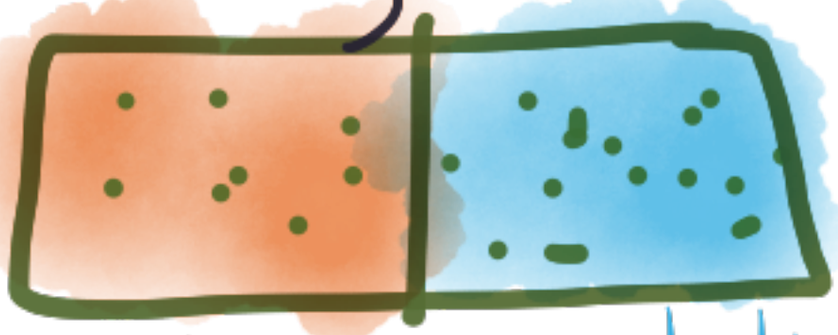
Generate a shared secret

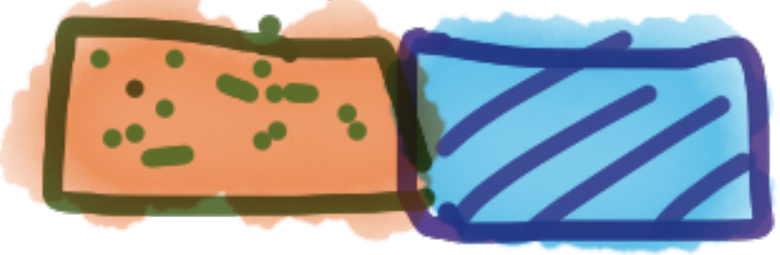
ALICE

BOB

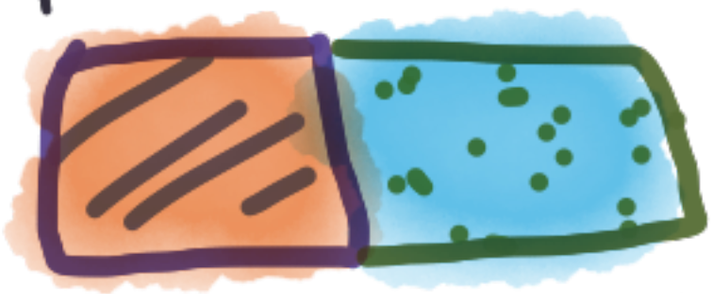
1. each generate a key with 2 parts



2. they share public parts only



same!



3. they now share a secret!

Because MATH.

Initial exchange uses 2 keys for each party with 2 parts

1. Alice generate 2 keys each with 2 parts

Bob

2. they exchange public parts

3. they now share a secret because: *MATH

Bob's Registration

Bob keeps private half of identity key. Private prekey half kept until use.

1 registers username (phone #)

2 sends identity and pre-keys (public) to server

Whisper Server

ALICE wants to send message to BOB

ALICE

requests prekey for BOB from server + Bob's identity Key !.

PK uploaded periodically by BOB

ALICE is ready to encrypt

see next page

ALICE WRAPS HER INITIAL MESSAGE

a) generates ephemeral

b) uses Bob's ephemeral/identity
to calculate **SHARED SECRET**
⇒ root of message chain

c) generates new sending key
ephemeral
⇒ new root key in sending chain key

d) generates **MESSAGE WRAPPER**
sends to BOB

(see next panel)

Creating a message (same for Alice or Bob)

1. Use existing sending chain key => message key / new sending chain key
2. message key => cipher key / mac key
3. encrypt message in AES256 CTR mode with ctr=# messages sent on chain. Append HMACSHA256(ciphertext) translated to 8 bytes to ciphertext
4. message = encrypted message + public ephemeral of sending chain

Bob receives prekey wrapped message from Alice

PROCESS WRAPPING

1. looks up prekey Alice used by id in wrapper

2. uses his identity key + Alice's identity and ephemerals to generate Shared Secret

⇒ new Root key, Chain key for sending Chain

3. Bob receives message wrapped can decrypt immediatly by going through receive pipeline (next page)

Receiving a MESSAGE

(same for Alice or Bob)

1. extract sender's identity, ephemeral key
2. retrieve my private ephemeral key on the sending chain and my identity key
3. ⇒ shared secret, new root key, new receiving chain key
4. derive message key ⇒ cipher, mac key, new receiving chain key from current receiving chain key.
5. AES256 to decrypt in CTR mode, verify 8 byte truncated MAC, update sending chain

---

6. generate new ephemeral keypair
7. combine w/ ephemeral just received ⇒ new root key, sending chain key

Alice's first send

server

3ECDH (A, A0 , B, B0)

gen

RK

gen A1, B0

CK, MK0 ⇒ cipher keys

CK, MK0 ⇒ ciphar, mac keys

Alice sends message encrypted under MK0 with counter=1) A1 to Bob wrapped in prekey message with A, A0 id of B

Legend

- private
- public
- identity
- Alice
- server
- Bob
- sending
- receiving

Bob's first receive

ECDH(A1, A0, B, B0)

RK

CK, MK0 (unused)

A1, B0

DECRYPT USING MK0

RK

gen B1, A1

RK

CK, MK0 → cipher, mac keys

CK, MK0 → cipher, mac keys

Sending Multiple Messages under same Chain
(send messages before reply)

Alice (Sending Chain)

CK-MK0

CK-MK1

CK-MK2

⋮

Bob (receiving chain)

CK-MK0

CK-MK1

CK-MK2

⋮

Where Next $MK = HMAC\text{-}SHA256(CK, 0x01)$

Next $CK = HMAC\text{-}SHA256(MK, 0x02)$

Derivations

MK ⇒ HMAC(**MK**, 0x0, "WhisperMessageKeys")

input key material · salt · info

new MK on chain

32 byte cipher key · 32 byte MAC key

RK ⇒ HMAC(ECDH(theirEphemeral, OurEphemeral), **RK**, "WhisperRatchet")

on receipt of new ephemeral

32 byte next **RK** · 32 byte sending **CK**

SS ⇒ HMAC(**SS**, 0 bytes, "WhisperText")

initial shared secret

32 byte first **RK** · 32 byte first sending **CK**