Mercury Chat Service

Fabio Colacio

The Problem

Alice wants to send confidential messages to Bob without any third party, Eve, being able to read them.

The Solution

Mercury (server)

- Allows users to share messages

Quicksilver (client)

- Encrypts & passes messages to Mercury
- Receives & decrypts messages from Mercury
- Manages keys

Assets & Stakeholders

- Message contents - Users
- User passwords - Users
- Service availability - Users & Server Admins

Assumptions & Adversaries

- All traffic to and from the server is TLS encrypted
- Outsiders are capable of sending arbitrary HTTP requests in high volume
- Server admins are capable of reading/writing to the database

Messaging Protocol

Messages Table (Server-Side)

```
sender INT,
recipient INT,
timesent TIMESTAMP,
message BLOB
```

Encryption With EECDH (Sending a Message)

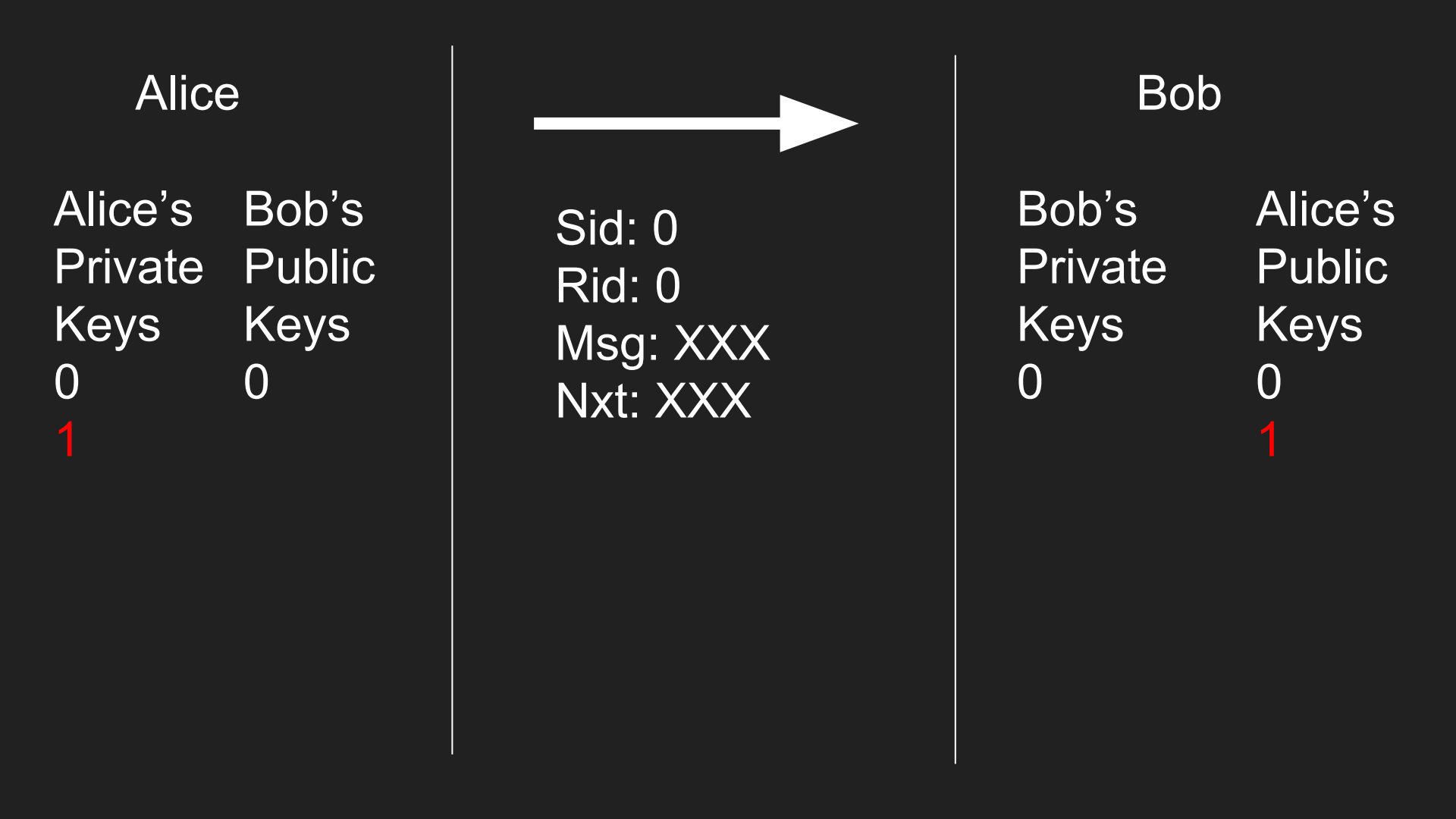
1. Secret = ECDH(myPrivate, yourPublic)
2. Key = KDF(Secret)
3. Cipher = Encrypt(Key, Message)
4. newPrivate, newPublic = NewEllipticKeys()
5. Send (Cipher, newPublic) to peer
6. myPrivate = newPrivate

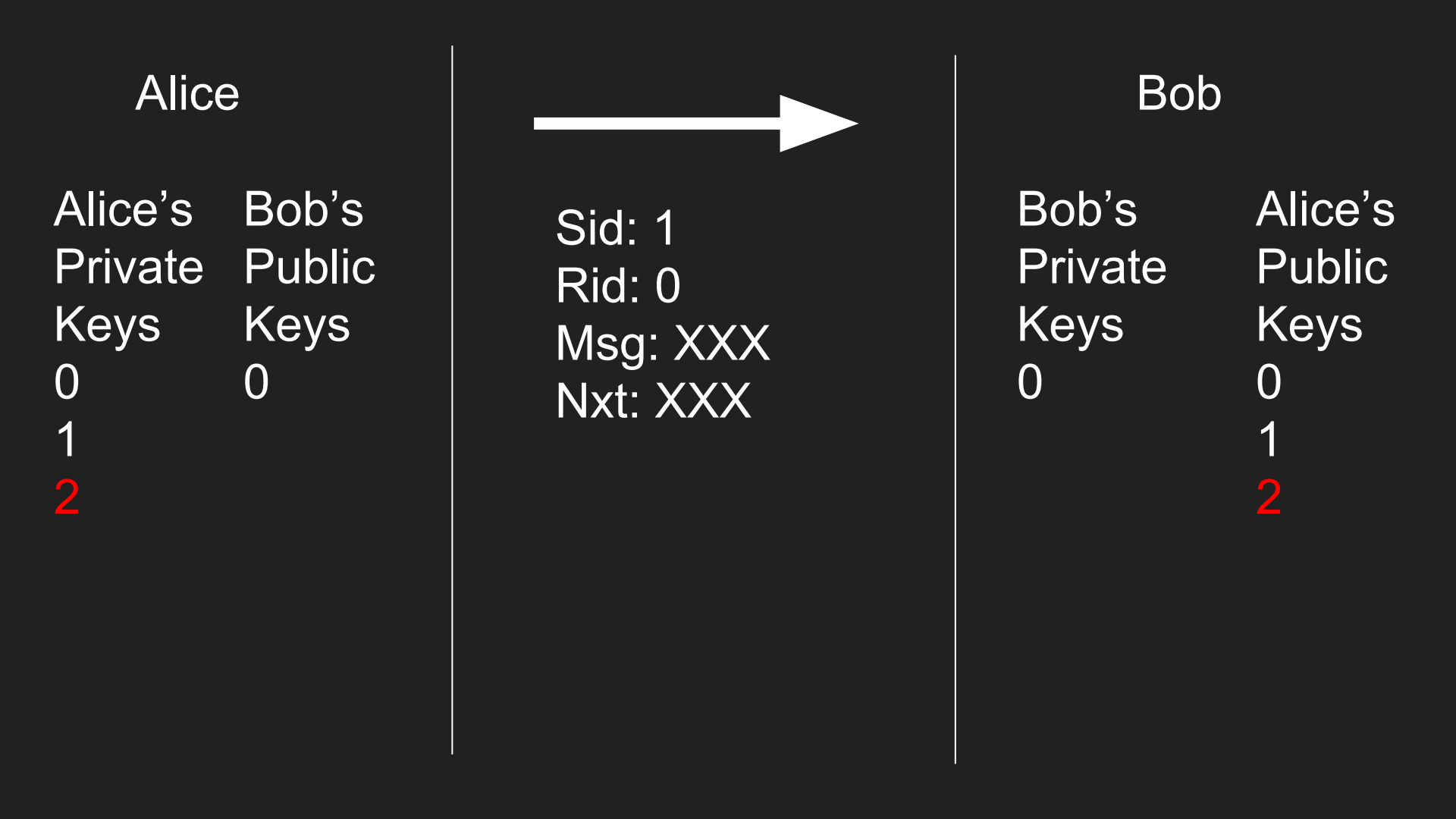
Decryption With EECDH (Fetching a Message)

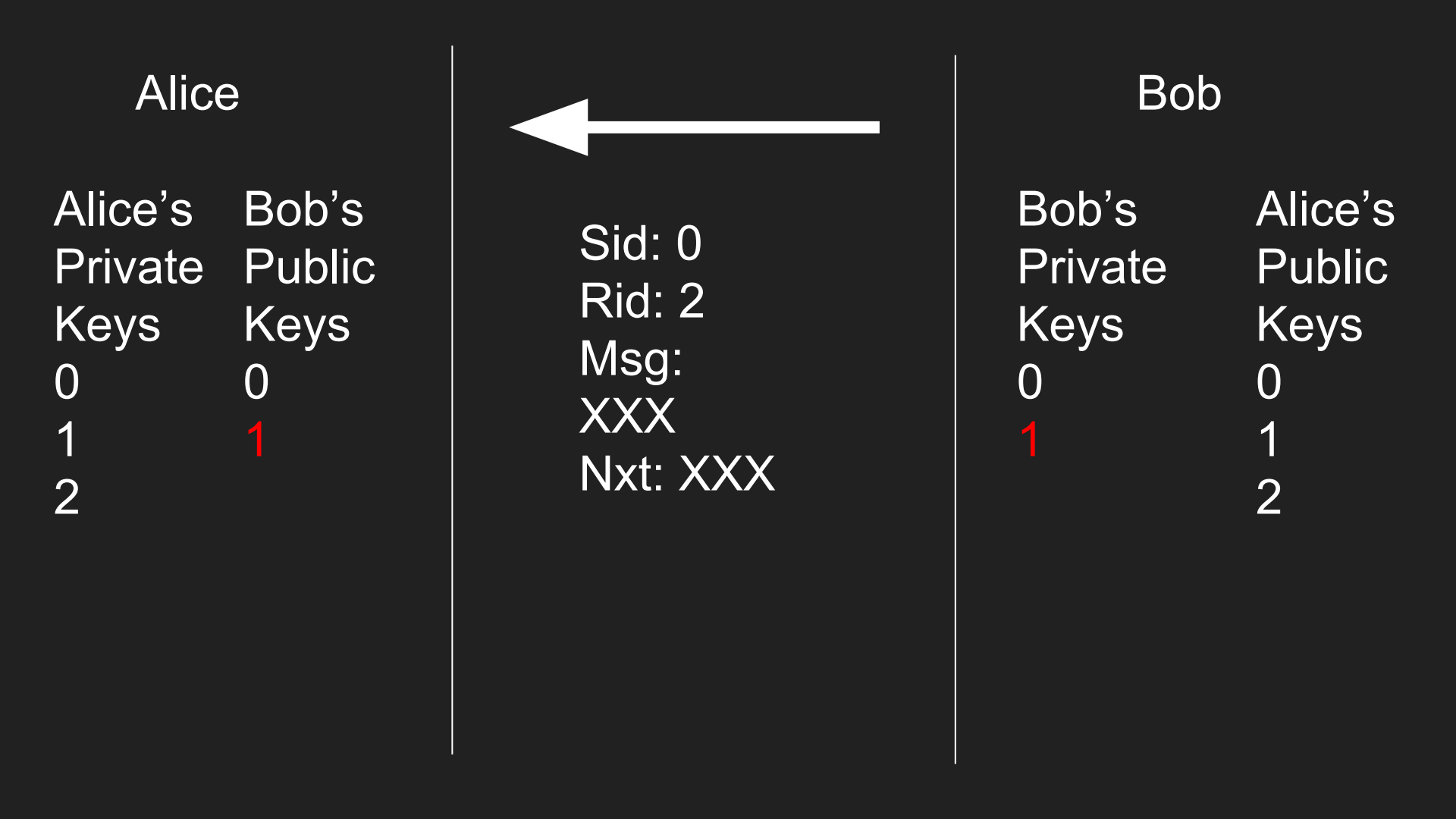
1. Get (Cipher, newPublic) from peer
2. Secret = ECDH(myPrivate, yourPublic)
3. Key = KDF(Secret)
4. Message = Decrypt(Key, Cipher)
5. yourPublic = newPublic

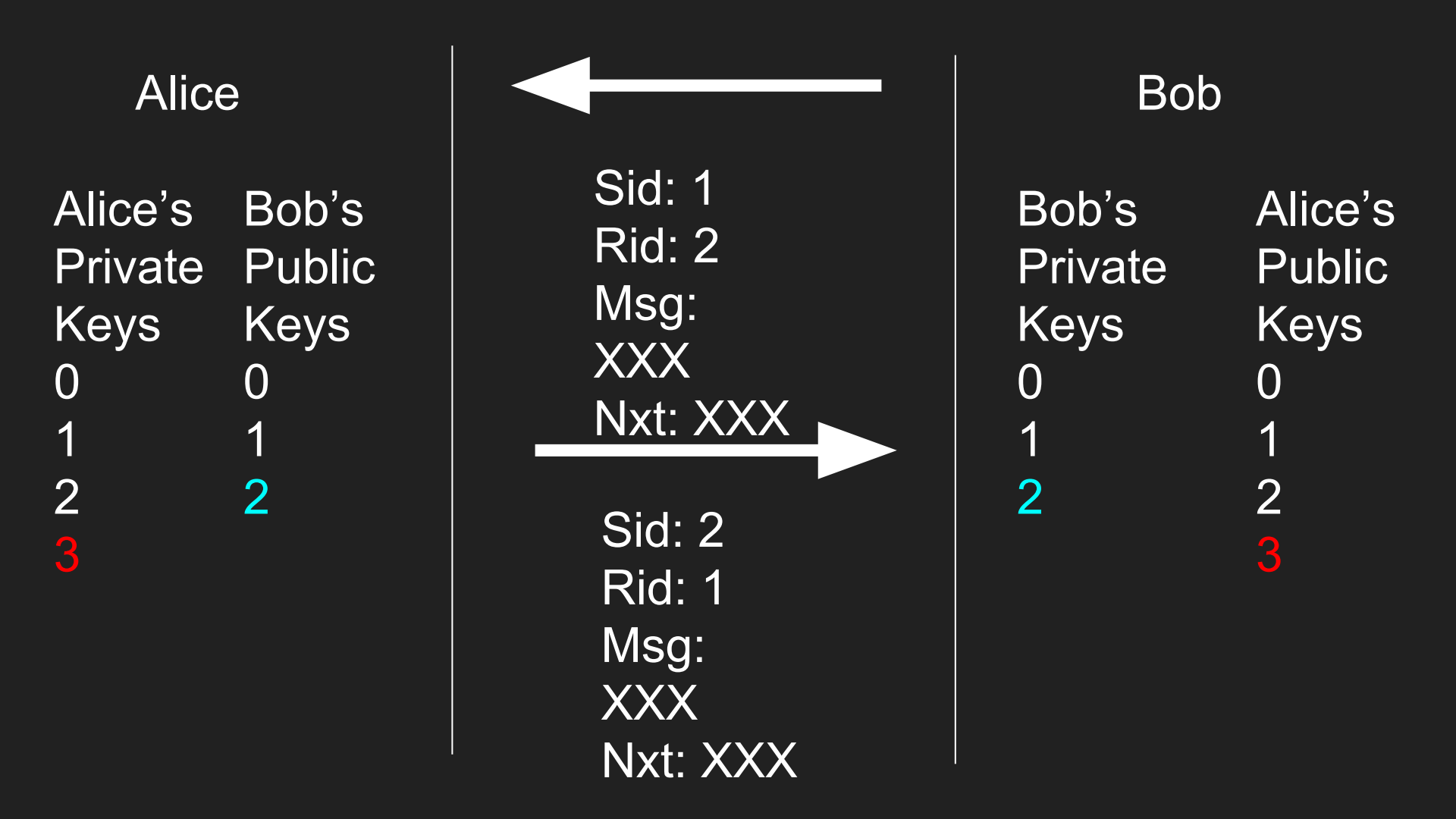
Message Structure

- **Sid** The id of the sender's key that was used to encrypt this message
- **Rid** The id of the receiver's key that was used to encrypt this message
- **Nxt** The next public key to use (encrypted)
- **IV** The AES initialization vector used for encryption
- **Msg** The encrypted message
- **Key** Encrypted HMAC key
- **Tag** HMAC integrity tag
    - HMAC(Nxt | Msg, Decrypt(Key))


Alice
Alice's Private Keys
0
1
Bob's Public Keys
0
Sid: 0
Rid: 0
Msg: XXX
Nxt: XXX
Bob
Bob's Private Keys
0
Alice's Public Keys
0
1


Alice
Alice's Private Keys
0
1
2
Bob's Public Keys
0
Sid: 1
Rid: 0
Msg: XXX
Nxt: XXX
Bob
Bob's Private Keys
0
Alice's Public Keys
0
1
2


Alice
Alice's Private Keys
0
1
2
Bob's Public Keys
0
1
Sid: 0
Rid: 2
Msg:
XXX
Nxt: XXX
Bob
Bob's Private Keys
0
1
Alice's Public Keys
0
1
2


Alice
Alice's Private Keys
0
1
2
3
Bob's Public Keys
0
1
2
Sid: 1
Rid: 2
Msg:
XXX
Nxt: XXX
Sid: 2
Rid: 1
Msg:
XXX
Nxt: XXX
Bob
Bob's Private Keys
0
1
2
Alice's Public Keys
0
1
2
3

Shortcomings

Lost Messages Prevent Communication

If one message gets lost, the conversation cannot continue because the receiver does not have the sender’s newest key, and cannot decrypt future messages.

Timestamp Manipulation

- Message timestamps are created by the server
- Server admins can manipulate timestamps so messages have the wrong times or appear in the wrong order

Exchanging Public Key Files Sucks

- Users are likely to use some insecure channel such as email to exchange initial keys
- It is inconvenient and cumbersome
- A mobile version of Quicksilver could fix this by using QR codes to exchange keys

Users Limited to One Device Per Conversation

- Keys are stored locally