A look into the Mobile Messaging Black Box

33rd Chaos Commmunication Congress #33c3

Roland Schilling   `@NerdingByDoing`

Frieder Steinmetz   `@twillnix`
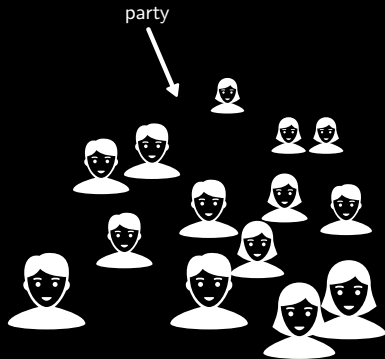
December 22, 2016

Hamburg University of Technology
Security in Distributed Applications

You're at a party

- Friend approaches you and needs to tell you something in private
- What do you expect when you say private?
- You enter a separate room, you trust the location
- What does a separate room offer you?

party
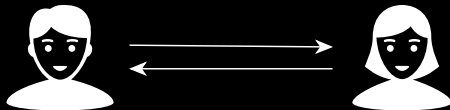
You are now alone in a closed room with your Friend

- Both of you have absolute Confidentiality that you are alone
- Nobody can overhear your talk
- Your exchange is completely private

We call this confidentiality

Since you're long-time friends, you're absolutely sure, whom you're talking to

- Nobody can impersonate your friend or you, without the other noticing
- You're talking directly, without a phone or webcam in between

We call this authenticity

The room you're in is small enough that you can always see each other

- You know that the words you speak are received just as you spoke them
- There is no way either of you hears something other than the other says

We call this integrity

Suppose somebody steps into the room

- They could overhear your conversation
- They would only learn the contents of this particular conversation
- They would not learn anything about past conversations you had

We call this forward secrecy

$\rightarrow$ After leaving they would not be able to listen to any future conversations you might have

We call this future secrecy

Perfect Forward- and Future Secrecy
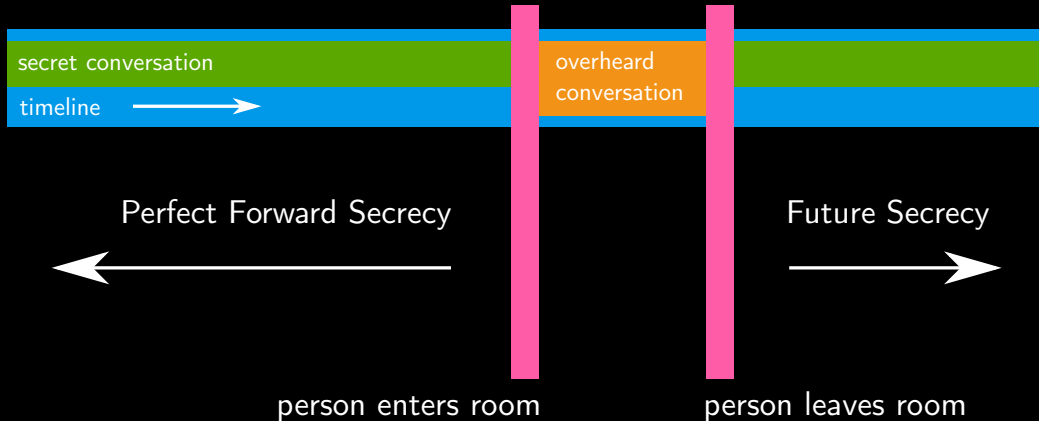
There are no witnesses in the room

- Either of you can later deny to other having made any statement
- Neither of you can prove to other that any of you have made a particular statement

We call this deniability

We started with a conversation analogy to identify our expectations of messaging

$\rightarrow$ Actually postal services are better to look at messaging from a technical point of view.
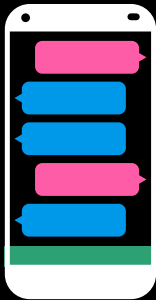
What if our party conversation had taken place via SMS?

Your providers (and other people on the same network)

- would know the contents of your exchange: no confidentiality
- could change the contents of your exchange: no integrity
- could reroute your messages and impersonate either of you: no authentication
- would know all messages you ever exchanged: no forward Secrecy
- would know all messages exchanged in the future: no future secrecy
- could store all messages and use them as proof of the exchange: no deniability

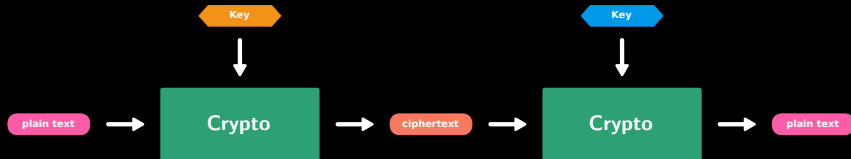$\rightarrow$ Messaging translates badly to our offline communication expectation

Symmetric Encryption:
→ Encryption and decryption with the same key

Symmetric Encryption:

$\rightarrow$ Encryption and decryption with the same key
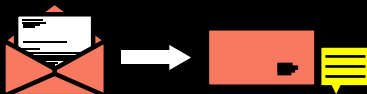
The Shortest Introduction to Encryption You Will Ever Get

Symmetric Encryption:

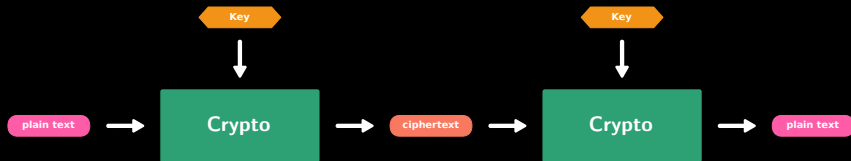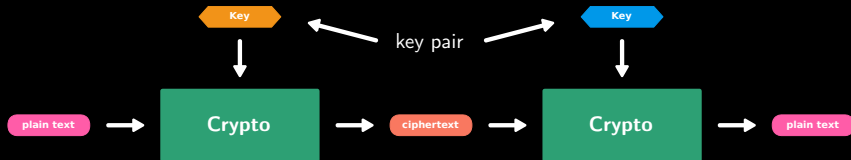$\rightarrow$ Encryption and decryption with the same key

Asymmetric Encryption:

$\rightarrow$ Encryption and decryption with different keys

Symmetric Encryption:
→ Encryption and decryption with the same key

Asymmetric Encryption:
→ Encryption and decryption with different keys

Public-Key Cryptography – In a Nutshell

| Secret Key | Public Key | Identity |

| Secret Key | Public Key | Identity |

- Both parties publish their identities and public keys
- Any message can be encrypted with anyone's public key and only be decrypted with its corresponding secret key
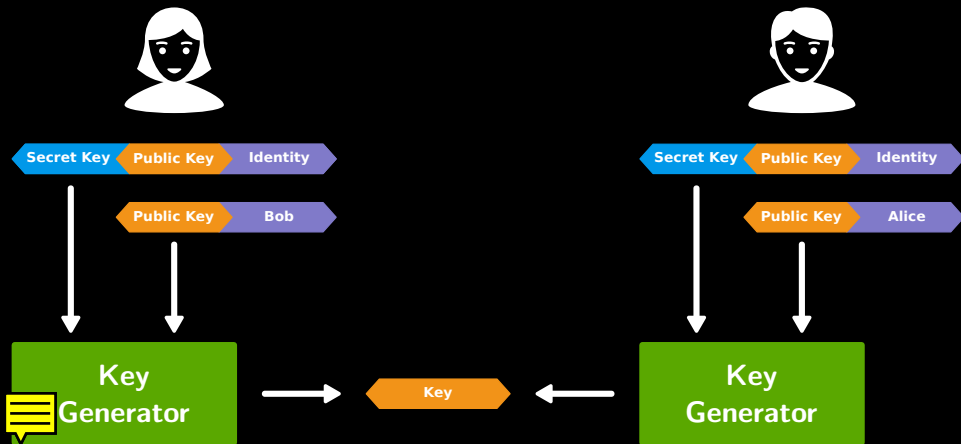
- Both parties publish their identities and public keys
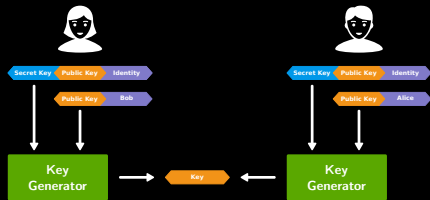- Any message can be encrypted with anyone's public key and only be decrypted with its corresponding secret key

- Both parties publish their identities and public keys
- Any message can be encrypted with anyone's public key and only be decrypted with its corresponding secret key

Asymmetric Encryption gives us IDs but is very expensive.
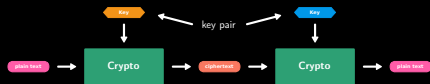
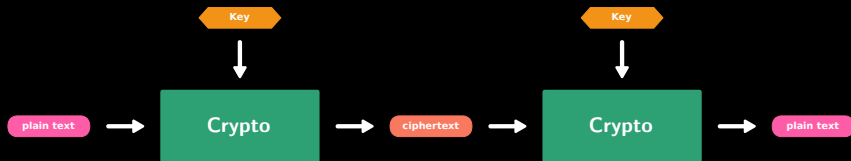Symmetric Encryption is cheap, but a key has to be shared by all participants before communication starts.

Authenticated Encryption allows us to create symmetric keys based on asymmetric key pairs.

But there's more…

Recap

Authenticated Encryption gives us:

- Confidentiality
- Deniability
- Authenticity

We don't have:
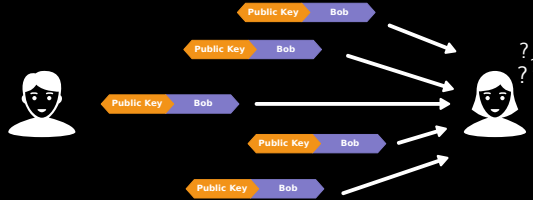
- Perfect Forward Secrecy
- Future Secrecy

$\rightarrow$ We are ignoring Integrity here, but we have that, too.

Cryptography is rarely, if ever, the solution to a security problem. Cryptography is a translation mechanism, usually converting a communications security problem into a key management problem.
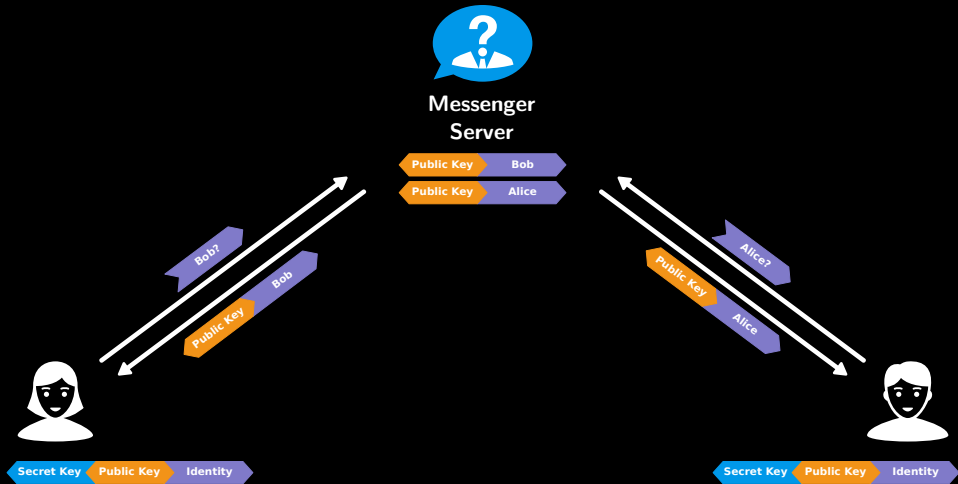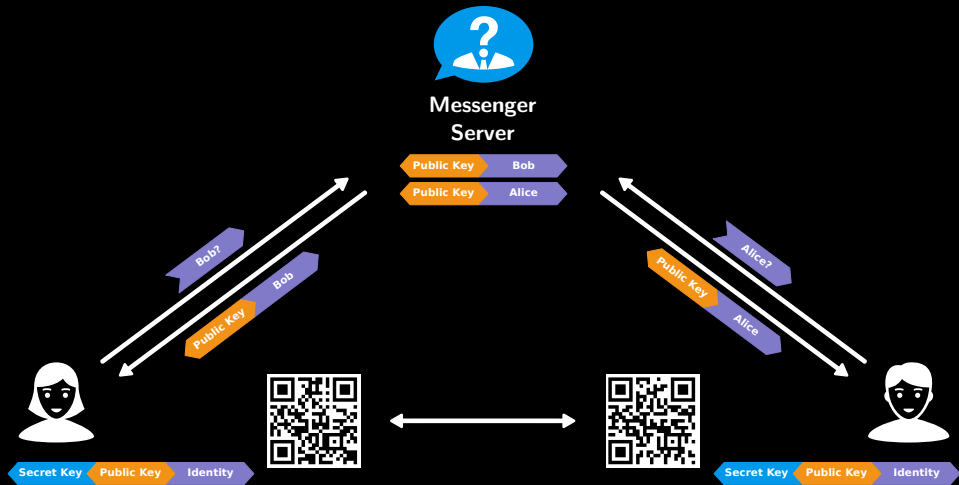
—Dieter Gollmann

How does Alice know which is Bob's public key?

- How to connect a key to a person?
  - → Key signing (PGP)
  - → Certificates (trusted third party)
  - → (Messenger) service-based directory (based on phone numbers or email addresses)
- How to deal with changing keys?
  - warnings are annoying
  - Threema's traffic light system encourages authentication but doesn't deal with changing keys (other than new identities for known phone numbers with yellow dots)

Everybody on the network can see:

- the sender of the message
- the intended receiver of the message
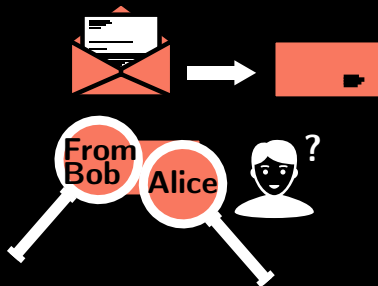
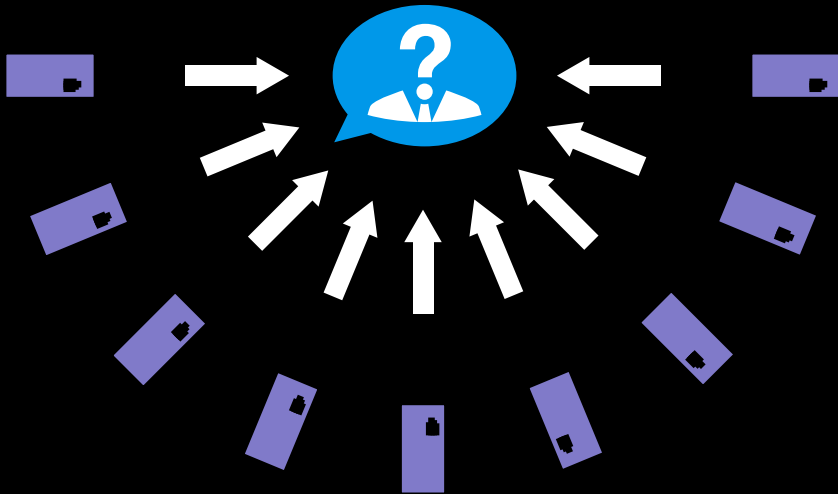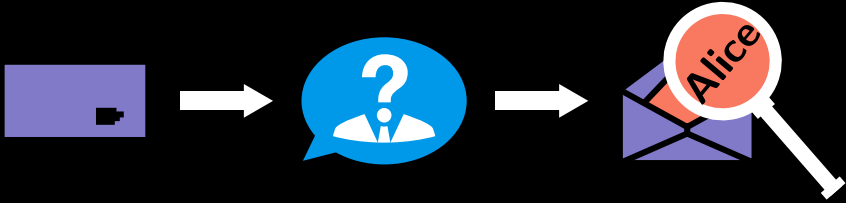Solution: wrap encrypted message in a second layer of encryption and address it only to the message server.

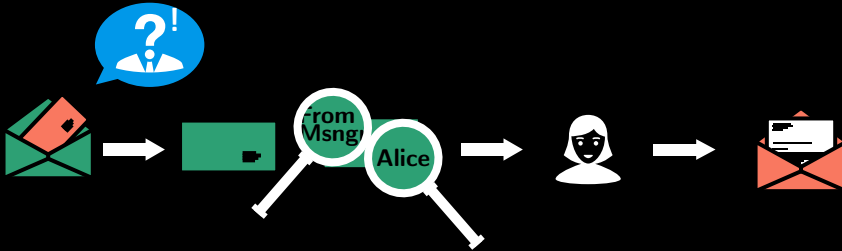The message server will remove the outer layer and add a new one, targeted at the receiver.

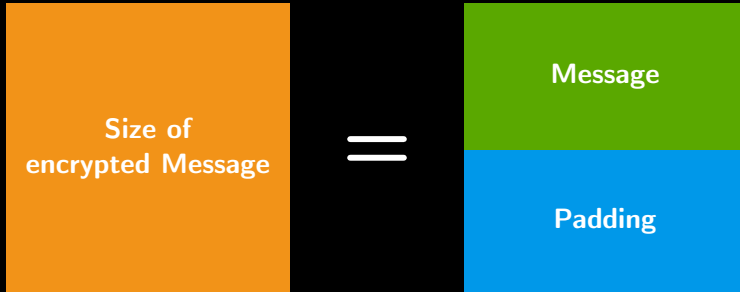This leaves us with an encrypted end-to-end tunnel, transmitted through two transport layer encryption tunnels.

The message server still knows both communication partners!

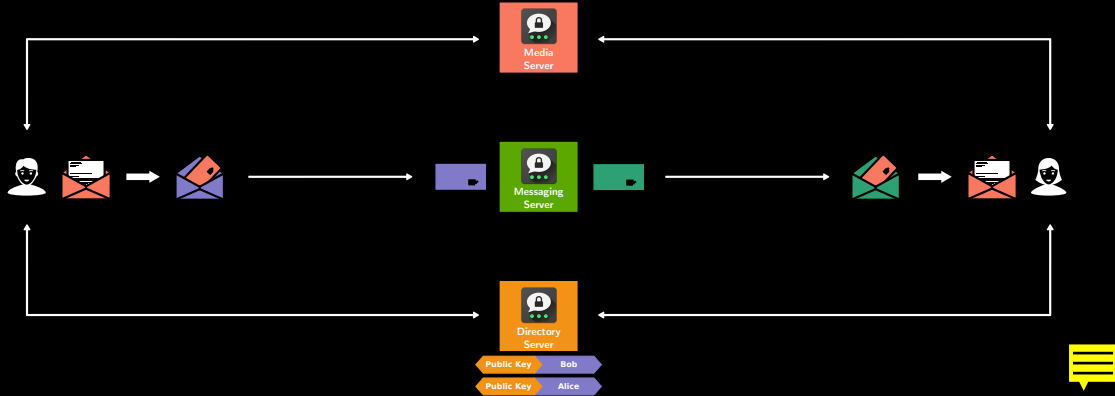We can obfuscate the size of a message with padding

We can obfuscate the size of a message with padding

Threema Fingerprints

NaCl and Threema
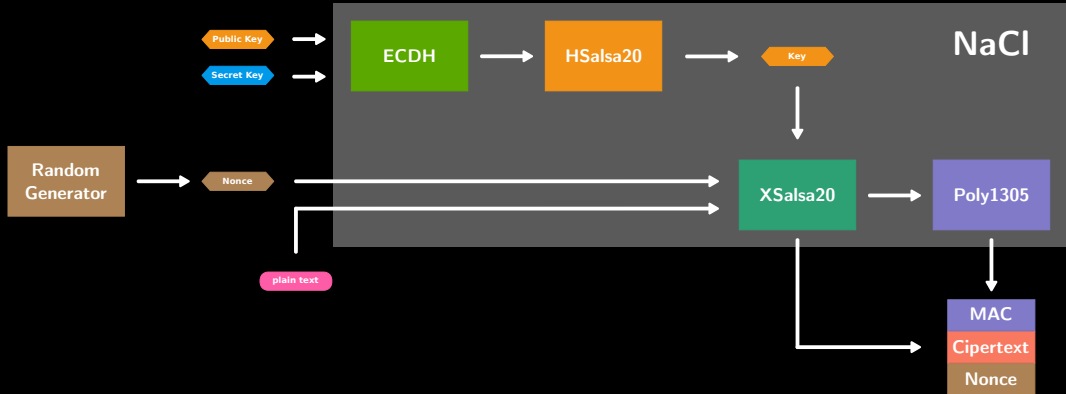
App

Threema

Client Hello →

← Server Hello

Client Auth Pkt →

← Server Ack
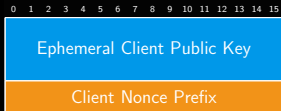
Exchange a set of ephemral keys and verify each others long term identity keys.

Client Hello Packet

- Client generates a ephemeral key pair
- Client generates random nonce prefix

App

Threema

Client Hello

Server Hello

Client Auth Pkt

Server Ack

Server Hello Packet

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Server Nonce Prefix

Ephemeral Server Public Key

Client Nonce Prefix
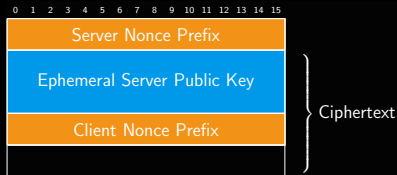
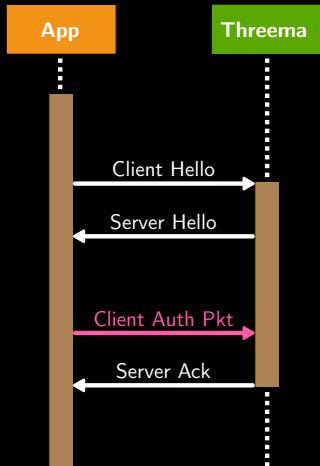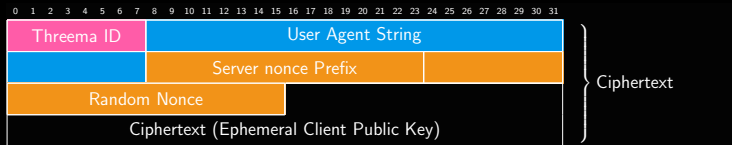Ciphertext

- Server generates ephemeral key pair
- Server generates random nonce
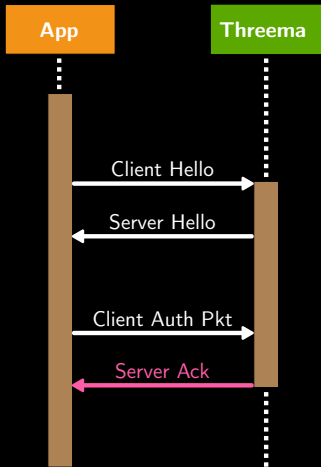- Ciphertext encrypted with Server Nonce, Client Ephemeral Key and Server Long-Term Key
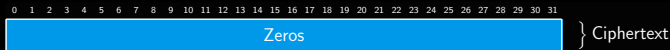
Client Authentication Packet

- Outer Encryption with ephemeral Keys
- Ciphertext links clients ephemeral key pair to it's long term key pair

Threema's App-to-Server Handshake

App → Threema:
- Client Hello
- Server Hello
- Client Auth Pkt
- Server Ack

Server Acknowledgement Packet

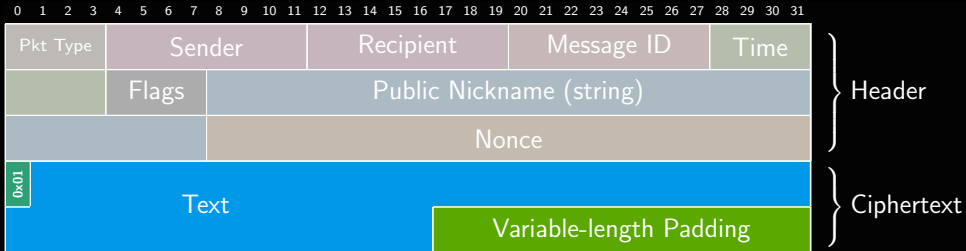| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 |
|---|
| Zeros |

} Ciphertext

- Server comfirms everything worked fine by encrypting something with both ephemeral keys
- We have established a forward secure channel between app and messaging server.
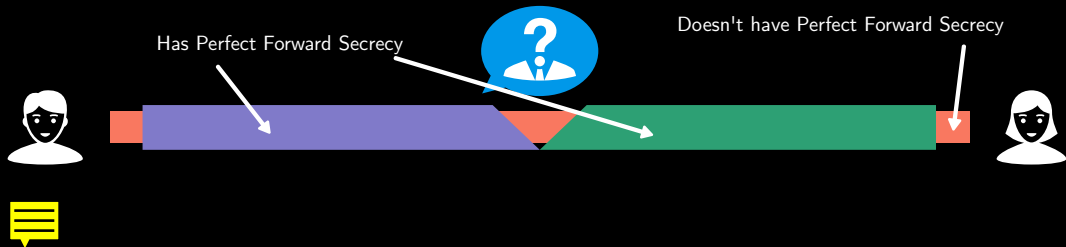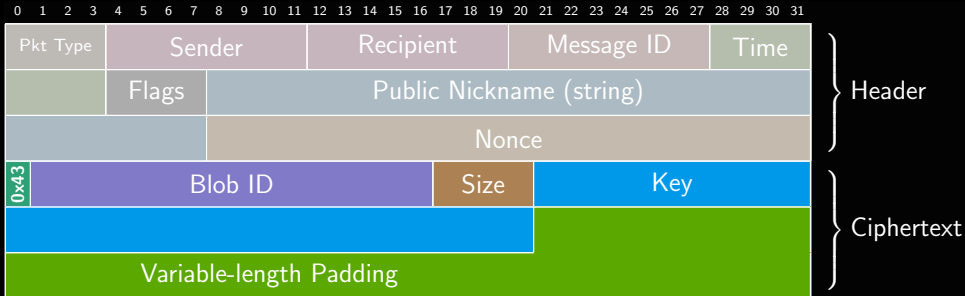
Media Server

Blob ID

Blob

Messaging Server

Directory Server
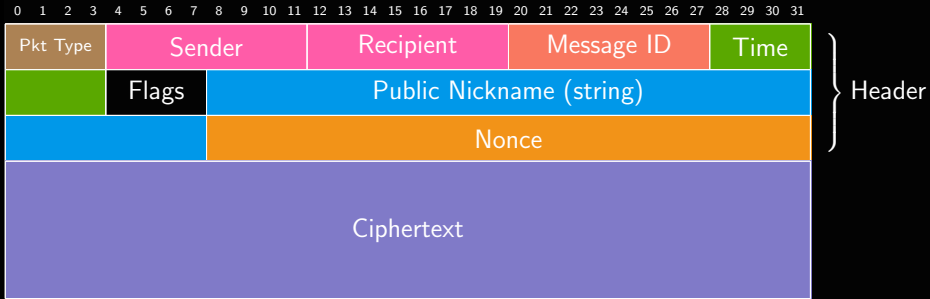
Public Key | Bob

Public Key | Alice
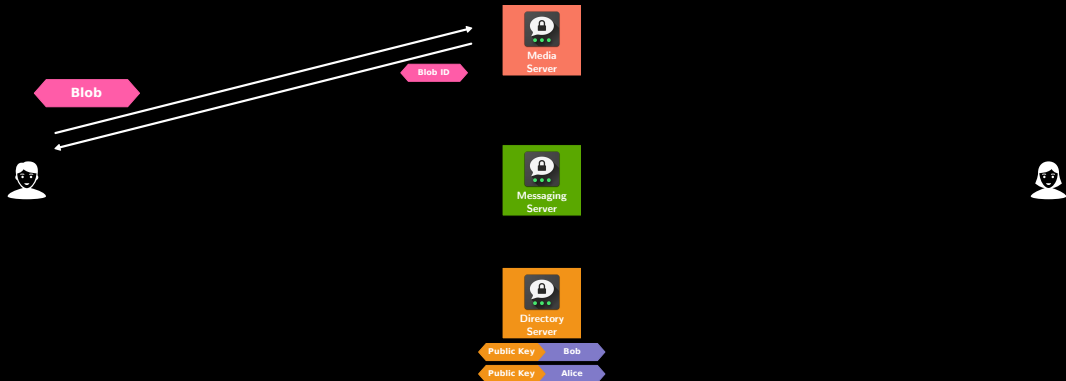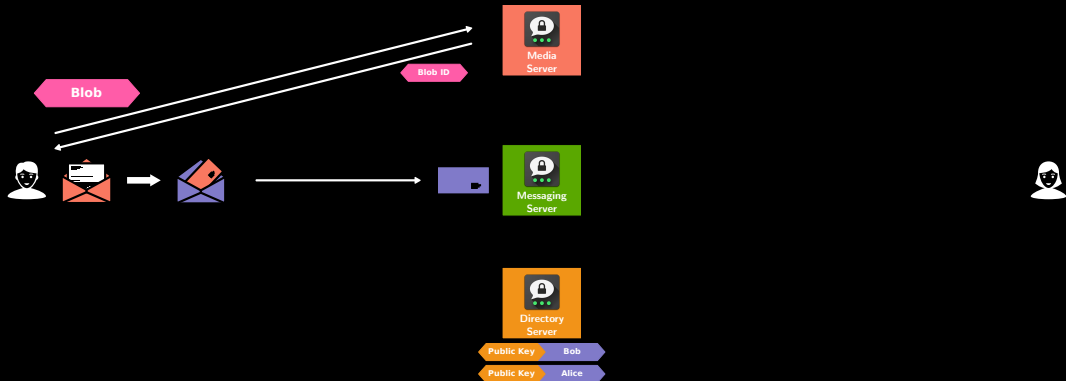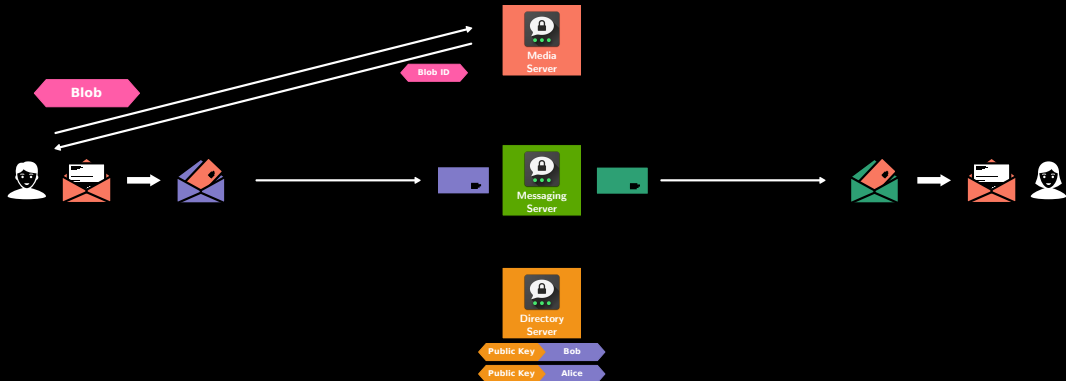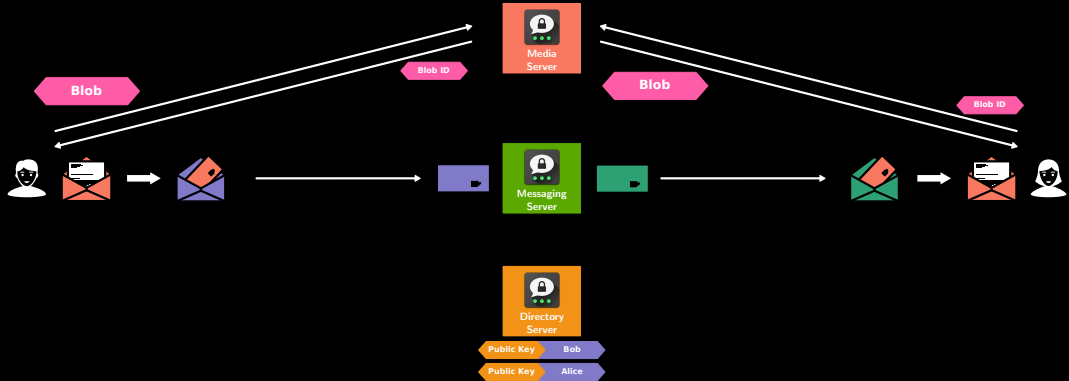
Basic messaging functionality achieved.
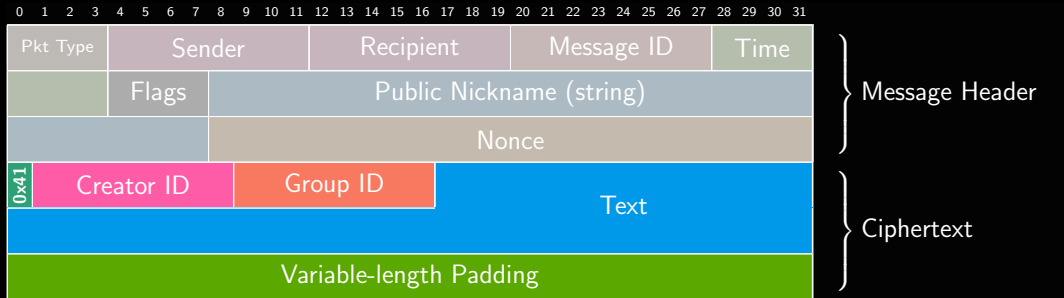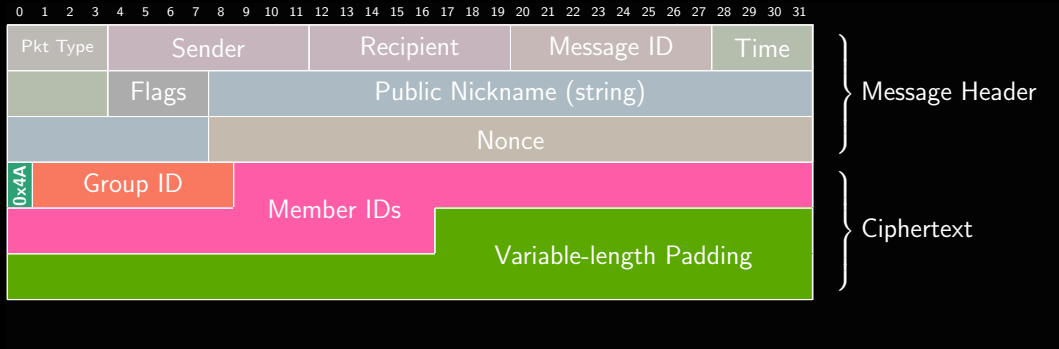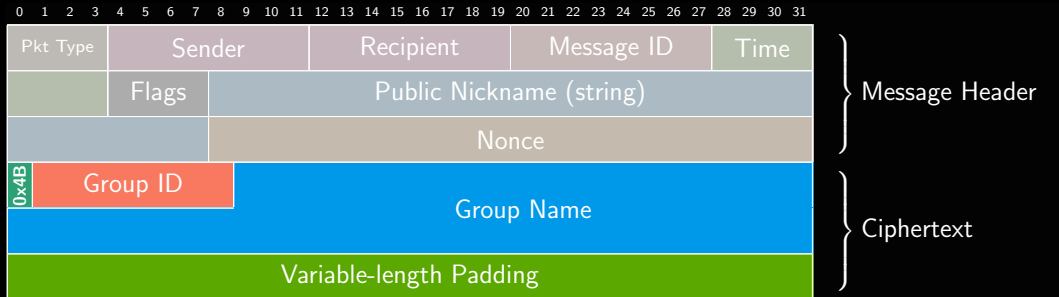
Sammlung kleinerer Dinge, die uns aufgefallen sind

- Media messages could be StageFright attach vectors
- The protocol implementation looks sound to us but the message design prevents feature upgrades on the protocol (not text-protocol) level

- Test for common pitfalls in implementation
  - Handling of TLS
  - Handling of keys and nonces
  - NaCl implementation errors
  - Uncommon data leaks
  - Bugs
  - …?
- Find out how protocol is designed
  1. Understand handshakes
  2. Understand protocol
  3. decipher messages

Positive side-note: Threema had released a security white paper early on

Thank You!

Roland Schilling

✉ schilling@tuhh.de

🐦 @NerdingByDoing

Frieder Steinmetz

✉ frieder.steinmetz@tuhh.de

🐦 @twillnix

Beamer Theme: Metropolis by Matthias volgelsang
Color Theme: Owl by Ross Chirchley
Icons: The BIG collection by Sergey Demushkin
                Foundation Icon Fonts 3 by ZURB

Thanks to Jan Ahrens and Philipp Berger – their work has made ours somewhat easier
Thanks to Maximilian Köstler for his initial work on Threema