice can send Bob a message.
So, how can Alice send messages if Bob isn't online?

Alice's full state



Suppose Bob sends a message next

1. Alice now has the other half of the key pair necessary to start a receiving chain
2. Bob's message includes a new , so Alice can perform a public ratchet to make a new sending chain

18.4 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



A

B



Evolve
public key

Output B^a

Output A^b



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



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

AuthEnc is
deniable



$\text{AuthEnc}_K(P_1)$

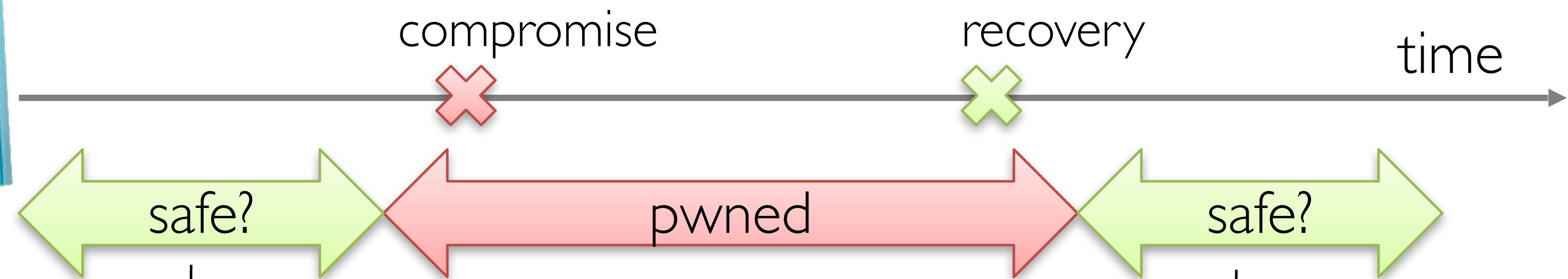
$\text{AuthEnc}_K(P_2)$

...



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.”

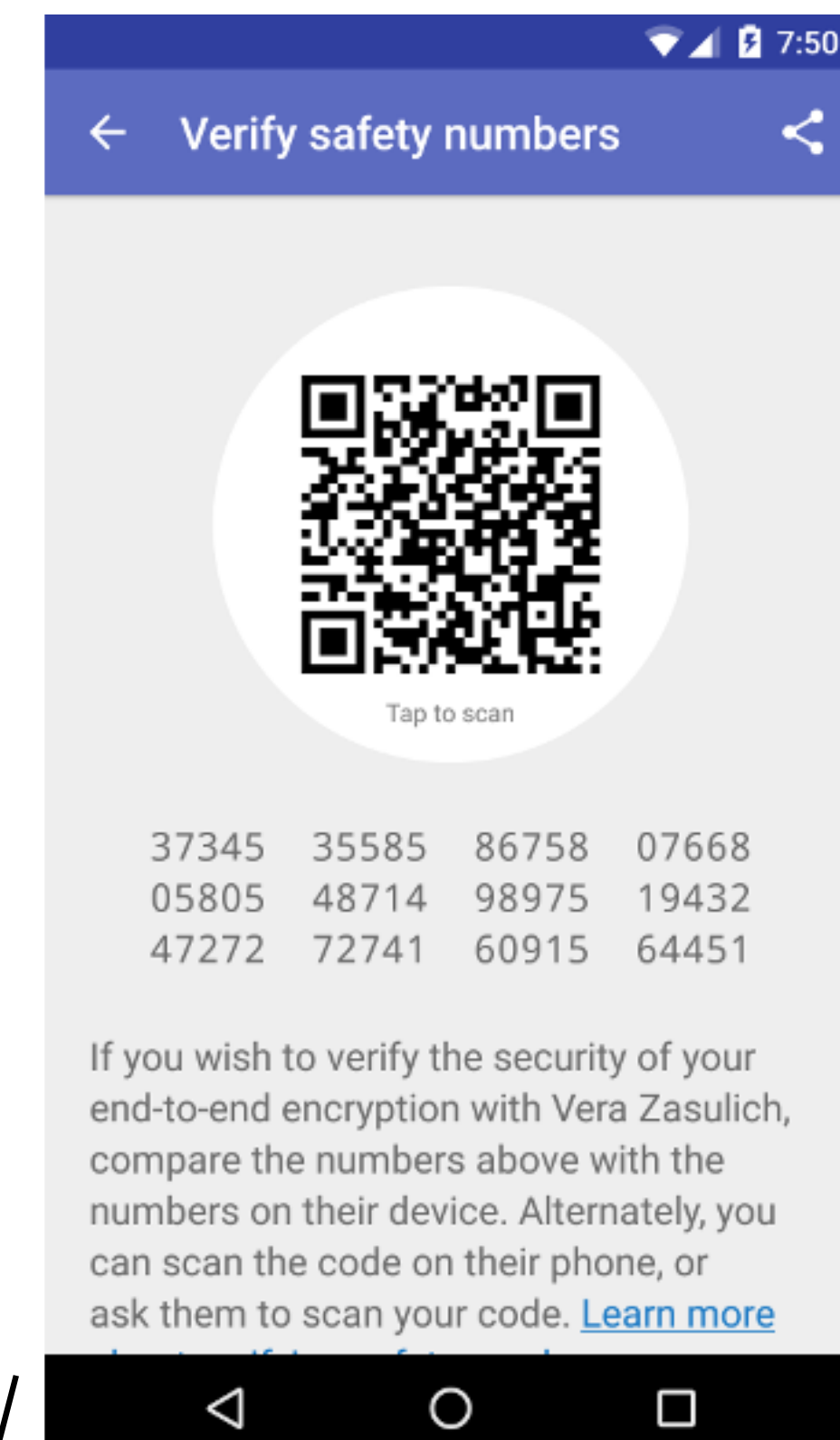
Ephemeral utopia

No long-term keys \Rightarrow great forward secrecy

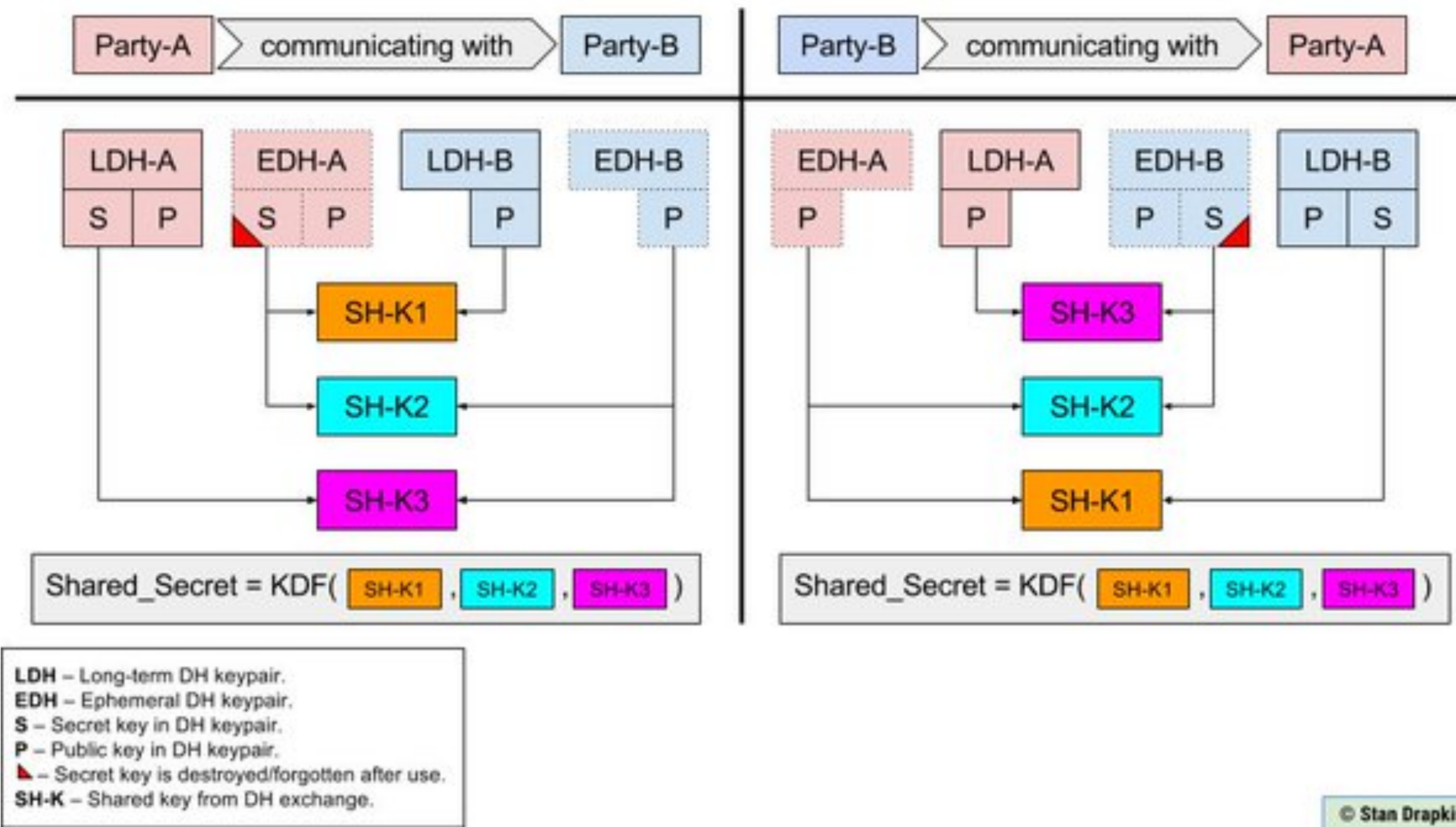
- Message key used to AuthEnc a message is used once and tossed
- Chain key used to construct msg key is refreshed in each public ratchet
- Diffie-Hellman key pairs chosen ephemeral in each public ratchet

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

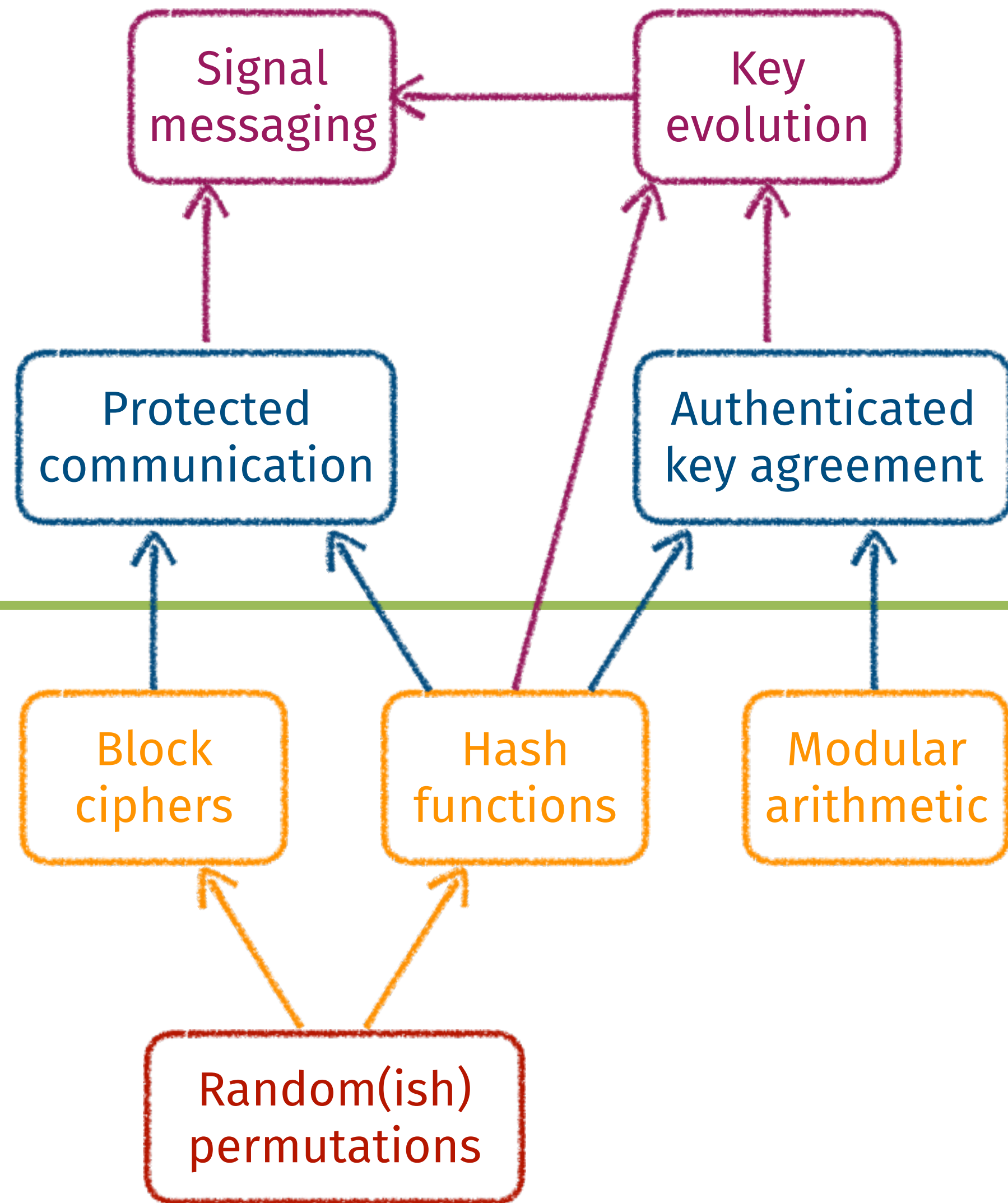


Source: <https://www.eff.org/deeplinks/2018/03/thinking-about-what-you-need-secure-messenger>

Roadmap of course so far

**Elegant
protocols**

**Utilitarian
tools**



Next week:
protecting data in use