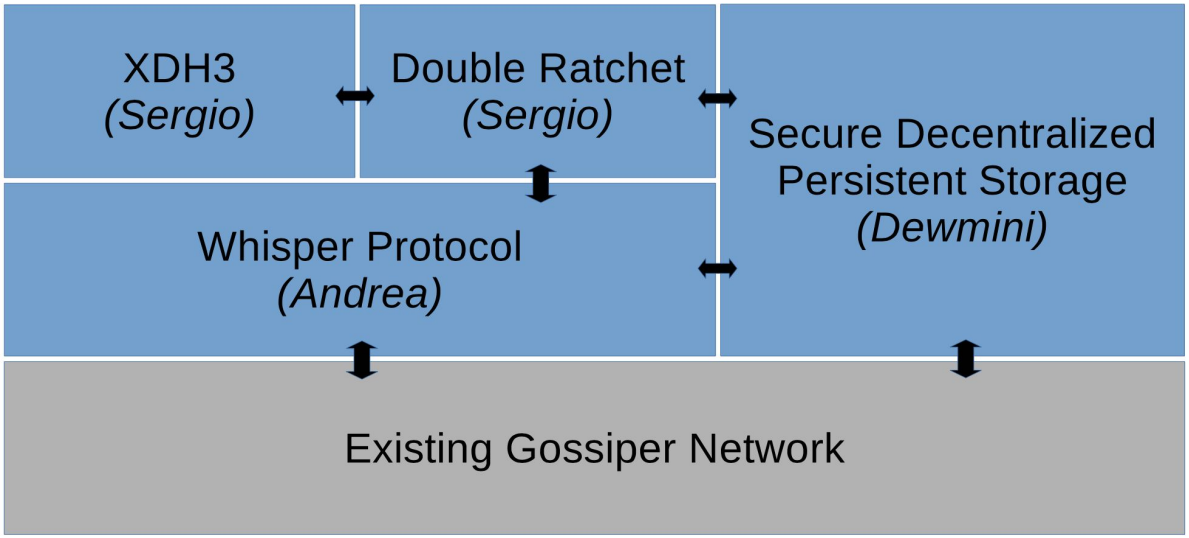
ÐSignal: A decentralized privacy-enhanced Signal protocol

Goal: enhance Peerster with security and privacy features and build a decentralized messaging protocol with state-of-the-art approaches

- **Signal protocol:** add confidentiality and security
- **Whisper protocol:** enhance privacy and metadata protection
- **Secure decentralized storage:** safely store messages, files and keys that are required by the above protocols in a decentralized network



Secure Decentralized Persistent Storage

Introduction

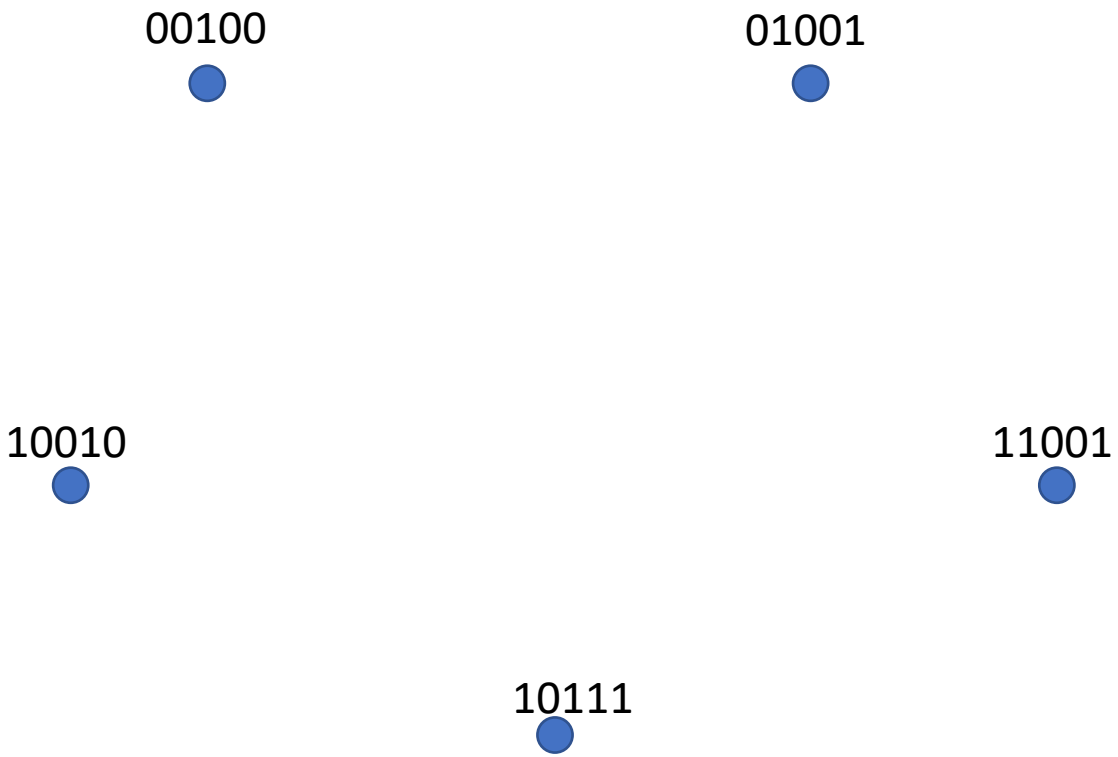
- Based on Kademlia
- Small chunks for efficient transportation
- Persistence through replication
- Standard file encryption before storing

Splitting File into Chunks

- Large files are split into small chunks
- Chunks are stored in DHT indexed by their hashes
- Hashes concatenated to create a metafile
- Metafile hash is the key for the whole file

Where To Store?

```
Hash: 01001
Data: … … …
```



Where To Store?

```
Hash: 01001
Data: … … …
```

$00100 \oplus 01011 = 01111$

$01001 \oplus 01011 = 00010$

$10010 \oplus 01011 = 11001$

$11001 \oplus 01011 = 10010$

$10111 \oplus 01011 = 11100$

Where To Store?

```
Hash: 01001
Data: … … …
```

00100 ⊕ 01011 = **01111**

01001 ⊕ 01011 = **00010**

10010 ⊕ 01011 = 11001

11001 ⊕ 01011 = **10010**

10111 ⊕ 01011 = 11100

Routing Table

- For each prefix of a Node's ID, there is a bucket.
- Bucket ***i*** stores up to k other Nodes whose IDs are equal to the Node's ID up to the ***i***-th bit and are different from Node's ID in ***(i+1)***-th bit.
- Table is updated each time a message is received.
- Each bucket employs LRU eviction policy. But least recently seen entry is only evicted if it is not alive. Use Ping to check.

Finding k Closest Nodes

- Given an ID, create a short list of k closest nodes in the node's routing table.
- Ask some nodes in the short list to find the k closest nodes in their routing tables.
- Add the nodes in the responses to the short list.
- Repeat the process until the k closest nodes in the short list do not change.

Challenges and Solutions

- Handling arbitrarily large files through multiple levels of meta hashing.

```
Type of chunk | Type of hash function | Hash of the chunk
(1 byte)      | (1 byte)               | (28 to 64 bytes)
```

- Handling multiple parallel RPCs using dedicated response channels and unique RPC identifiers.

Achieving Darkness: Whisper

A decentralized privacy-preserving messaging protocol

Andrea Piccione

What is Whisper?

- Part of the P2P Ethereum suite
- Can be considered an Hybrid DHT and messaging system
- Designed to guarantee “darkness”, i.e. metadata protection and plausible deniability at high cost of bandwidth and computational costs (configurable efficiency-privacy tradeoff)
- Ensures sender and receiver anonymity by forwarding all messages

Why Whisper for messaging?

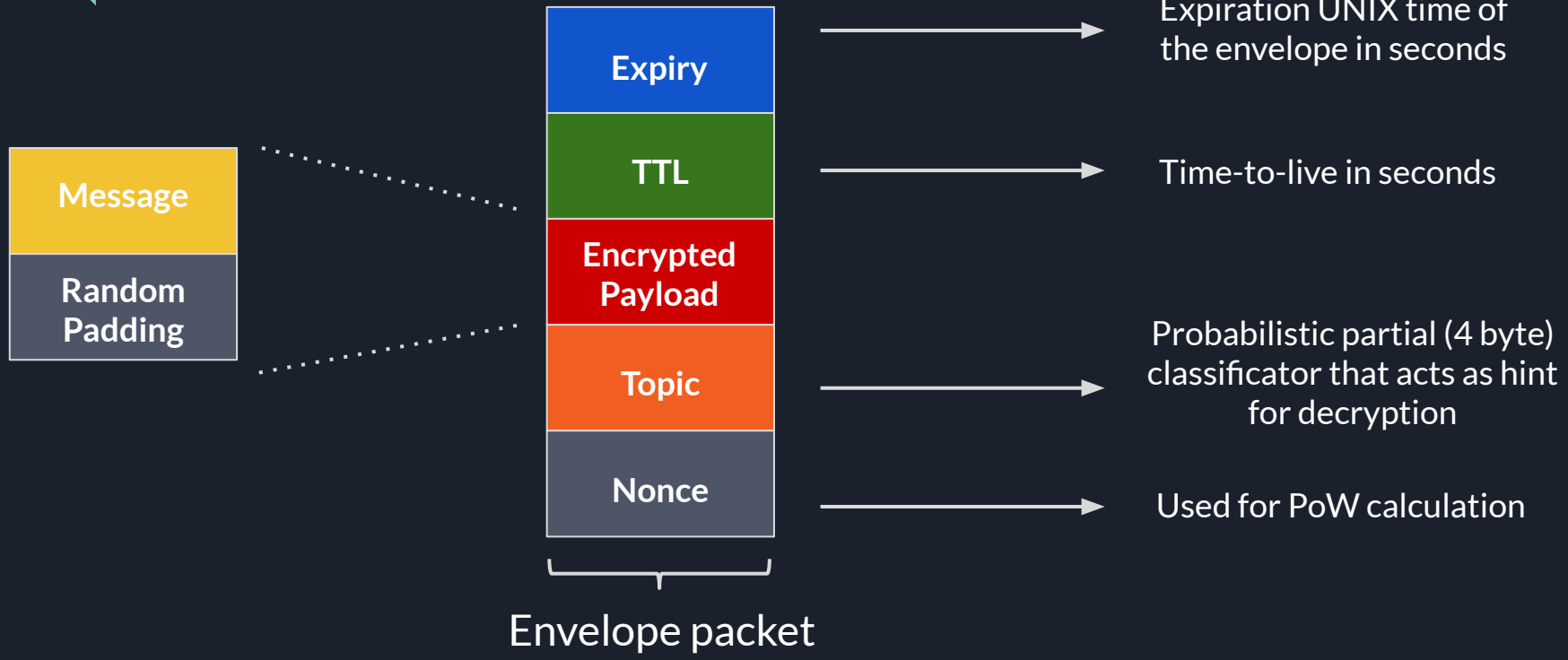
- **Goal**: decentralized private messaging (currently only status.im)
- Protect communication from third parties and meta-information leakage (all major messaging apps don't have such a feature)
- Naturally extend Peerster with security and privacy capabilities
- Built as a library on top of Homework 1 features (Gossip protocol)

Whisper main features

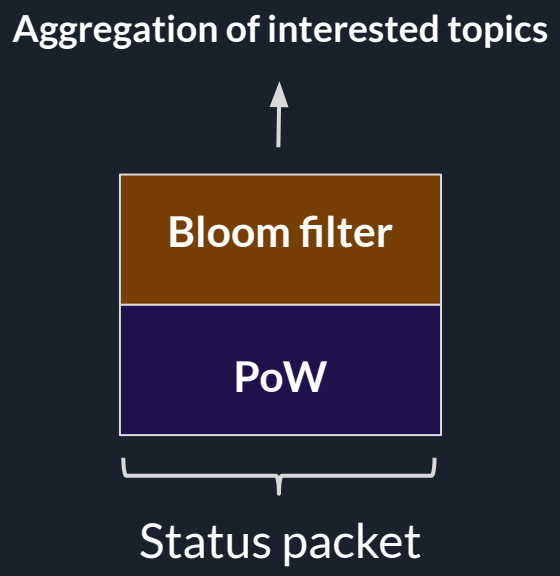
Implemented Whisper protocol version 6 with very few changes. The main aspects of the protocol are:

- end-to-end encryption support, both symmetric (AES-GCM) and asymmetric (ECIES with SECP-256k1) encryption
- probabilistic message forwarding (topic-based bloom filters)
- incentive compatibility for peers to punish malicious nodes (blacklisting and DoS prevention with Proof-of-Work)

Architecture: envelope packet

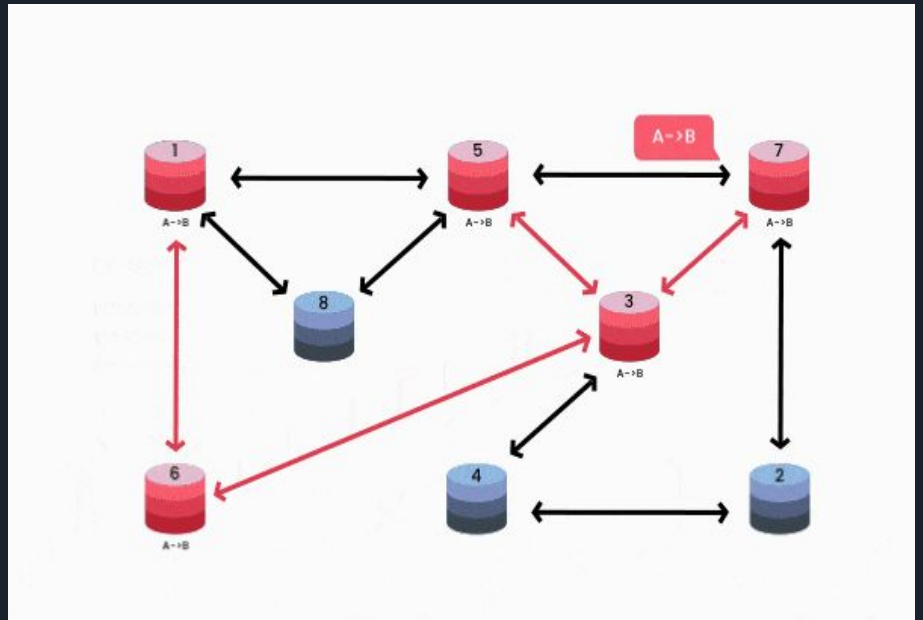


Architecture: status packet



- Nodes use the underlying rumor-mongering protocol to gossip their status at start, upon any change and periodically
- Nodes decide how “dark” they want to be by revealing a more or less accurate bloom filter
- Nodes keep track of all the requirements they receive from each peer in a routing table
- Peers only forward messages to you that match the aggregated filter, violators are blacklisted

Architecture: message forwarding



Source: https://medium.com/caelumlabs/whisper-shh-bc5416ec0046

- Nodes spend PoW to send a message (spam prevention)
- Messages are forwarded periodically and cached until their TTL expire
- Nodes notify the client upon receiving a message that matches any of the installed filters AND can be decrypted
- Peers generally can't tell if a neighbour peer is the message originator

Architecture: API

- **NewWhisperMessage(NewMessage)**: create a message with arbitrary payload and topic and other parameters
- **NewMessageFilter(FilterOptions)**: create a new filter (subscriber) with desired topics for incoming messages and other options
- **GetFilterMessages(string)**: retrieve all messages filtered by the filter with specified ID from the last time the method was called
- **Identity management**: create, add, delete symmetric/private/public keys
- Setting and modifying Whisper **parameters** directly (PoW required, BloomFilter, MaxMessageSize, timeouts etc.)

Main challenges in development

Make up for the absence of underlying ÐΞV-p2p ethereum backend	• Provides ability to “rate” and steer the set of peers according to the utility delivered • Overcomed with ad-hoc blacklisting and routing table forwarding
Efficiency-privacy tradeoff	• Broadcast all messages or filter them only based on the next hop was not feasible • Improved the protocol with aggregated-status routing and status gossiping

Achieving Security: Signal

A secure end-to-end messaging protocol

Sergio Roldán

What is Signal?

- Signal is a non-federated cryptographic protocol that can be used to provide end-to-end encryption to instant messaging conversations

- The protocol combines Double Ratchet algorithm, prekeys, a triple Elliptic Curve DH (X3DH) handshake, AES in CBC mode and HMAC

- The protocol provides confidentiality, integrity, participant consistency, forward secrecy, future secrecy...

Why use Signal?

- Goal: lifelessness secure messaging (used in , , ...)
- Protect the communication against passive and active attacks and provide both future and forward secrecy in case of compromised keys
- Naturally extend Peerster with a strong security capability
- Built as a library on top of Homework 1 features (Gossip protocol). We rely on the network to distribute and store messages and keys

Signal main features

Implemented Signal as in signal.org/docs/ with slight modifications:

- Extended Triple Diffie-Hellman using the NIST P-256 Elliptic Curve, ECDSA, SHA-256 and HKDF
- Double Ratchet without header encryption using the same Curve, HKDF + plain HMAC, SHA-256 and AEAD encryption using AES-256 in CBC mode with HMAC
- Sesame was out of the scope of this project

Extended Triple DH

Bob (B)

Alice (A)

IK = Identity Key Pair - Master
SPK + Sign = Signed PreKey
OPK = One-time PreKey

Generate: IK, SPK, Sign, OPK

X3DHIdentity

Compute:

- $DH1 = DH(IK_A, SPK_B)$
- $DH2 = DH(EK_A, IK_B)$
- $DH3 = DH(EK_A, SPK_B)$
- $DH4 = DH(EK_A, OPK_B)$

$SK = KDF(DH1, DH2, DH3, DH4)$
$AD = IK_A \| IK_B$

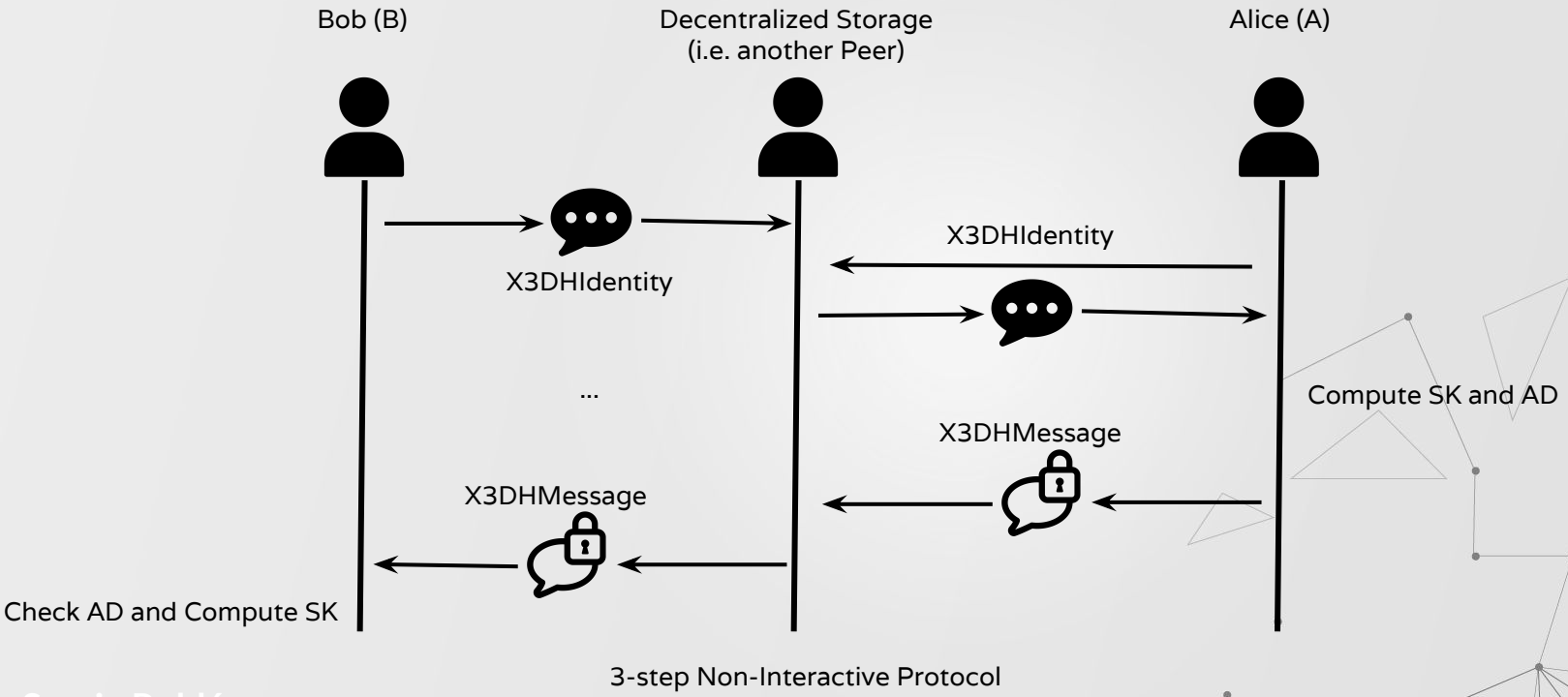
X3DHMessage

Check AD and Compute SK

3-step Interactive Protocol

Sergio Roldán

Extended Triple DH



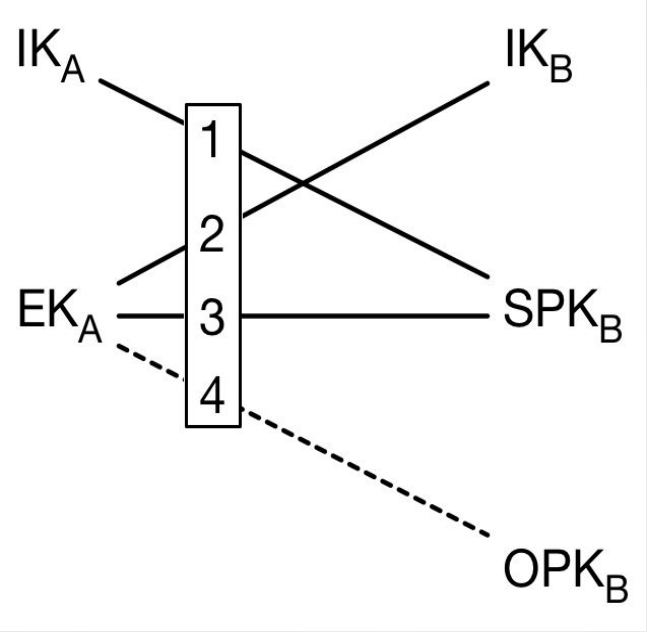
3-step Non-Interactive Protocol

Sergio Roldán

Extended Triple DH

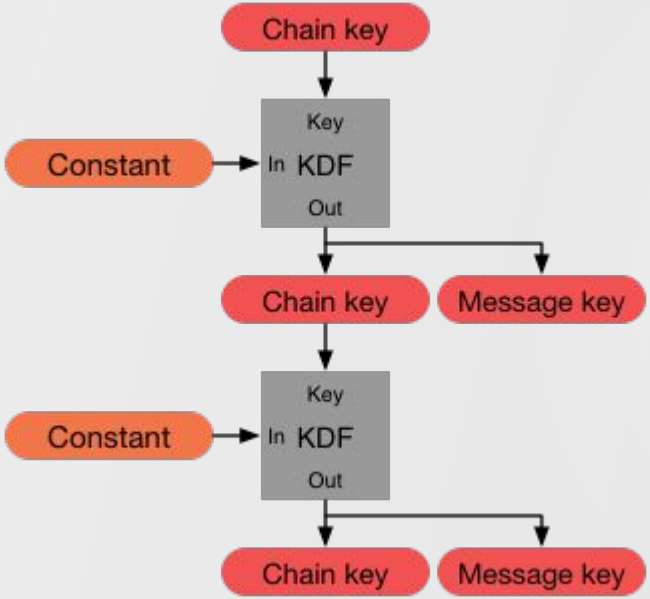
- X3DH provides forward secrecy and cryptographic deniability
- Supports asynchronous configuration

- DH1 and DH2 provide mutual authentication
- DH3 and DH4 provide forward secrecy
- SPK is updated periodically and OPK is used only once

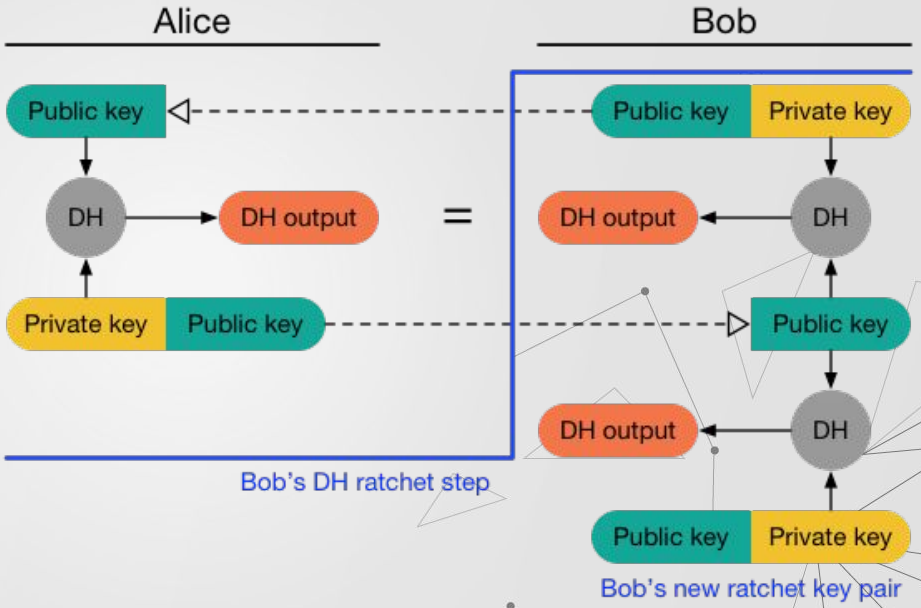


https://signal.org/docs/specifications/x3dh/

Double Ratchet



Symmetric Ratchet*

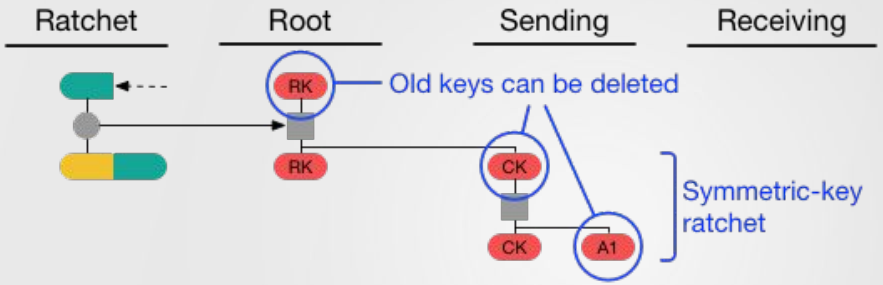


Diffie-Hellman Ratchet*

Sergio Roldán

*https://signal.org/docs/specifications/doubleratchet

Double Ratchet

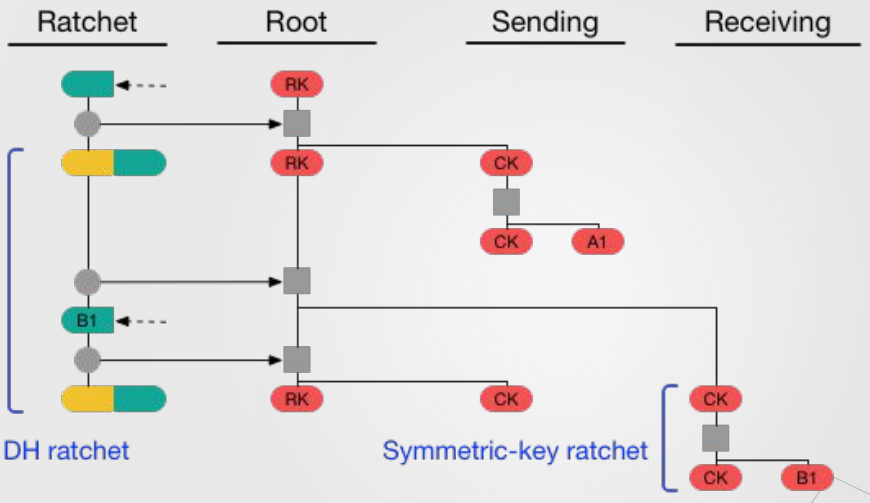


Double Ratchet Step 1*

Sergio Roldán

*https://signal.org/docs/specifications/doubleratchet

Double Ratchet



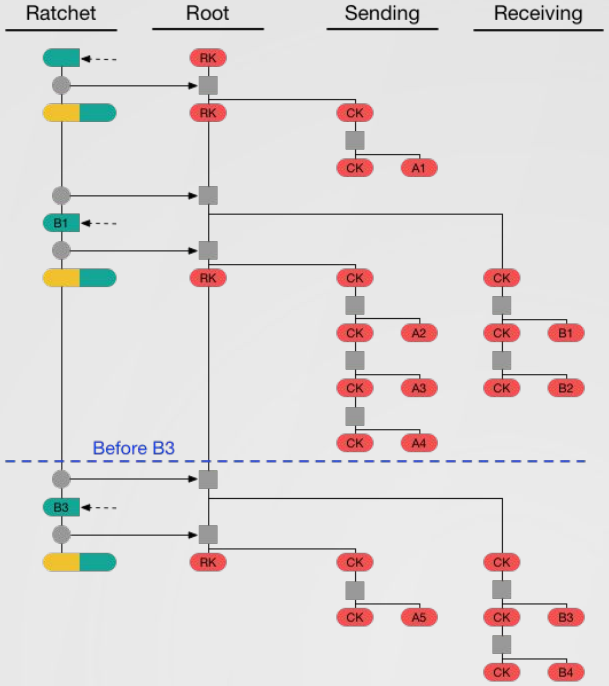
Double Ratchet Step 2*

Sergio Roldán

*https://signal.org/docs/specifications/doubleratchet

Double Ratchet

- Combines DH Ratchet with Symmetric Ratchet to provide forward and future secrecy
- Messages are protected using AEAD deriving keys from the Ratchet
- All keys should be deleted once the Ratched as derived them



*https://signal.org/docs/specifications/doubleratchet

Signal library

The library includes:

- adsAEAD: AEAD encryption/decryption
- adsDoubleRatchet: Double Ratchet Protocol
- adsX3DH: Extended Triple DH
- adsElliptic: Helper for operation over Elliptic Curves
- adsKDF: HMAC and HKDF KDF operations
- adsECDSA: Signature helper
- adsECDH: Diffie-Hellman operations
- signalhandler: X3DH protocol and Double Ratchet message exchange

Main development challenge: Integrate the different protocols with diverse data structures