The Signal Protocol @ VanLug

BorisReitman.com

Agenda

- OTR
- Double Ratchet
- X3DH
- Sesame

Overview

- First the parties will use X3DH key agreement protocol to agree on a shared secret key.
- Then, the parties will use the Double Ratchet to send and receive encrypted messages.

Who uses Signal?

- ChatSecure[a]
- Conversations[a]
- Cryptocat[a][9]
- Facebook Messenger[b][c][10]
- G Data Secure Chat[c][11][12]
- Gajim[a][d]
- Google Allo[e][c][13]
- Haven[c][14][15]
- Pond[16]
- Riot[f][17]
- Signal[c]
- Silent Phone[g][18]
- Skype[h][c][19]
- Viber[i][20]
- WhatsApp[c][21]
- Wire[j][22]

Understanding the Problem

Diffie-Hellman Exchange

- DH = Diffie-Hellman
- Alice sends g^x to Bob.
- Bob sends g^y to Alice.
- Both sides compute shared key: g^(xy).

Problem 1

- Alice sends g^x to Bob
- How does Bob knows that g^x came from to Alice?
- Answer: digital signatures (asymmetric) or MACs (symmetric)

MAC vs Digital Signature

- Digital Signatures are based on asymmetric keys (public / private keys)

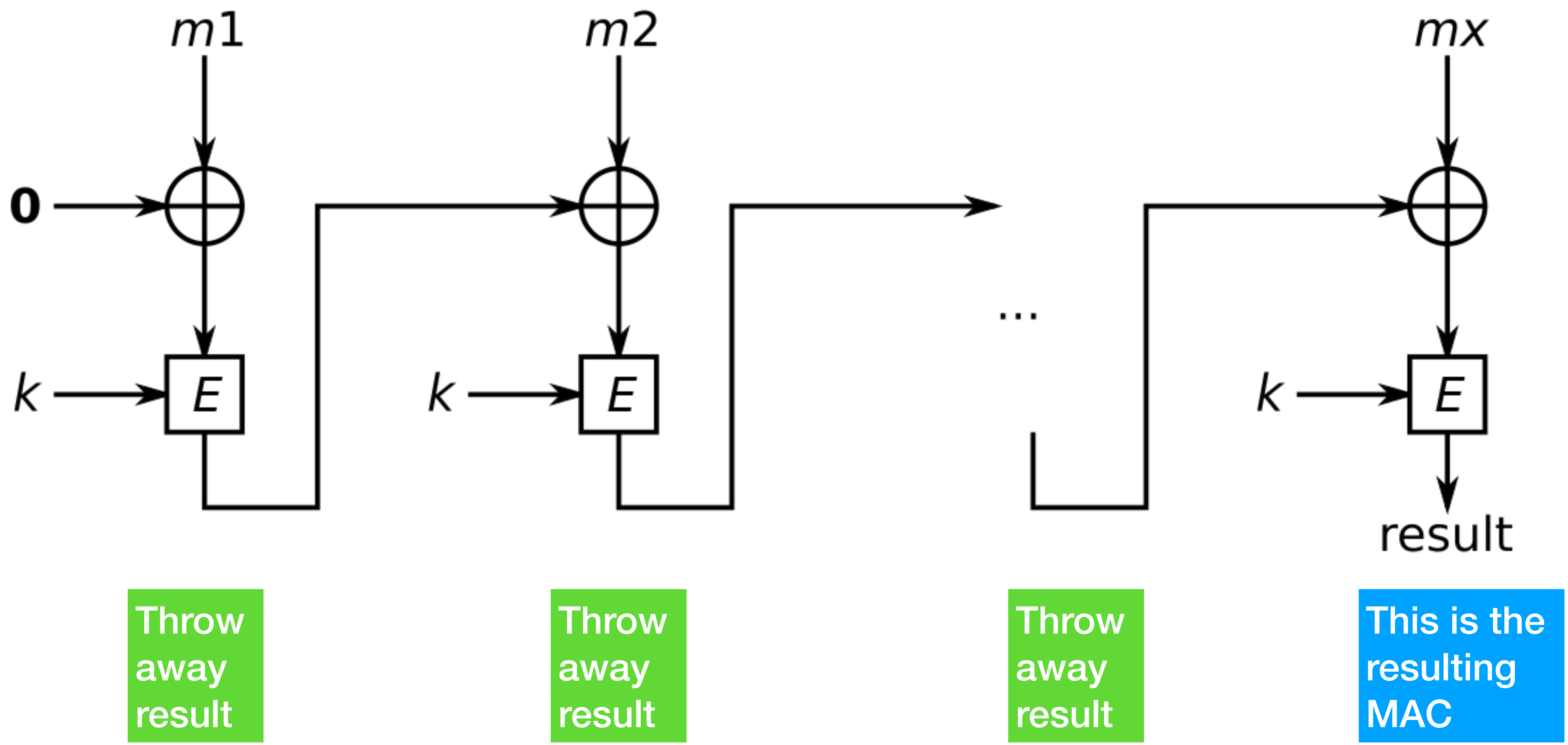
- MACs are based on a symmetric key.

RSA Digital Signature

$$(x^e)^d \equiv x \pmod{N}$$

CBC-MAC

Original Message   M = m1 || m2 || m3 || … || mx



Problem 2

- Let's say Alice and Bob are communicating securely.
- What happens if Bob's communication device is compromised?

Answer

- A compromise of encryption key exposes previously encrypted messages.
  - Solution: encrypt with temporary keys.

Answer

- A compromise of signing key does not invalidate past signatures.
- Solution:
  - Sign with long-term keys.
  - Tell your friends your long term ID public key.

Problem 3

- If Alice sends a message to Bob, she signs it.
- But that means that Alice can be blackmailed.
- Alice can’t deny that she even sent the message.
- Repudiation: ability to deny that you have sent the message.

Solution

- We don’t want to sign messages directly with main public key.
  - Only sign temporary key with public key.
  - Encrypt with temporary key.
  - We can also sign MACs.

Solution

- Alice sends a message to Bob
- She sends a MAC based on a secret key.
- Bob knows the secret key, and can reconstruct the MAC on his end, and compare.
- Bob can prove to himself that it was Alice.
    - But: Bob can’t prove this to anyone else, because he may himself made up the MAC.

Asymmetric auth + MACs

- Asymmetric keys authenticate the establishment of the first key (X3DH).
- This would tell Bob that he got DH key from Alice, indeed.
- Now, we can use MACs to authenticate individual messages, because MACs offer the Repudiation property.

Double Ratchet

OTR Paper

Off-the-Record Communication, or, Why Not To Use PGP

Nikita Borisov
UC Berkeley
nikitab@cs.berkeley.edu

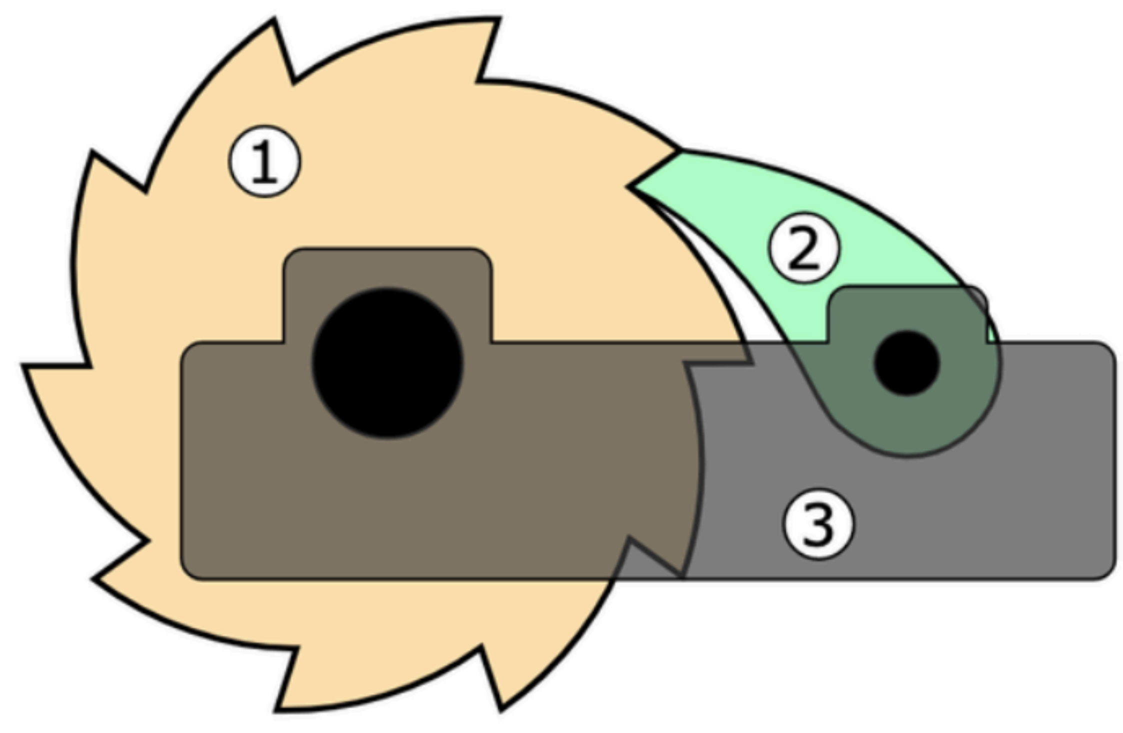
Ian Goldberg
Zero-Knowledge Systems
ian@cypherpunks.ca

Eric Brewer
UC Berkeley
brewer@cs.berkeley.edu

Paper from 2004

OTR DH Ratchet

- The encryption keys roll forward, as messages are received.
- Alice sends Bob message #5 encrypted with key #4, and also includes DH value to make key #5.
- Bob deletes key #4 after he decrypted the message #5.
- Bob replies with message #6 encrypted by key #5, and also includes DH value to make key #6.



Example

Note: the notation here is
using powers instead of multiplying by scalar.

The next round of DH pub key g^x or g^y
is sent together with each message.

$$A \longrightarrow B: \quad g^{x_1}$$

$$B \longrightarrow A: \quad g^{y_1}$$

$$A \longrightarrow B: \quad g^{x_2}, E(M_1, k_{11})$$

$$B \longrightarrow A: \quad g^{y_2}, E(M_2, k_{21})$$

$$A \longrightarrow B: \quad g^{x_3}, E(M_3, k_{22})$$

The encryption
key k is derived
from the last received and
last sent g^x and g^y values.

Problem with OTR

- DH ratchet advances based on responses only.
- What if Alice sends many messages, while Bob is just reading them, but does not reply?
- Possible fix: Bob should send automatic empty messages.
- Another idea? … ratchet each message from Alice using a KDF ratchet, until Bob answers.

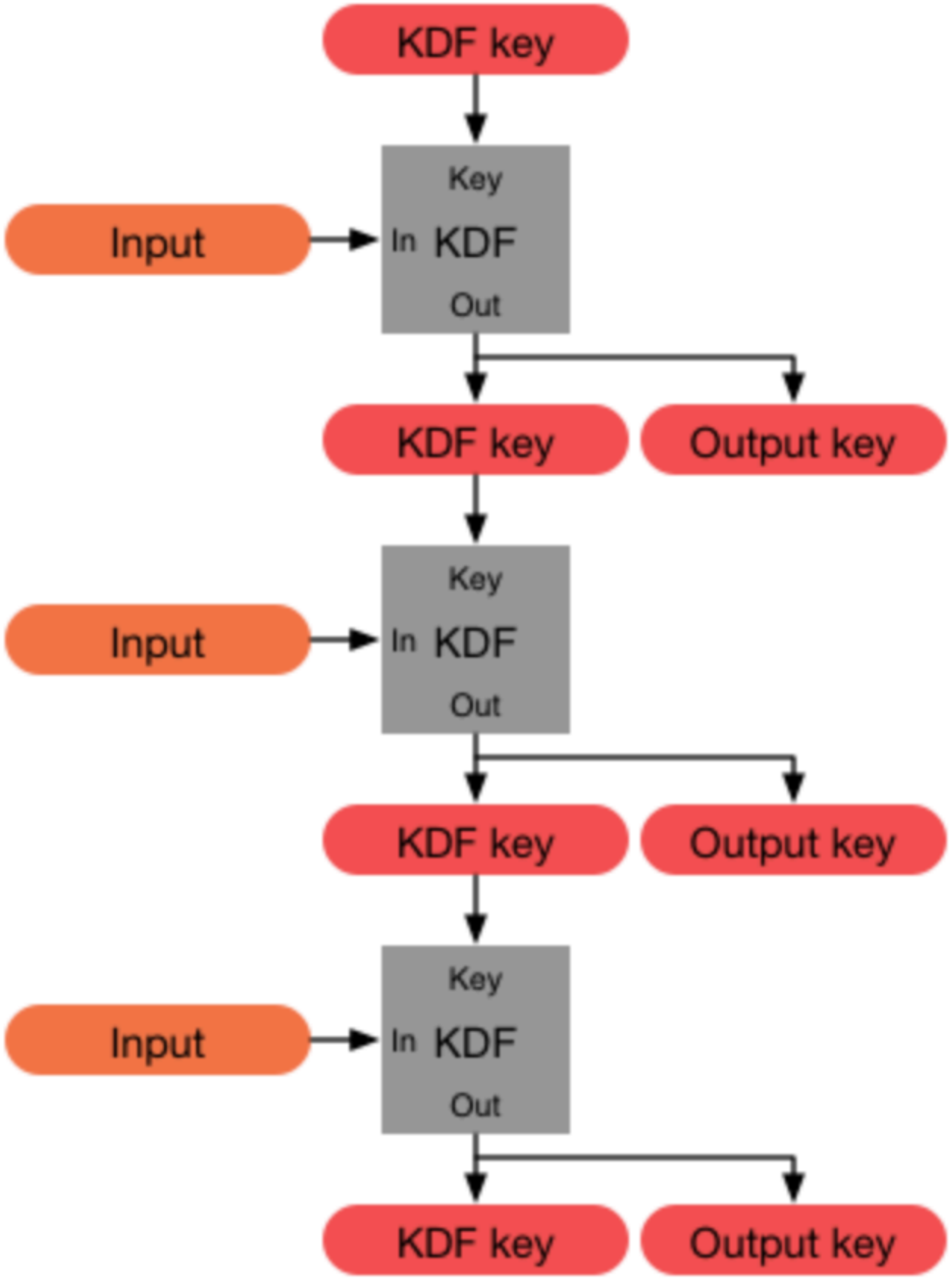
HMAC KDF

- Use HMAC to derive many keys from the same secret key.
- Each key derived key does not reveal the others.

```
H0(AlicePassword)≡HMAC(AlicePassword,0x00)
H1(AlicePassword)≡HMAC(AlicePassword,0x01)
H2(AlicePassword)≡HMAC(AlicePassword,0x02)
H3(AlicePassword)≡HMAC(AlicePassword,0x03)
```

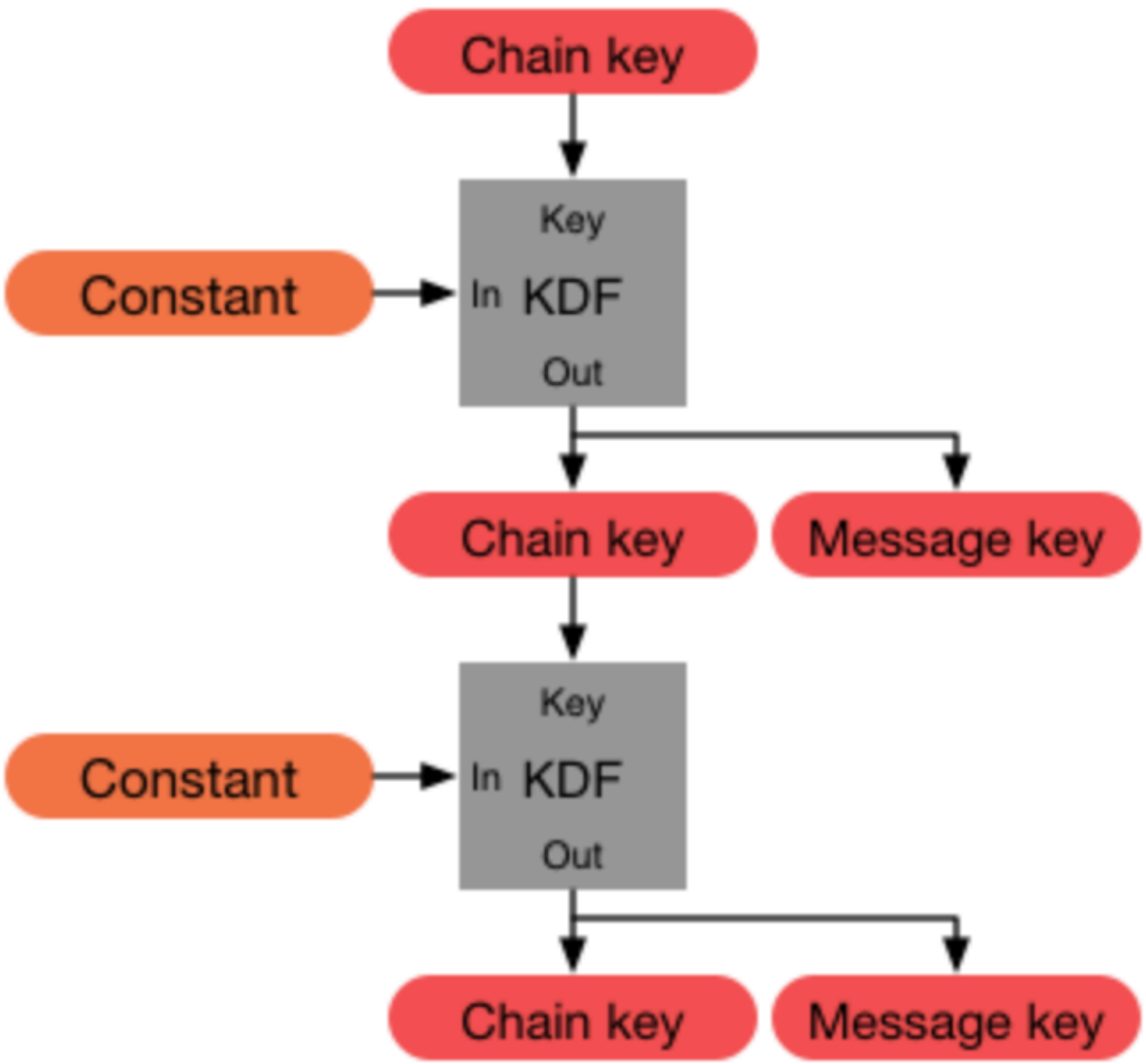
H0, H1, H2, H3 are resulting independent secret keys.

KDF chain



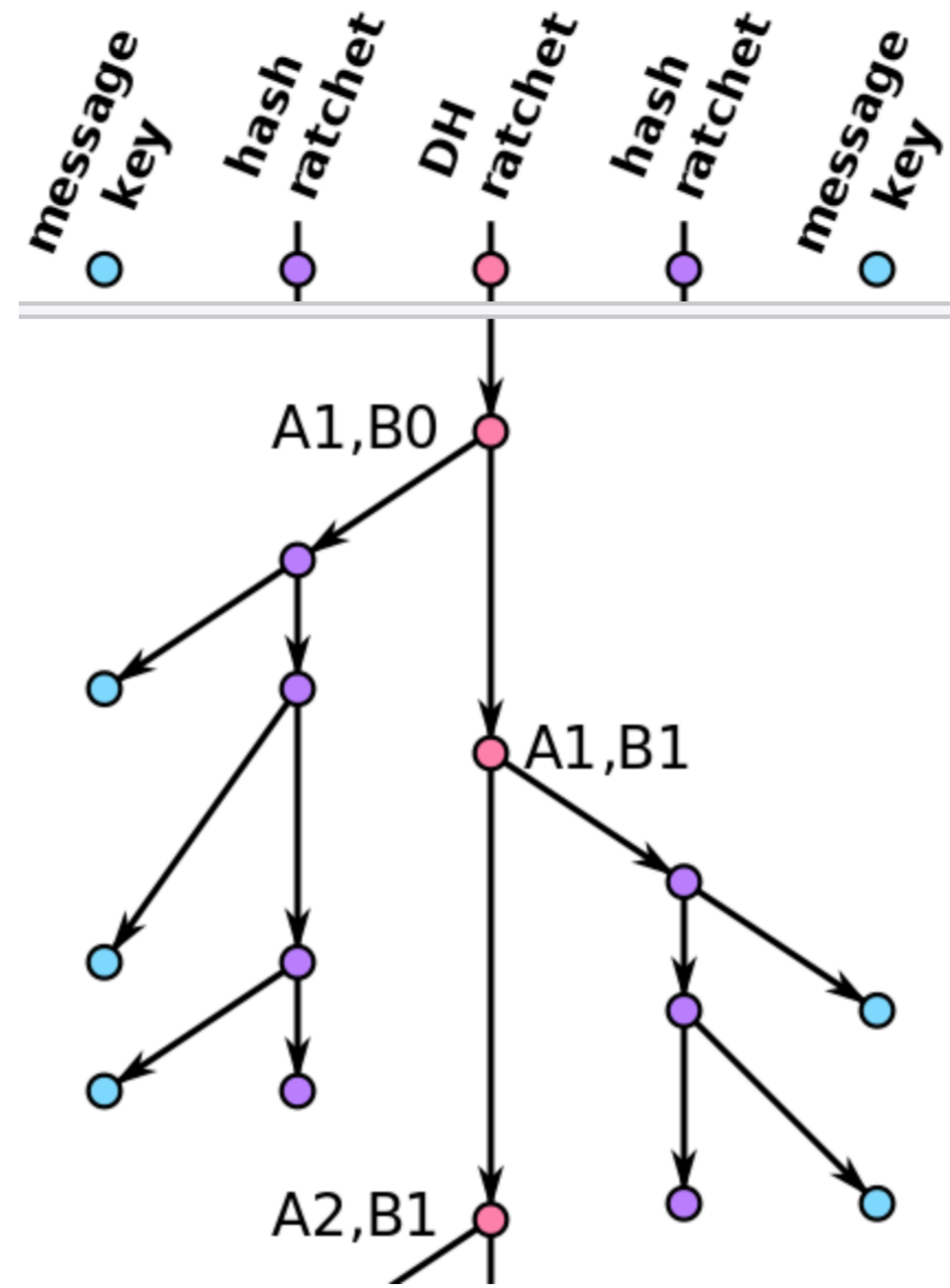
Part of the output is used as output key, and another part as another KDF key for the next round.

Using constant as “input”



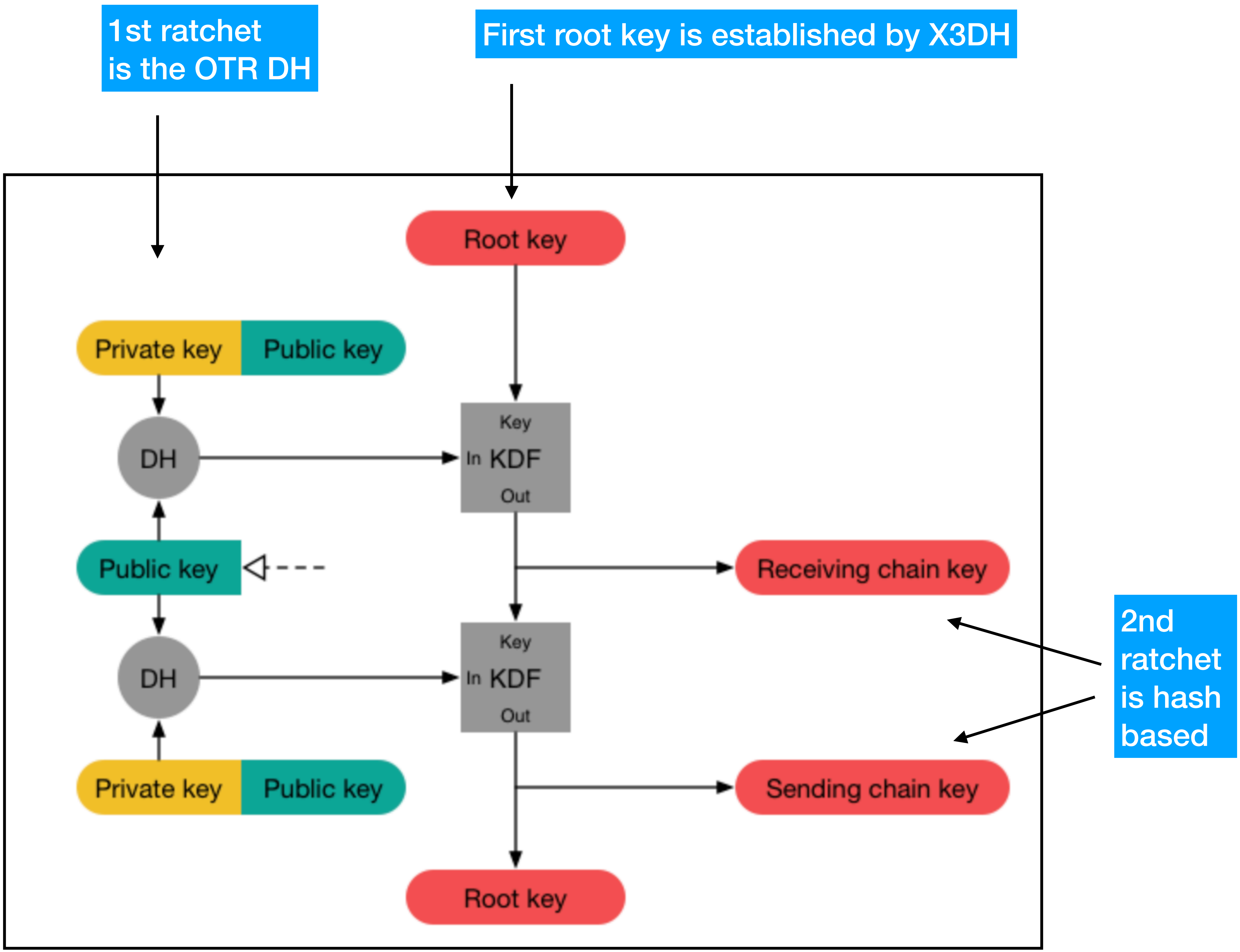
Double Ratchet

- First level: DH ratchet
- Second level: KDF ratchet

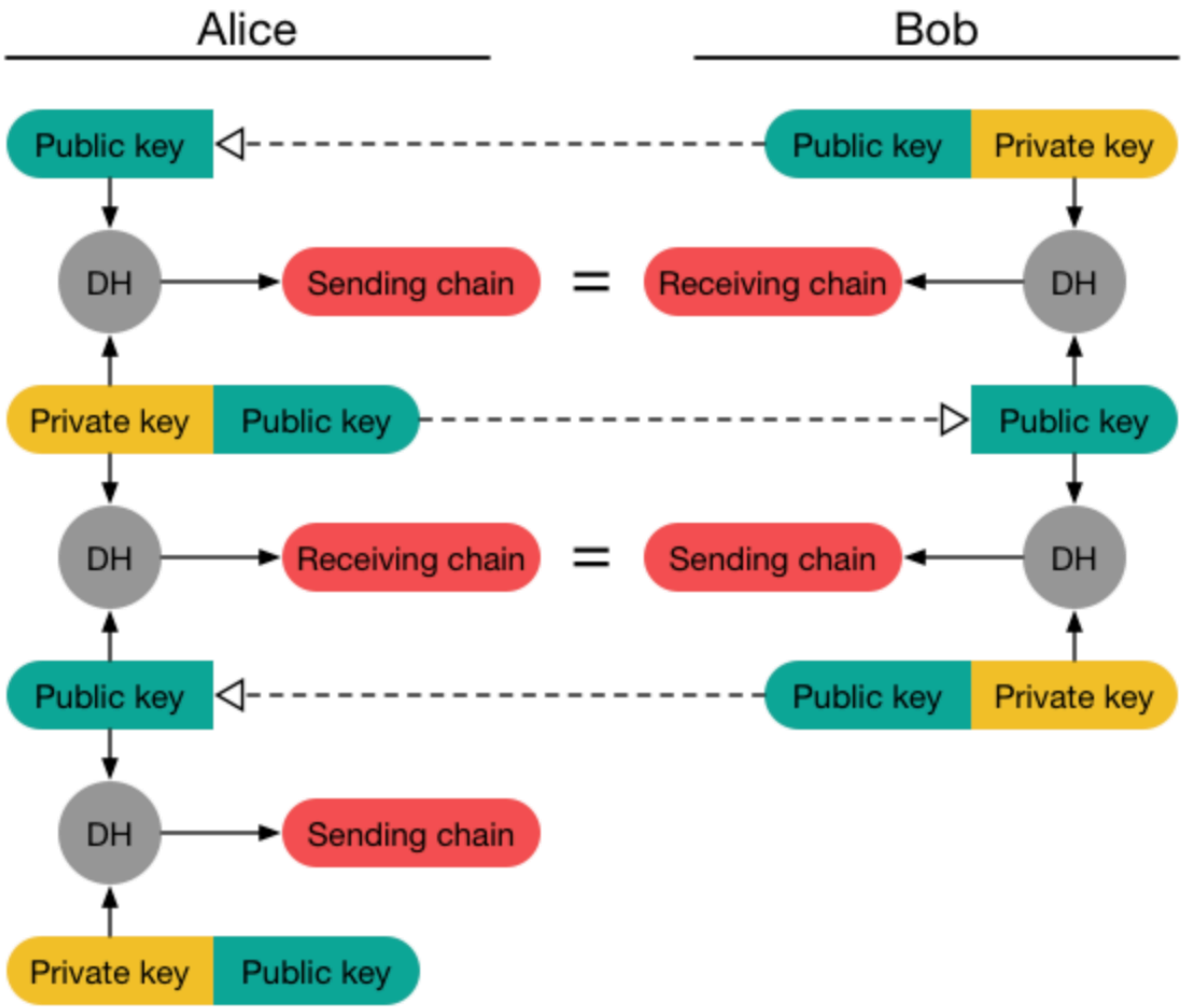


Terminology

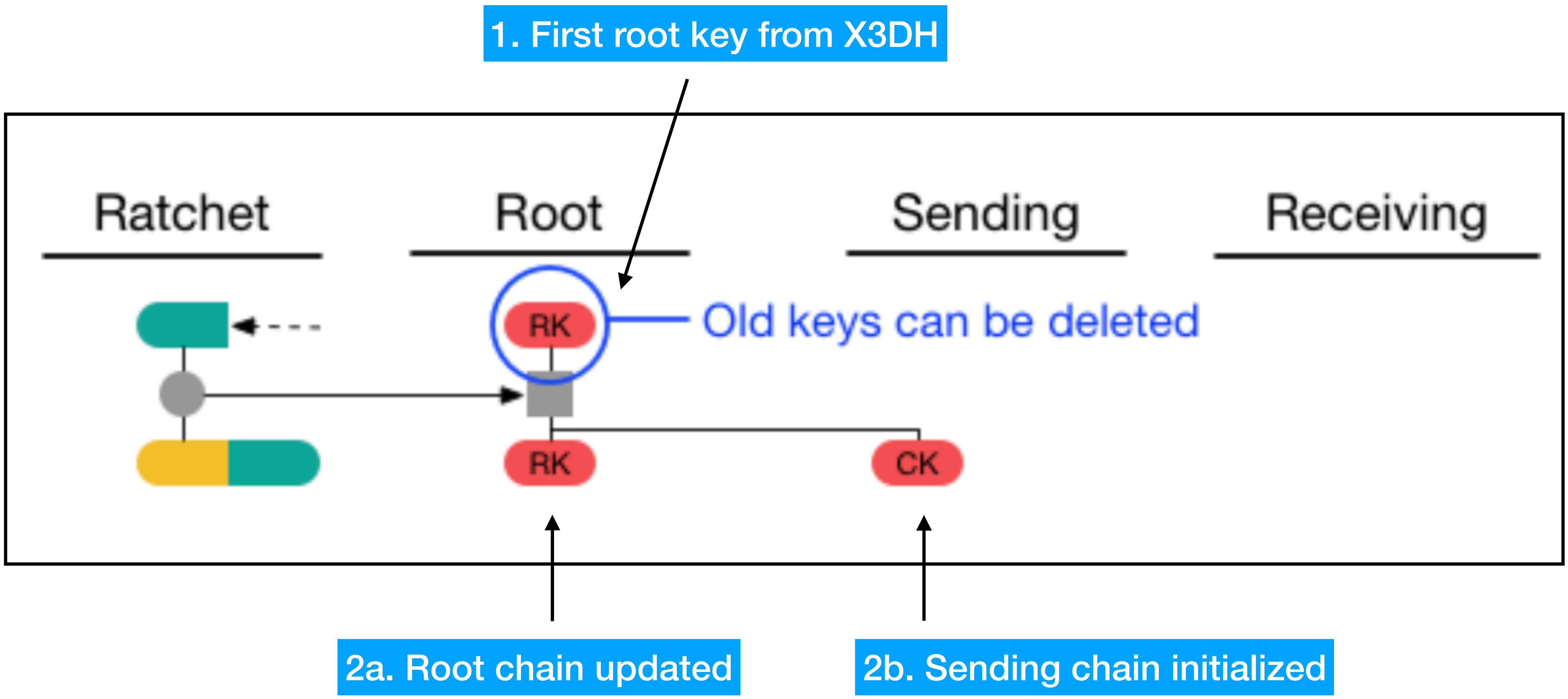
- 1st ratchet = DH ratchet (like OTR)
- Root KDF chain
  - maps DH shared keys to starting keys in 2nd-level ratchet.
- 2nd ratchet = Symmetric-key ratchet
  - sending ratchet (sequence of keys used to encrypt)
  - receiving ratchet (sequence of keys used to decrypt)


1st ratchet
is the OTR DH
First root key is established by X3DH
Root key
Private key
Public key
DH
Key
In KDF
Out
Public key
Receiving chain key
DH
Key
In KDF
Out
Private key
Public key
Sending chain key
Root key
2nd
ratchet
is hash
based

Sending chain = Receiving chain

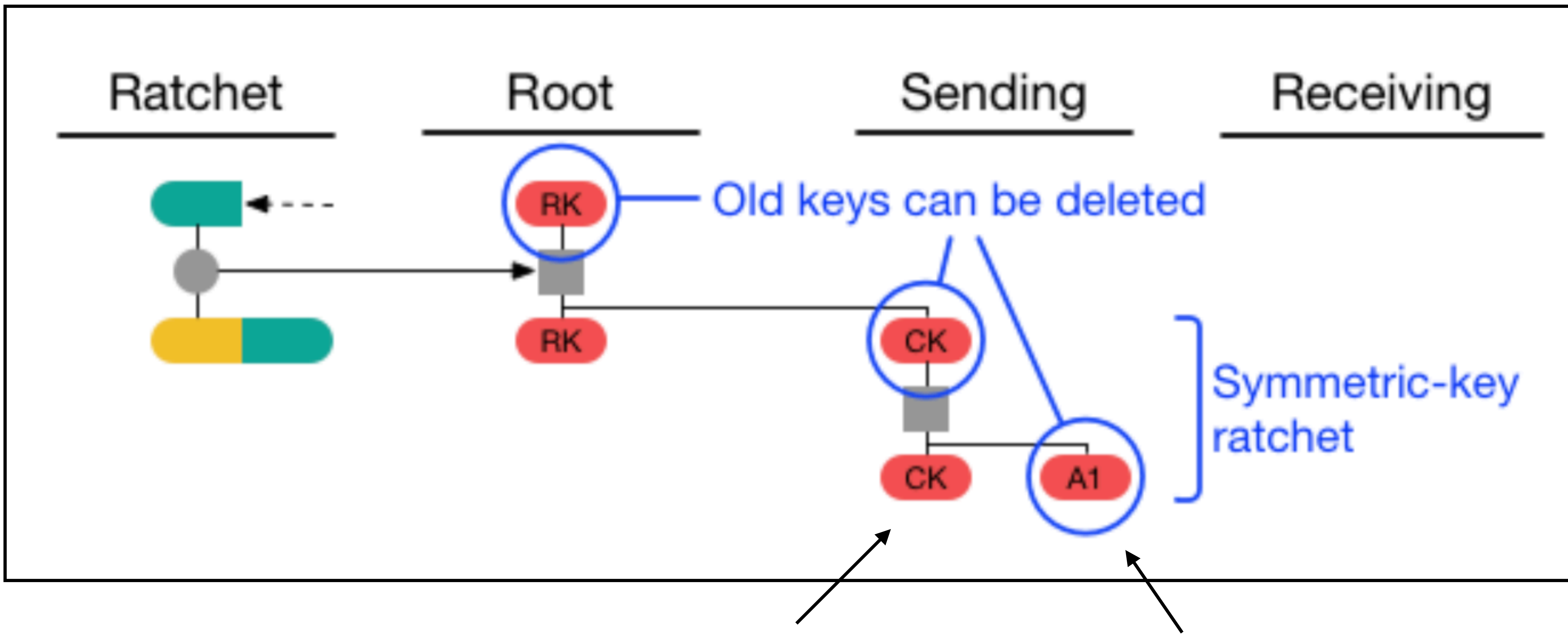


Initialize



1. First root key from X3DH
Ratchet
Root
Sending
Receiving
RK
Old keys can be deleted
RK
CK
2a. Root chain updated
2b. Sending chain initialized

Sending 1st message

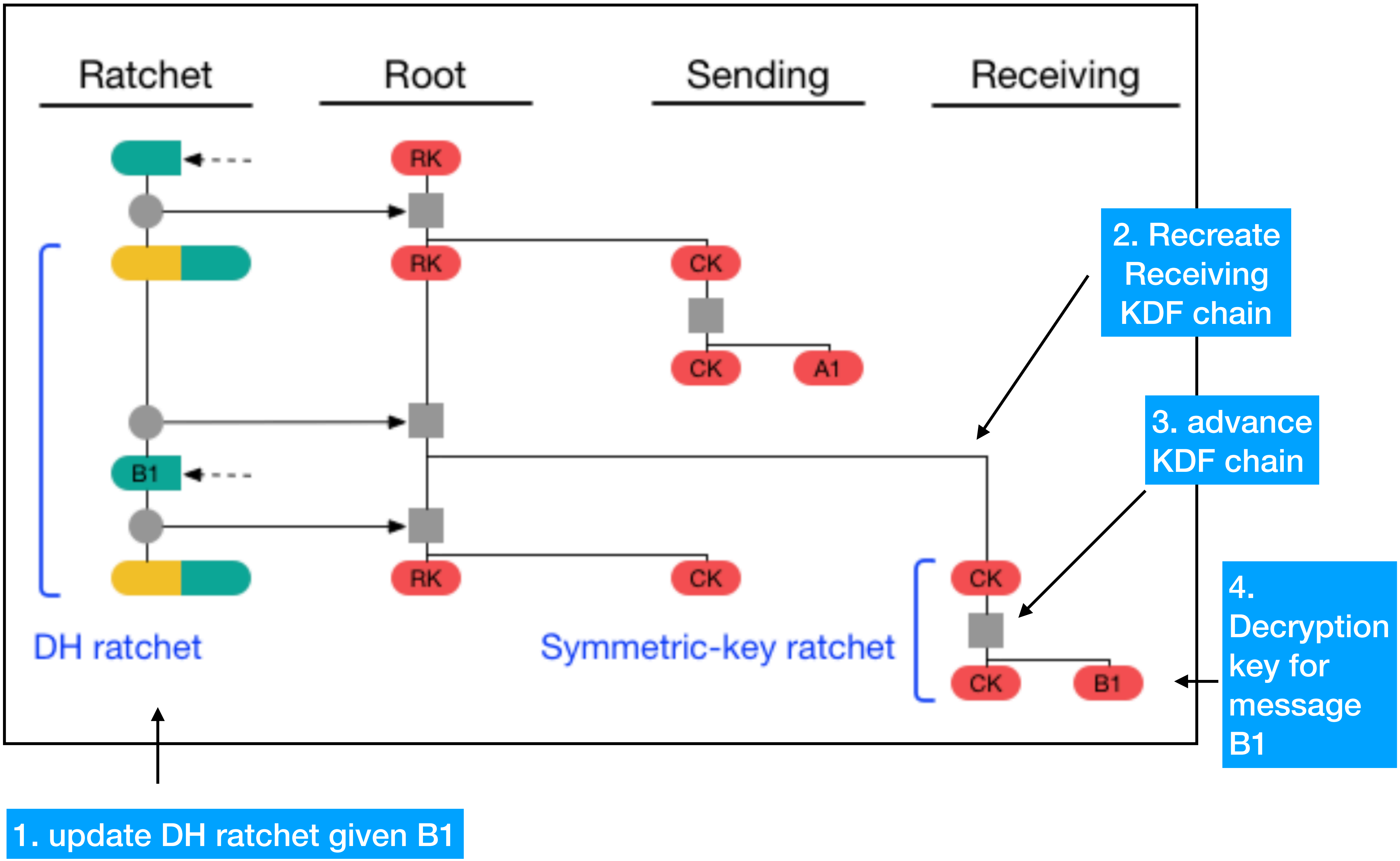


Move chain forward

Get message key A1

Use A1 to encrypt, then discard it.

Handle a reply from Bob



X3DH

X3DH

- X3DH = Extended Triple Diffie-Hellman
- “X3DH is designed for asynchronous settings where one user (Bob) is offline but has published some information to a server. Another user (Alice) wants to use that information to send encrypted data to Bob, and also establish a shared secret key for future communication.”

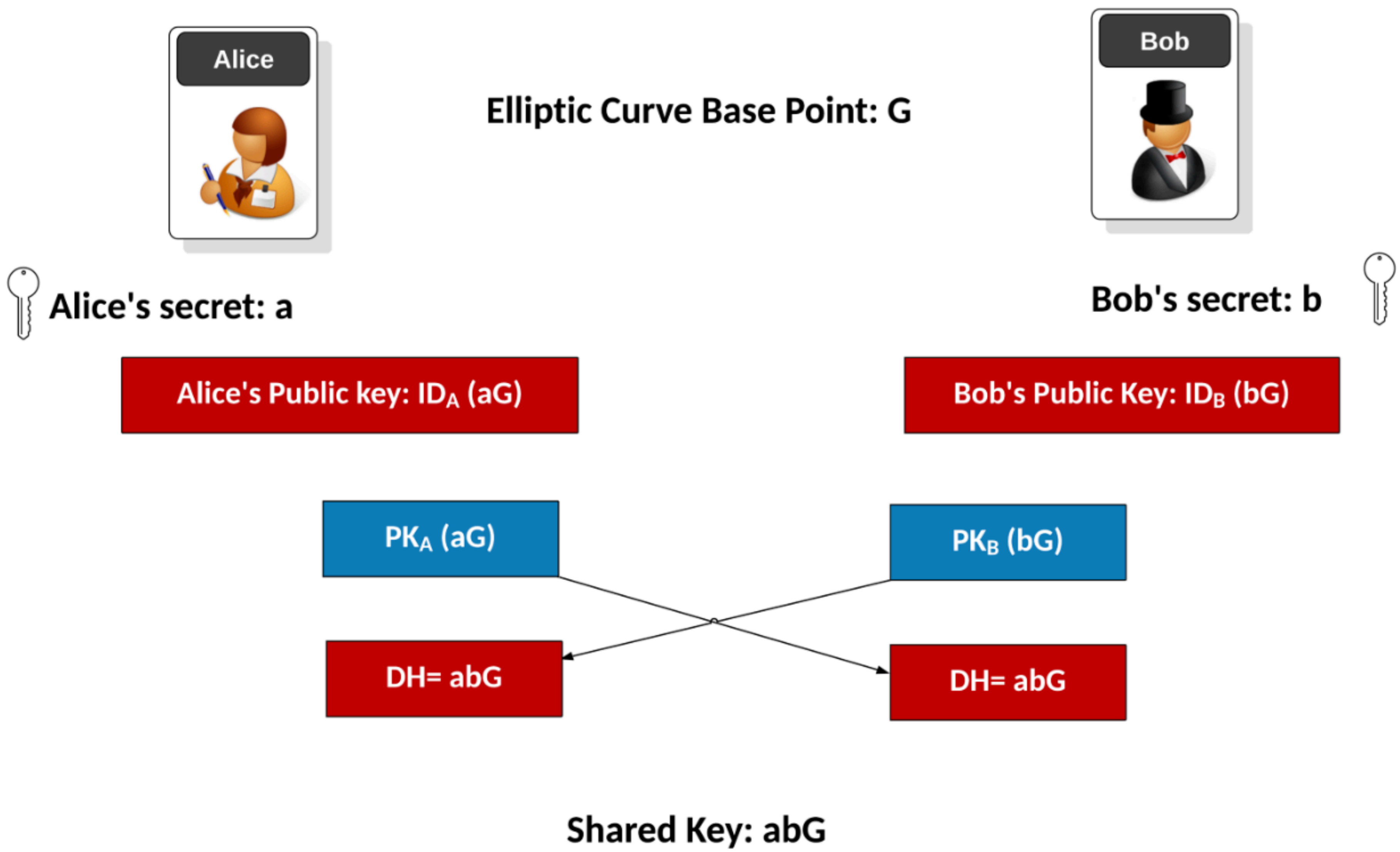
Group Operation

- A set of elements is a group if they can be combined by some operation, and the result is still in the set.
- If the operation is "+", the notation is C = A + B
- If the operation is "x", the notation is C = AB

Group Operation

- If the operation is “+”, then: A + A + A = 3A
- If the operation is “x”, then: A x A x A = AAA = A^3

Basic ECDH


Alice
Elliptic Curve Base Point: G
Bob
Alice's secret: a
Bob's secret: b
Alice's Public key: ID_A (aG)
Bob's Public Key: ID_B (bG)
PK_A (aG)
PK_B (bG)
DH= abG
DH= abG
Shared Key: abG

Offline Problem

- Alice can’t form the shared key unless Bob has sent her “bG”.
- But what if Bob is offline?
- To send message to Bob, Alice must wait to get “bG” from Bob.
- She can’t send him a message until then.

Possible Solution:

- Bob must send “bG” to Alice in advance, before he goes to sleep.
- Intermediate server must hold it until Alice needs it.
- The server must be trusted not to mix up the keys.

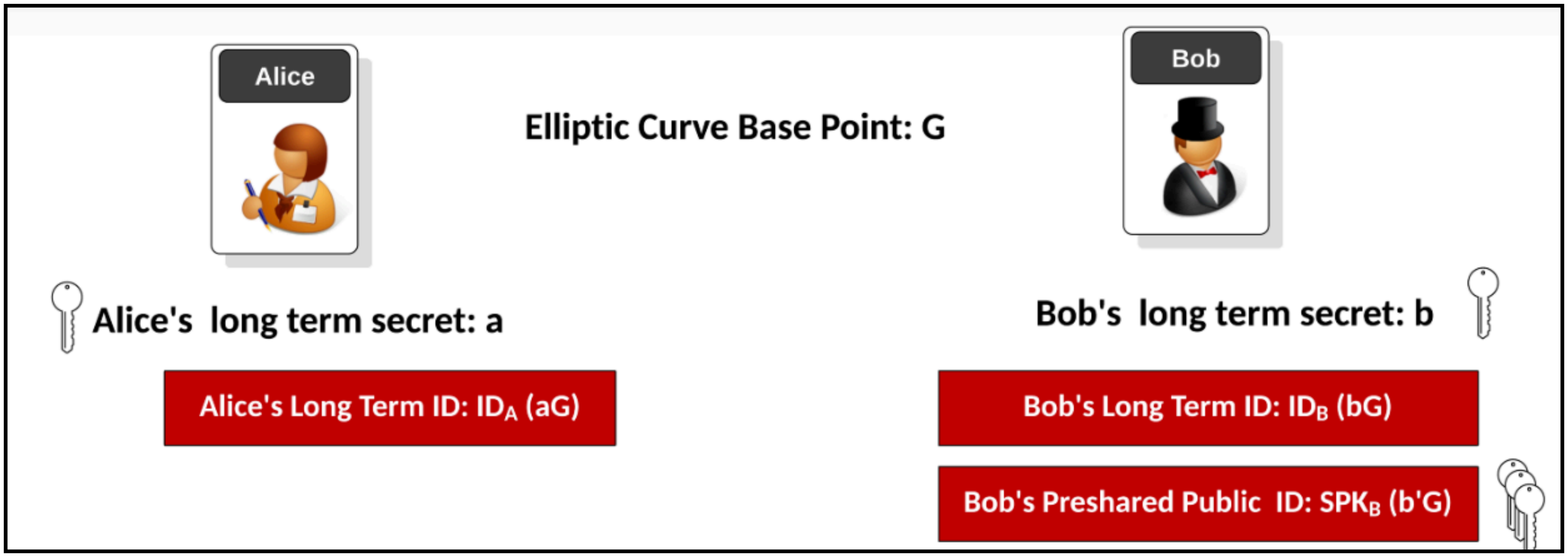
Forward Secrecy

- FS = Forward Secrecy
- Forward secrecy means that if an encryption key is stolen, then it won't help decrypt past messages.
  - We want separate keys for each communication session.

So?

- This means that Bob must upload to the server several keys:
    - b1*G,
    - b2*G,
    - b3*G,
    - …

Setup

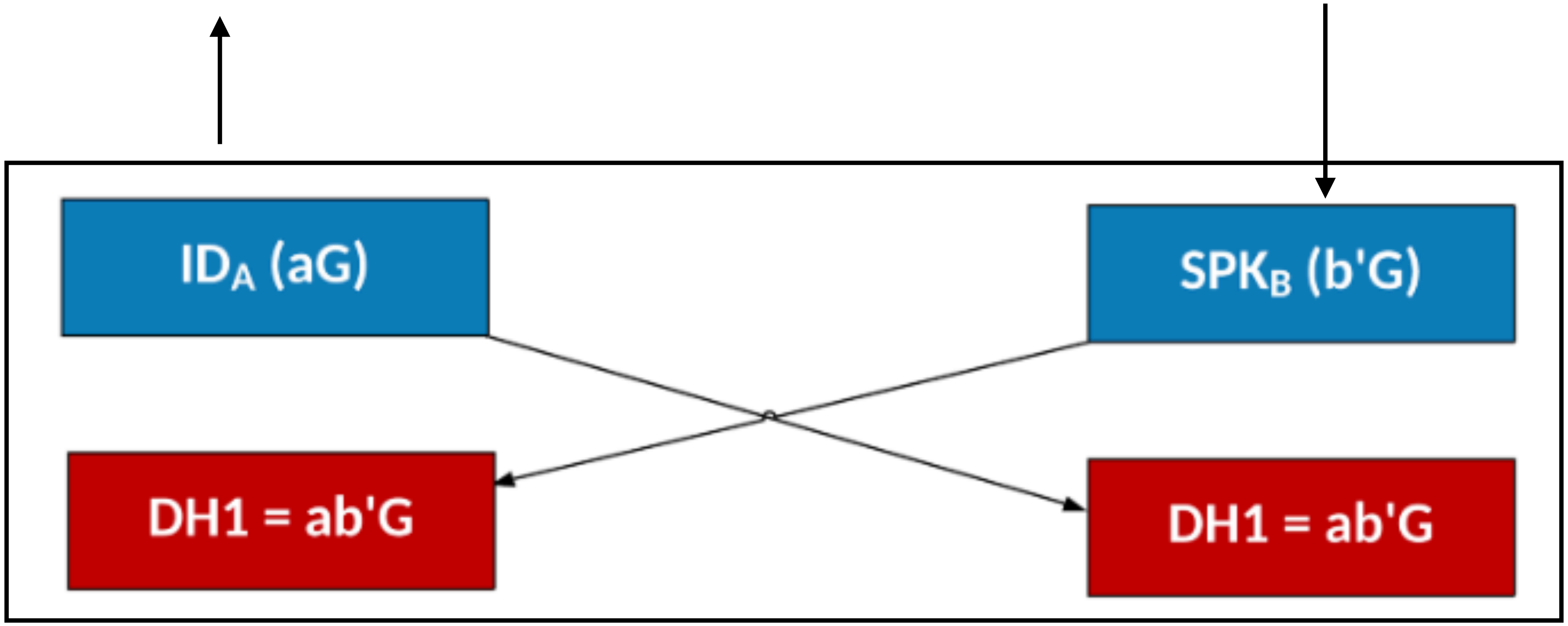


- Identity keys ID_A and ID_B are not used directly to form shared session key.
- Using them directly => same session key for every session

1st DH {a, b’}

Bob would know Alice’s ID when he wakes up.
It’s part of Alice’s contact entry metadata.

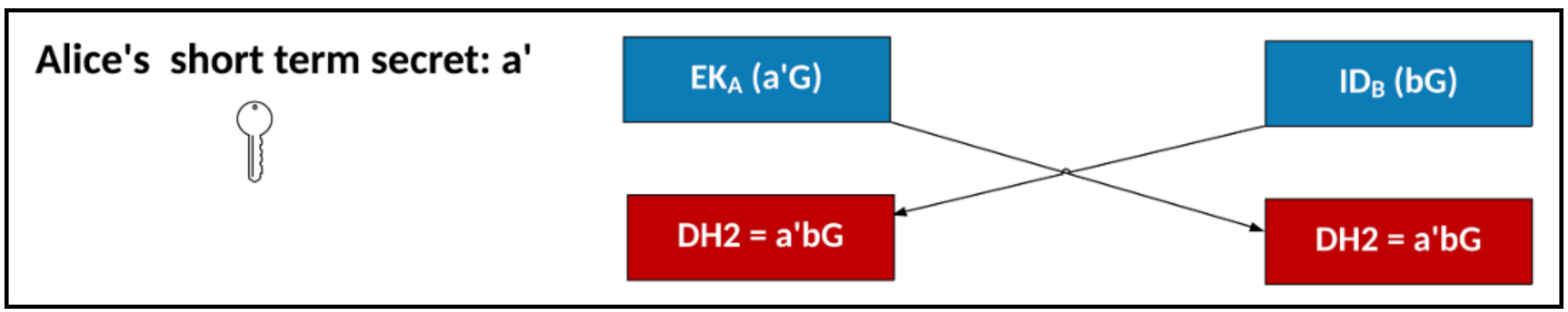
Bob uploaded this to server,
before he went offline.



SPK = pre-shared Public Key

There are many signed pre-keys SPK_B on the server that belong to Bob. Alice chooses a random one.

2nd DH { a’, b }

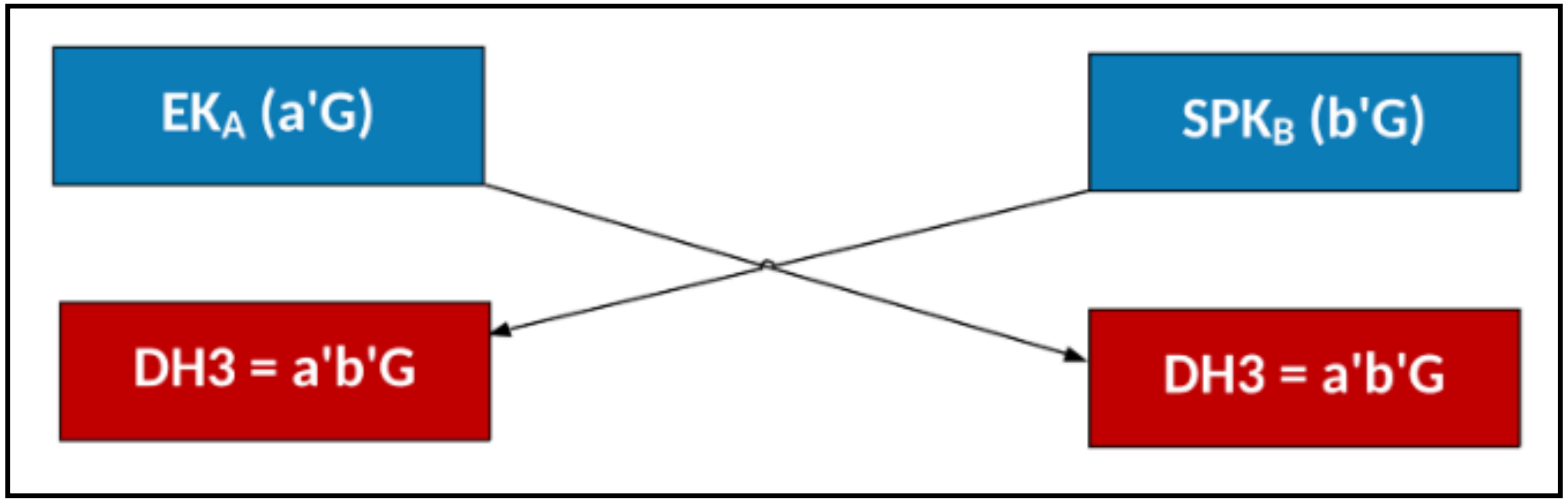


EK = Ephemeral Key

- **Alice knows Bob’s ID.**
- **It’s part of Bob contact entry metadata.**

3rd DH { a', b' }

This is the basic DH exchange, with ephemeral keys.
This results in a new session key every time.



Result:

- Shared key = KDF(ab'G || a'bG || a'b'G)
- KDF = key derivation function

How to use?

- Alice will send her encrypted message, together with:
  - her ID public key: ID_A = aG
  - her ephemeral public key: EK_A = a’G
  - the signed pre-key of Bob that she decided to use: SPK_B = b’G

When Bob wakes up

- Bob needs to construct the same key that Alice used to encrypted the message, from the parts he received.
- He received the public key of his that Alice chose SPK_B = b'G.
- He looks up in his database and finds the corresponding private key: b'
- He verifies that ID of Alice aG matches his contact info for Alice.
- He accepts the ephemeral key EK_A = a'G that Alice sent him.
- He can form now: ab'G || a'bG || a'b'G.

Fourth DH exchange

- There is an additional fourth DH exchange that uses One-Time Prekey.
- Its output value is concatenated to the other DH outputs.
- Key = KDF (DH1 || DH2 || DH3 || DH4)

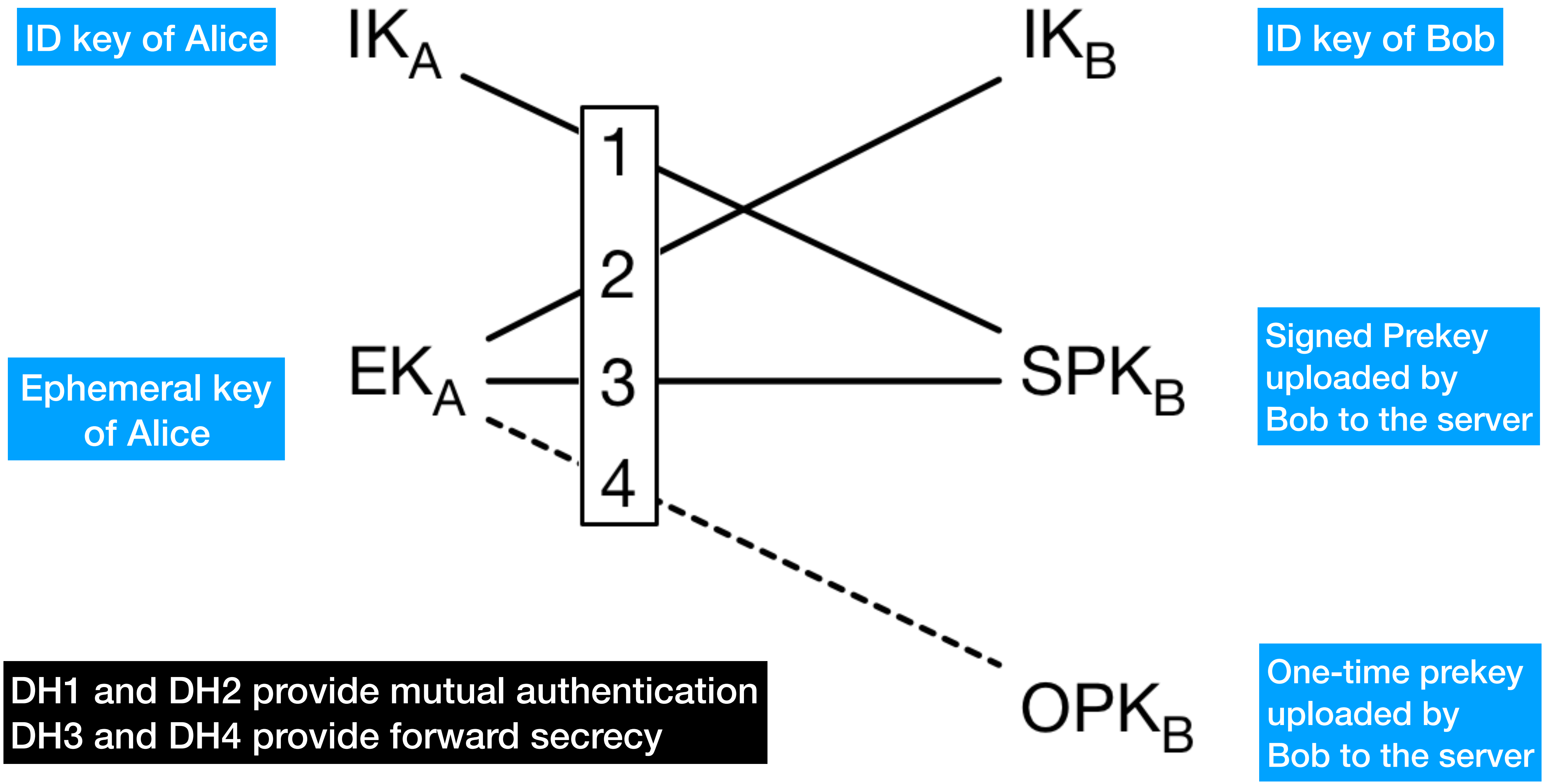
One-Time Prekeys

- OPK_B in the docs. (Bob’s one-time prekey)
- Bob uploads a bunch of them to the server.
- Once used by Alice, they are removed from the server.
- When they are fully depleted, Bob will upload more.
- If there’s none left on the server, Alice will not do the fourth DH exchange.

Etimology of “Pre-key”

- Normally a protocol is described interactively.
- Here, Bob publishes a key to the server, before Alice is going to initiate the protocol run.
- Therefore, the name is “pre-key”.

X3DH: Four DH Exchanges



Result Session Key = KDF(DH1 || DH2 || DH3 || DH4)

Sesame

Docs

The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [PDF]

Moxie Marlinspike, Trevor Perrin (editor)

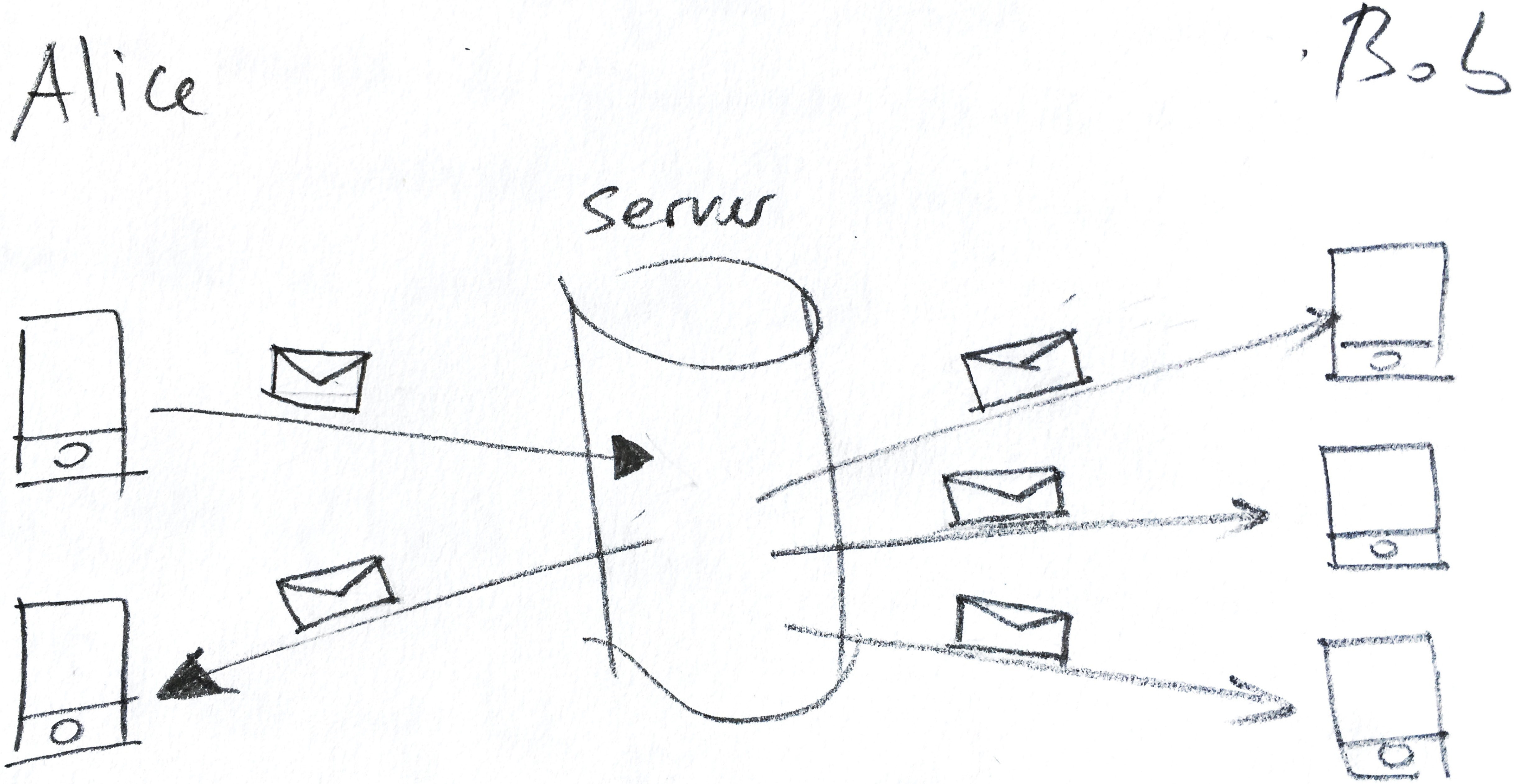
Issues

- Alice has multiple devices.
- Alice & Bob may simultaneously initiate a conversion with each other.
- Alice may erase her device, making Bob have info about Alice’s keys, causing a mismatch.
- Messages may be lost; may arrive out of order; clock synchronization.

How to manage state?

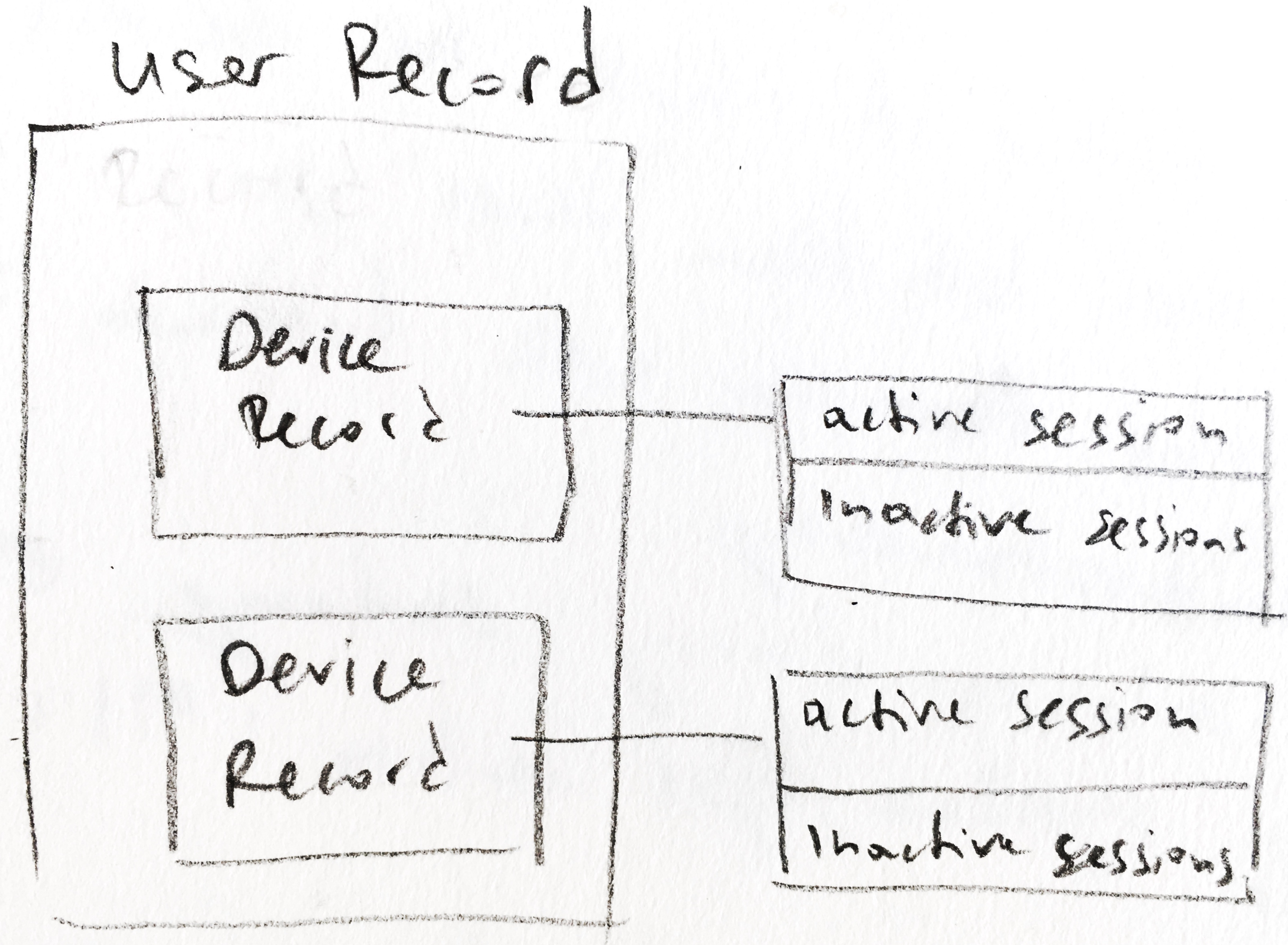
- Each user can have multiple devices.
- Each device has its own chain of keys
- The sending and receiving chains must match in state across communicating devices.
- Used keys should be deleted, but not too soon to allow for delayed messages.
- Server must hold data for offline devices: messages and prekeys.

Sending Message



Sesame Terminology

- User has a single UserRecord
- UserRecord contains many DeviceRecords
- DeviceRecord contains many Sessions
- Session contains states of the ratchets


User Record
Device Record
active session
inactive sessions
Device Record
active session
inactive sessions

Sesame Server

- Server stores current record of all users and devices.
- Server stores a mailbox per each device of queued messages.

What’s stored on Device

- Device stores UserRecords for other peers.
- Device stores its own UserRecord
- Device does *not* store it’s own DeviceRecord, but only DeviceRecords of other devices that belong to this user.

Sending message

- Find UserRecord for target user.
  - For each DeviceRecord in the UserRecord
    - Encrypt using the active session in the DeviceRecord.
- Send each ciphertext per target DeviceID to the server.

Identify a target

- The tuple (UserID, DeviceID, identity public key)
  - Identifies a target record to be updated
  - Or a target messages recipient
- The public key must match a record identified given (UserID, DeviceID). If mismatched, a new record is created, and replaces the old one.

Active Session

- Each DeviceRecord tracks the currently active session.
- Inactive sessions are kept around to decrypt delayed messages.
- Inactive sessions will be eventually deleted.

Message Authentication Codes

Encryption vs Authenticity

- Not everything needs to be encrypted.
- Sometimes the data is public. We just need to know if it is authentic.
- Example: Title to a house.

MAC

- MAC = Message Authentication Code
- ~~MAC = Media Access Control (networking)~~
- MAC is a checksum of some text mixed with a secret key.
- Used is symmetric encryption, for each encrypted block of data.

MAC vs Digital Signature

- Digital Signatures are based on asymmetric keys (public / private keys)
- MACs are based on a symmetric key.

HMAC

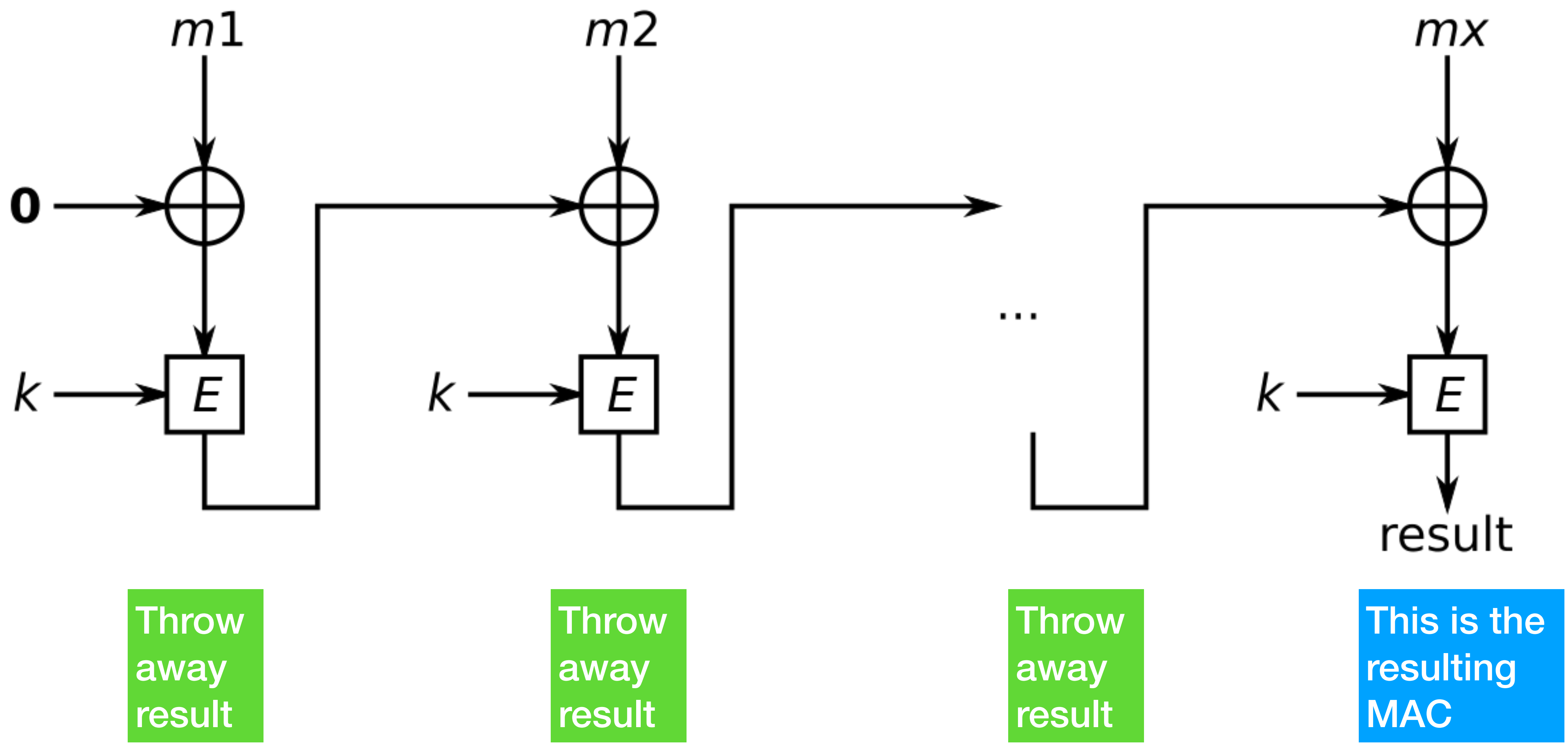
- HMAC is a type of MAC that uses a hash function as a building block.
- You can use any underlying cryptographically strong hash function.
- For example: HMAC-SHA256 uses SHA256.

Other MACs

- You can generate MACs in a different way as well.
- Example: CBC-MAC
  - Encrypt message with CBC,
  - use only last encrypted block.
  - that value is the MAC.

CBC-MAC

Original Message M = m1 || m2 || m3 || … || mx



Repudiation

Encryption End-points

- Let’s say Alice and Bob are communicating securely.
- What happens if Bob’s communication device is compromised?
- How can Alice protect herself?

Signatures vs Encryption

- A compromise of encryption key exposes previously encrypted messages.
  - Encrypt with different keys.
- A compromise of signing key does not invalidate past signatures.
  - Sign with long-term keys.
  - Tell your friends your long term ID public key.

Repudiation

- If Alice sends a message to Bob, she signs it.
- But that means that Alice can be blackmailed.
- Alice can’t deny that she even sent the message.
- Repudiation: ability to deny that you have sent the message.

Repudiation

- Therefore: we don’t want to sign messages directly with ID key.
    - Only sign Diffie-Hellman public key with ID key.
    - Encrypt with DH-derived session key.
    - We can also sign MACs.

MACs & Repudiation

- Alice sends a message to Bob
- She sends a MAC based on a MAC secret key.
- Bob knows the MAC secret key, and can reconstruct the MAC on his end, and compare.
- Bob can prove to himself that it was Alice.
  - But: Bob can't prove this to anyone else, because he may himself made up the MAC.

Combining Digital Signatures and MACs

- Digital Signatures authenticate a Diffie-Hellman exchange between Alice and Bob.   Sign(ID_A, aG)
- This would tell Bob that he got DH key from Alice, indeed.
- This establishes a DH shared secret.
- Now, we can use MACs to authenticate individual messages, because MACs offer the Repudiation property.
    - Because MACs are based on a symmetric key.

Elliptic Curves

Group Theory

- A set of elements is a group if they can be combined by some operation, and the result is still in the set.
- If the operation is "+", the notation is $C = A + B$
- If the operation is "x", the notation is $C = AB$

Repetition notation

- If the operation is “+”, then: $A + A + A = 3A$
- If the operation is “x”, then: $A \times A \times A = AAA = A^3$

“Null” element

- Note that the set must have an element called “identity” that doesn’t do anything.
  - A + “0” = A
  - A x “1” = A
- Note: a group based on “x” never has a 0 element.

Cycles in groups

- It could be that A + A + A + … + A = A
- Example: bits
  - 1+1+1 = 1

Cyclic group

- If every element of a group can be expressed in the form A + A + A + … + A, that means that the group has all the elements on a circle.

- Example: { 2, 2*2, 2*2*2, 2*2*2*2, … } mod 3

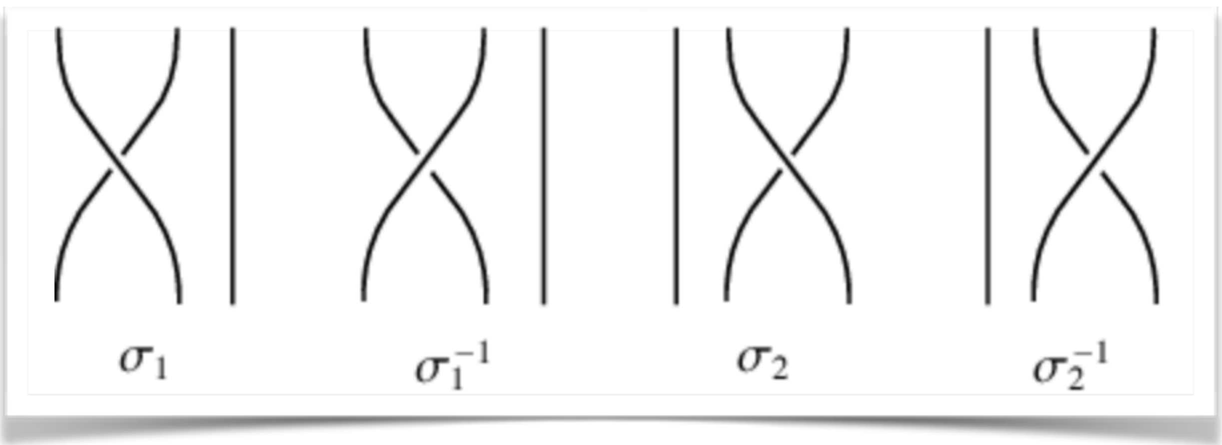
  - = { 2, 4, 8, 16, … }

  - = { 2, 1, 2, 1, … }

  - = { 1, 2 }

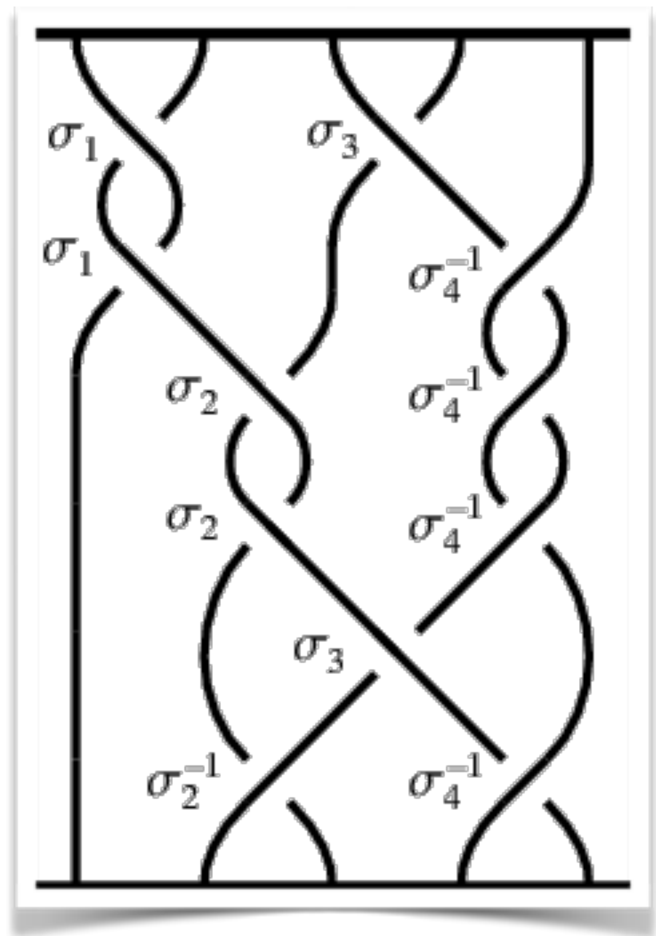
  - 2 is the generator of the group { 1, 2 } mod 3, under “x”.

Group Example: Braids



Basic braid patterns
can generate a complex
longer braid, when combined.

Combinig Braids

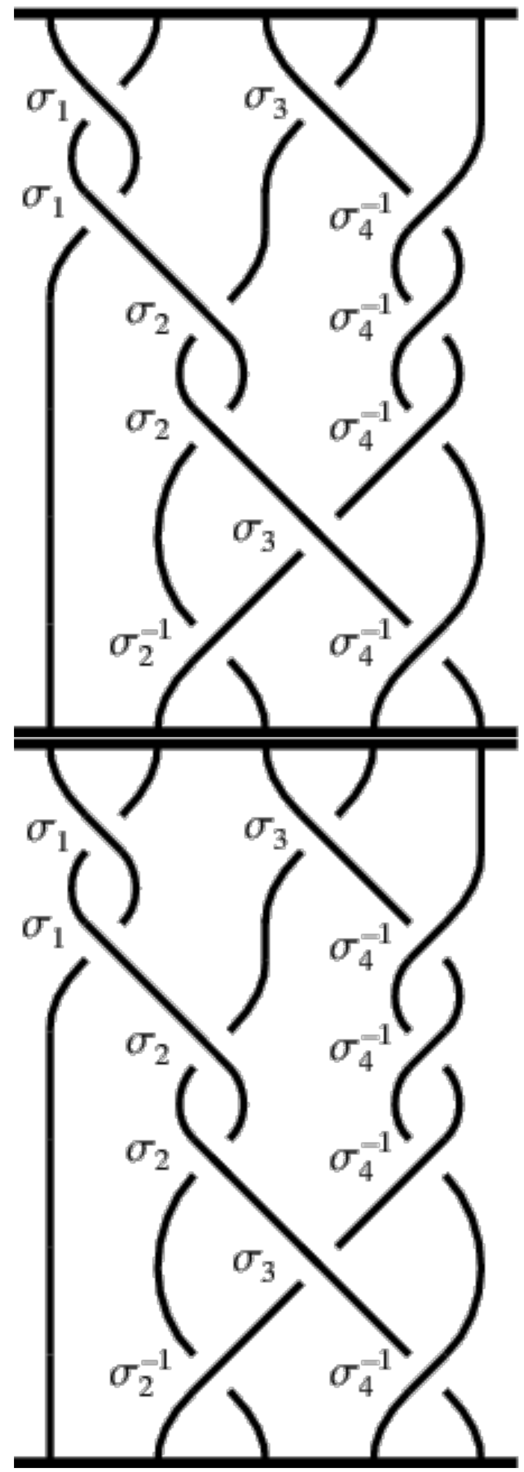


The group operation is concatenation, and it is represented by multiplication “x” operation.

This braid is uniquely identified by this expression:

$$\sigma_1 \sigma_3 \sigma_1 \sigma_4^{-1} \sigma_2 \sigma_4^{-1} \sigma_2 \sigma_4^{-1} \sigma_3 \sigma_2^{-1} \sigma_4^{-1}$$

“Multiply” two braids



To ponder

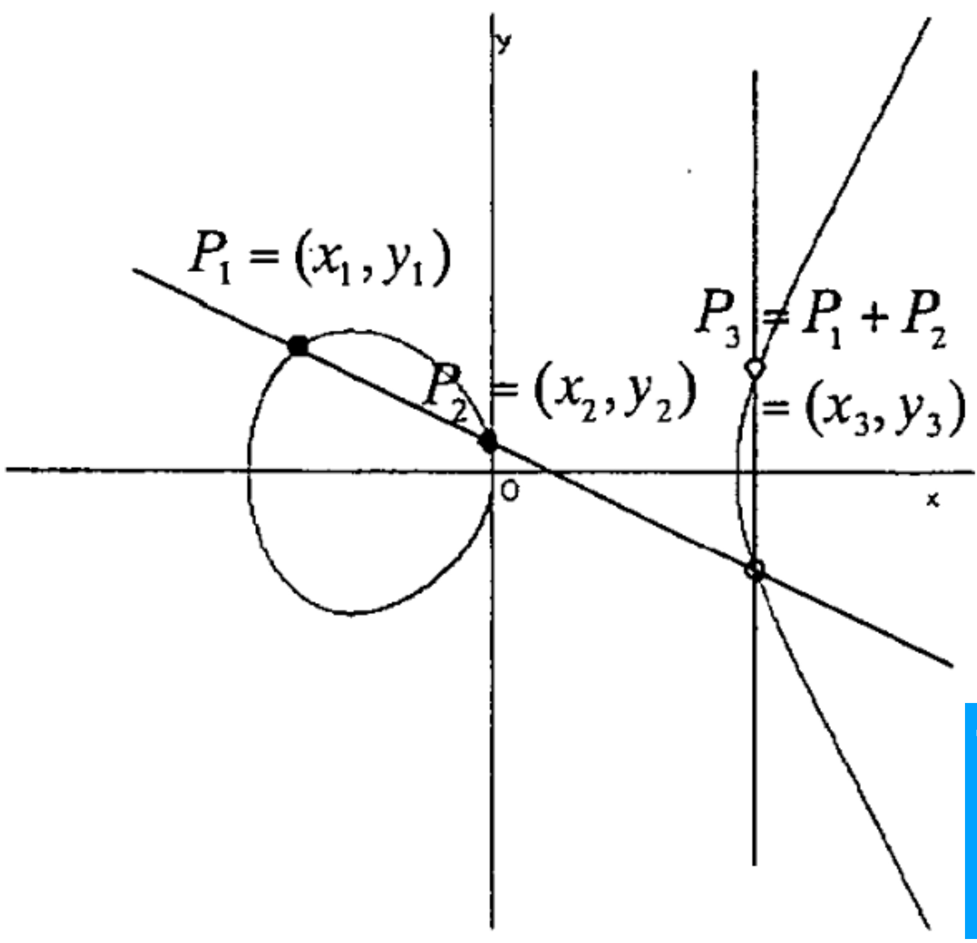
- Some concatenations may untangle braids.
- Some concatenations may tangle them more.
- Imagine: you are given a very tangled braid and you are asked to write down a formula for it in terms of generators.
- In cryptography, you are given a number, and asked to write a formula for it based on a basic element and exponent.

Elliptic Curve group

- There is an elliptic curve identified by a set of points (x,y)
- There is a generator "G" point on the curve.
- kG = G + G + … + G   // k times
- k is called "scalar".
- G is called "generator".
- What is "+" ?   It's not a normal addition operation.

(x1,y1) "+" (x2,y2)

$E: y^2 = x^3 + ax + b$ (ELLIPTIC CURVE OVER A PRIME FIELD)



$y^2 = x^3 - x$

ELLIPTIC CURVE ADDITION (ECADD)

$$x_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)^2 - x_1 - x_2$$

$$y_3 = \left(\frac{y_1 - y_2}{x_1 - x_2}\right)(x_1 - x_3) - y_1$$

The "+" operation mixes together x and y coordinates according to weird formula that arises from a geometric construction.

Curve25519

- based on prime field $\mathbf{F}_p$, with p = $2^{255} - 19$.
- Curve: $y^2 = x^3 + Ax^2 + x$ over $\mathbf{F}_p$

Generated points

- G is a base point on the curve
- { kG | for all integers k } is a finite set of points
- and it it is a group under the “+” operation.

Use in encryption

- Given k and G, it is easy to compute point kG
- But, given point G and kG it is hard to compute k.

Commutativity

- Note that (a*b) G = (b*a) G.

Private vs. public keys

- Make the scalar k to be the secret value.
- Make the product kG to be the public value.

Example:

- Alice has private key “a”
    - That means her public key is aG.
- Bob has private key “b”
    - That means his public key is bG.