A look into the Mobile Messaging Black Box

33rd Chaos Commmunication Congress #33c3

Roland Schilling   @NerdingByDoing
Frieder Steinmetz @twillnix
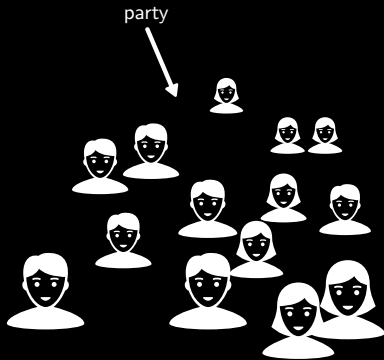
December 22, 2016

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
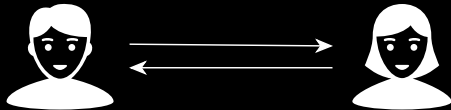- What does a separate room offer you?

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

It's a One-Time Talk

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
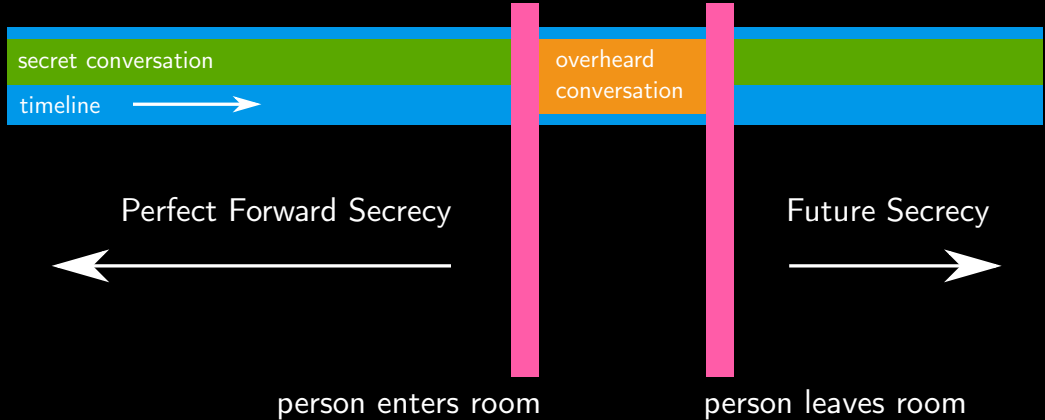
We call this future secrecy

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
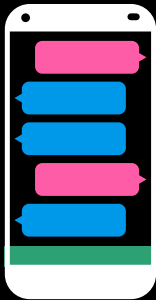
We call this deniability

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

→ Actually postal services are better to look at messaging from a technical point of view.

Example: Traditional Messaging

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
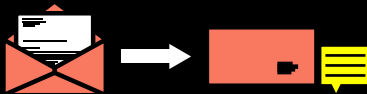- could store all messages and use them as proof of the exchange: no deniability

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

Symmetric Encryption:
$\rightarrow$ Encryption and decryption with the same key
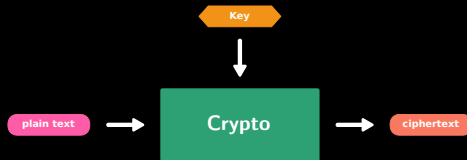
Symmetric Encryption:

→ Encryption and decryption with the same key

```
          Key                              Key
           │                                │
           ▼                                ▼
plain text → [ Crypto ] → ciphertext → [ Crypto ] → plain text
```
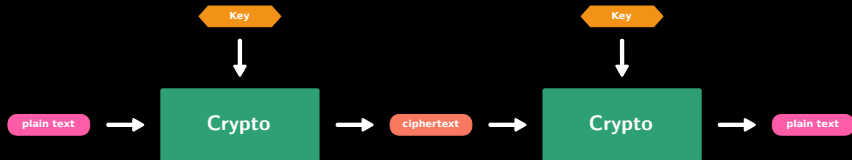
The Shortest Introduction to Encryption You Will Ever Get

Symmetric Encryption:
→ Encryption and decryption with the same key
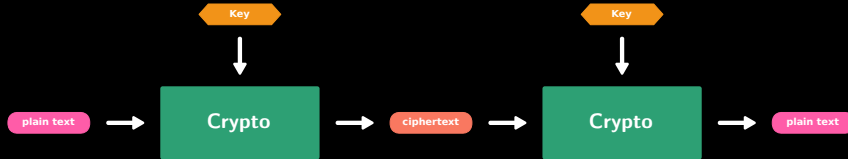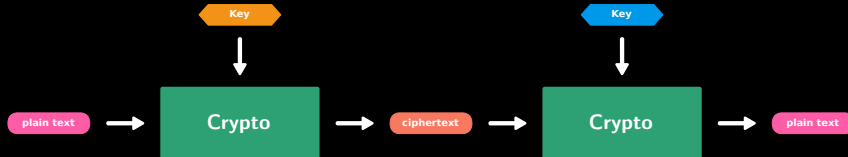

Asymmetric Encryption:
→ Encryption and decryption with different keys

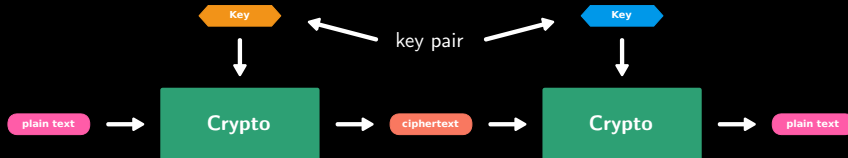# The Shortest Introduction to Encryption You Will Ever Get

Symmetric Encryption:
→ Encryption and decryption with the same key

| Key | | | Key | | |

plain text → **Crypto** → ciphertext → **Crypto** → plain text

Asymmetric Encryption:
→ Encryption and decryption with different keys

Key ← key pair → Key

plain text → **Crypto** → ciphertext → **Crypto** → plain text

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
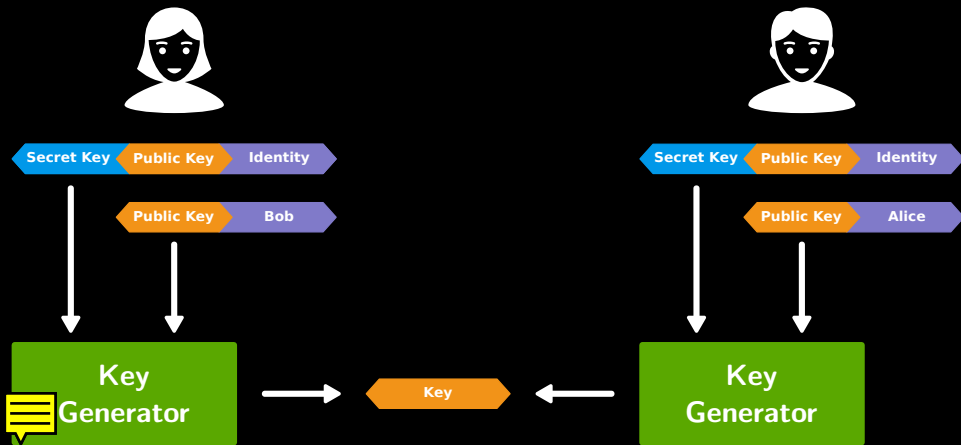
Public-Key Cryptography – In a Nutshell

- 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
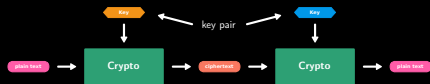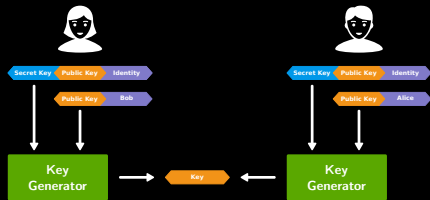
Asymmetric Encryption gives us IDs but is very expensive.

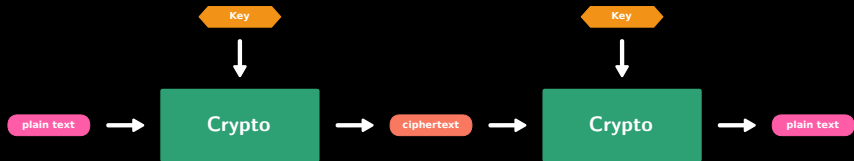**Symmetric Encryption** is cheap, but a key has to be shared by all participants *before* communication starts.

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

But there's more…

secret messages

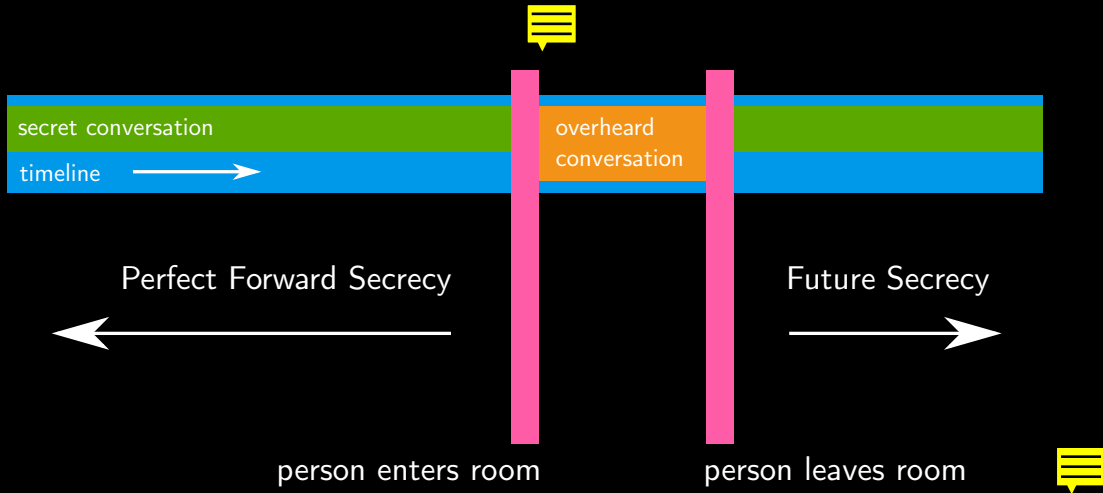compromised messages

timeline ⟶

Perfect Forward Secrecy

Future Secrecy

key compromise

key renegotiation

Authenticated Encryption gives us:

- Confidentiality
- Deniability
- Authenticity

We don't have:

- Perfect Forward Secrecy
- Future Secrecy

→ 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.
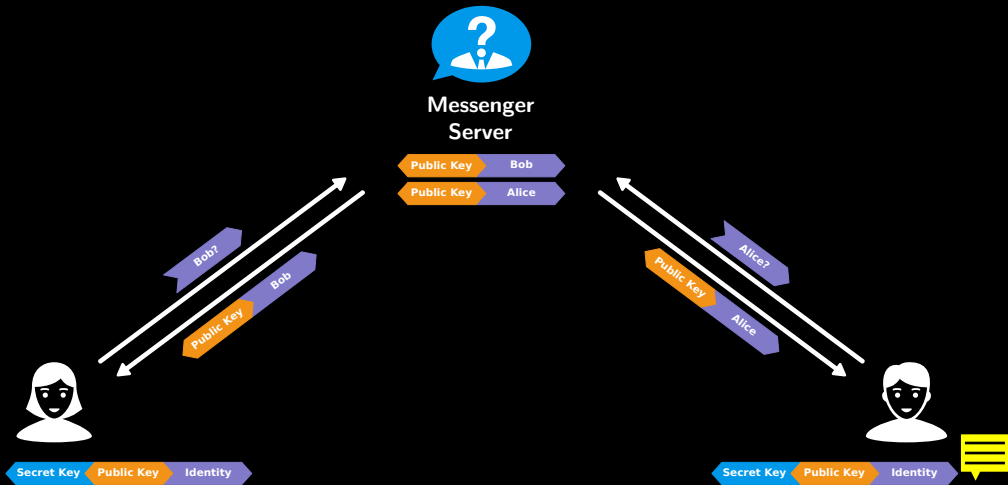
—Dieter Gollmann

We can ask for IDs, but what is an ID?

- A phone number?

- An email address?

- Something else?

We can ask for IDs, but what is an ID?

- A phone number?
    - → Can identify a user. But is also considered personal information.
- An email address?

- Something else?

We can ask for IDs, but what is an ID?

- A phone number?
  - → Can identify a user. But is also considered personal information.
- An email address?
  - → Same thing as with phone number. But a temporary email can be used.
- Something else?

We can ask for IDs, but what is an ID?

- A phone number?
    - → Can identify a user. But is also considered personal information.
- An email address?
    - → Same thing as with phone number. But a temporary email can be used.
- Something else?
    - → Dedicated IDs offer anonymous usage, but ID ownership must be verifiable.

We can ask for IDs, but what is an ID?

- A phone number?
  → Can identify a user. But is also considered personal information.
- An email address?
  → Same thing as with phone number. But a temporary email can be used.
- Something else?
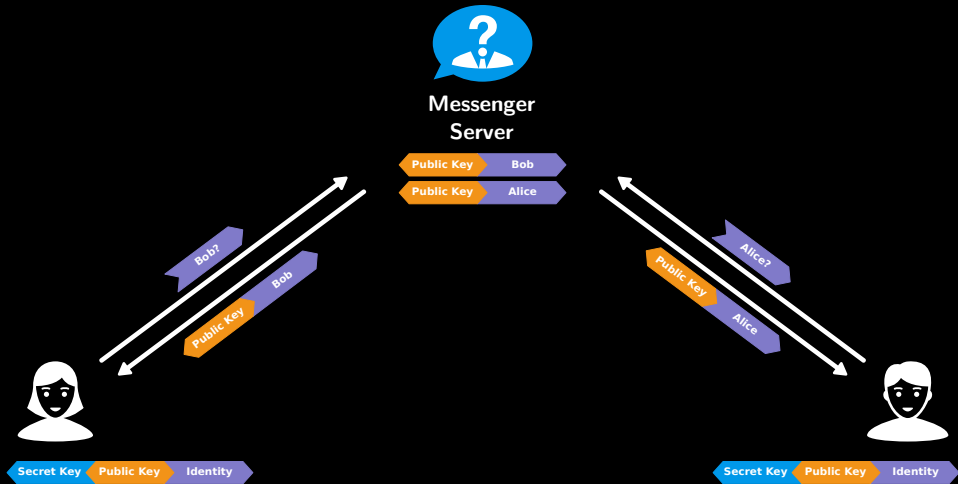 → Dedicated IDs offer anonymous usage, but ID ownership must be verifyable.

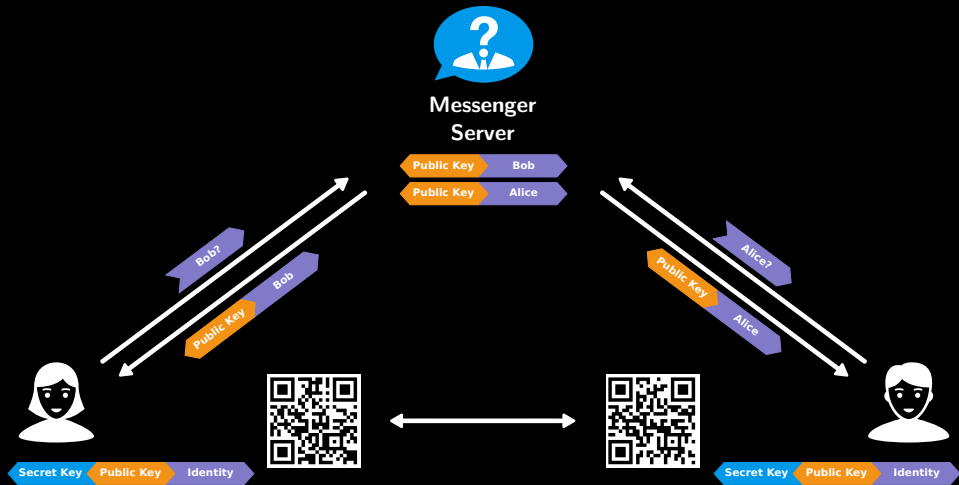→ Dedicated IDs are preferrable. But only if we find a way to verify ID ownership

How does Alice know which is Bob's public key?
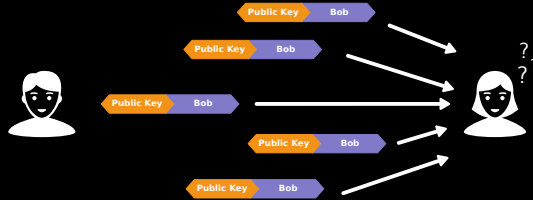
Mobile Messaging Key Management

- How to connect a key to a person?
  - $\rightarrow$ Key signing (PGP)
  - $\rightarrow$ Certificates (trusted third party)
  - $\rightarrow$ (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
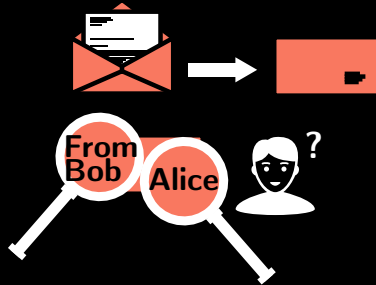
Solution: wrap encrypted message in a second layer of encryption and address it only to the message server.
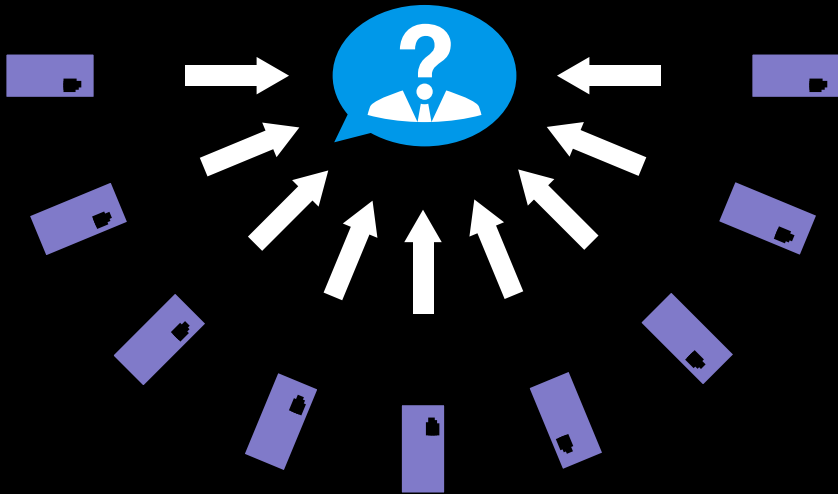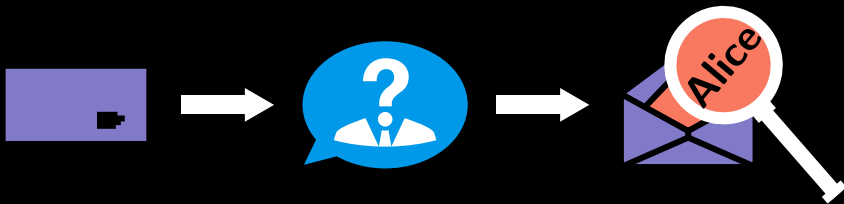
The message server will remove the outer layer and add a new one, targeted at the receiver.
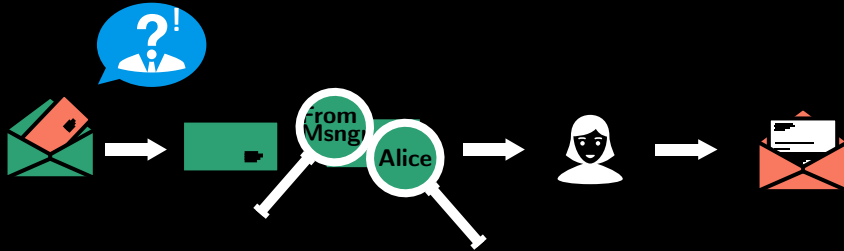
This leaves us with an encrypted end-to-end tunnel, transmitted through two transport layer encryption tunnels.

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
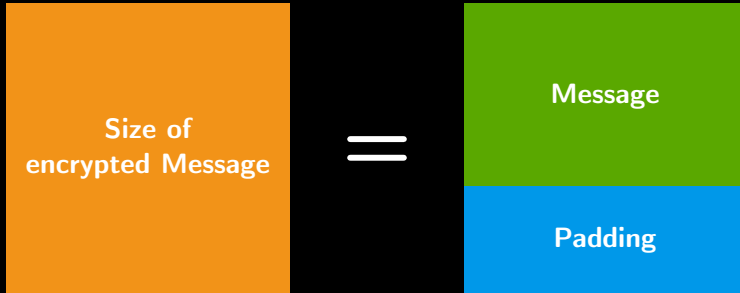
Threema's Architecture
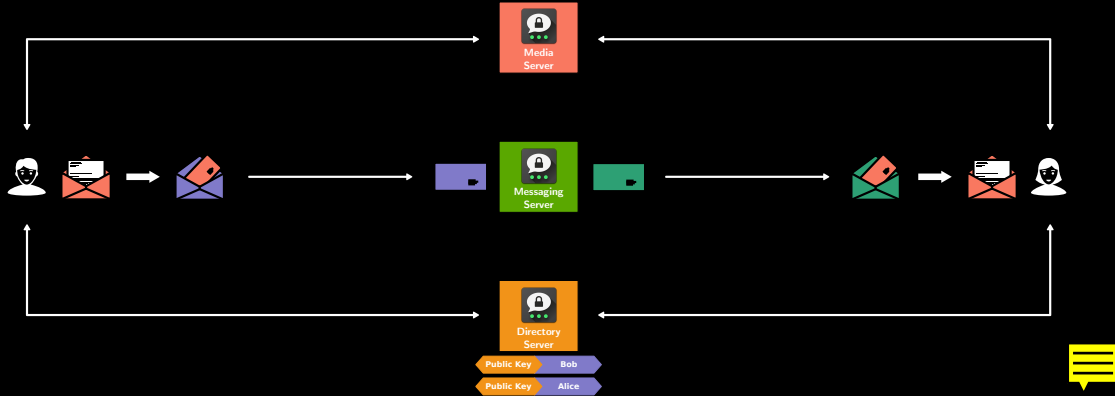
Threema Fingerprints
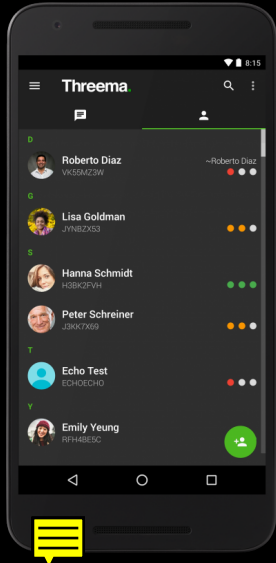
Threema offers dedicated IDs

- Users *may* provider their phone number and email.
- If provided, phone number and email are used for identification with the directory server.
- If no additional data is provided, IDs can only be exchanged manually.
- In either case, manual verification using QR codes is encouraged.
- The app permanently tracks the verification status of each peer ID.
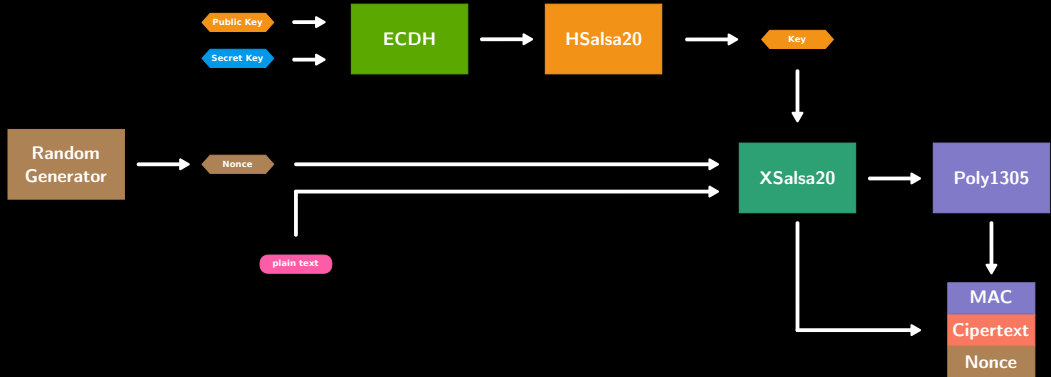
Threema's Handshake Between the App and the Messaging Server

Threema App

Threema Messaging Server

Client Hello →

← Server Hello

Client Auth Pkt →

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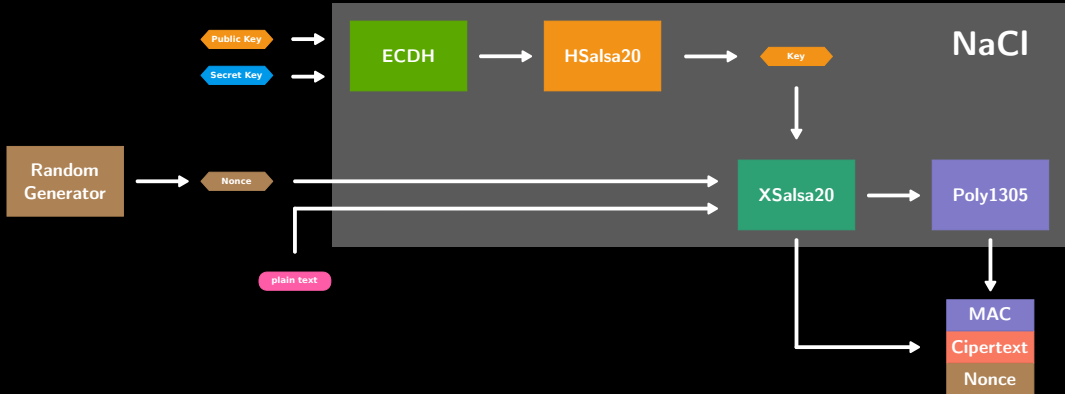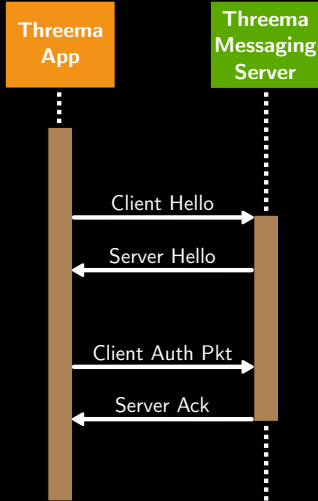
Exchange a set of ephemeral keys and verify each others long term identity keys.

Threema's Handshake Between the App and the Messaging Server
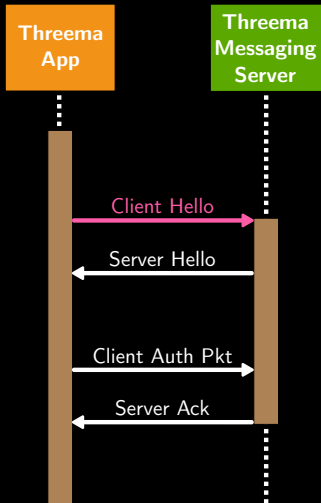
Client Hello Packet

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

Ephemeral Client Public Key

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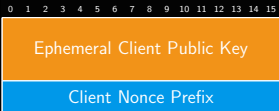
- Client generates a ephemeral key pair
- Client generates random nonce prefix
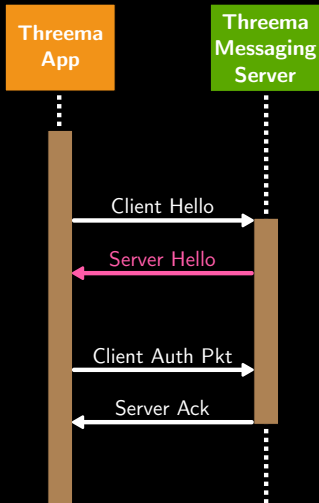
Server Hello Packet
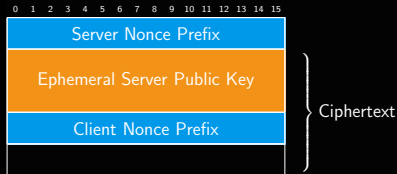
- Server generates ephemeral key pair
- Server generates random nonce
- Ciphertext encrypted with Server Nonce, Client Ephemeral Key and Server Long-Term Key

Threema's Handshake Between the App and the Messaging Server

Client Authentication 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

Threema ID — User Agent String
— Server nonce Prefix —
Random Nonce
Ciphertext (Ephemeral Client Public Key)
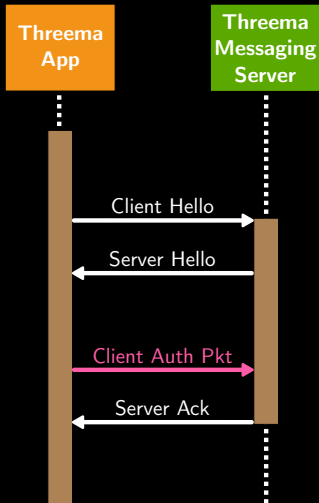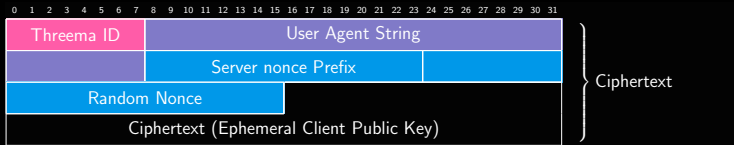
Ciphertext

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

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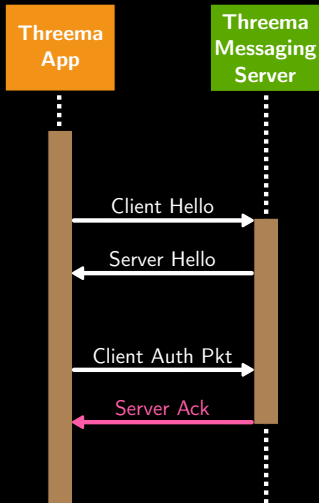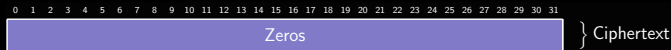
- Server comfirms everything worked fine by encrypting something with both ephemeral keys
- We have established a forward secure channel between app and messaging server.

Threema App

Threema Messaging Server

Client Hello

Server Hello

Client Auth Pkt

Server Ack
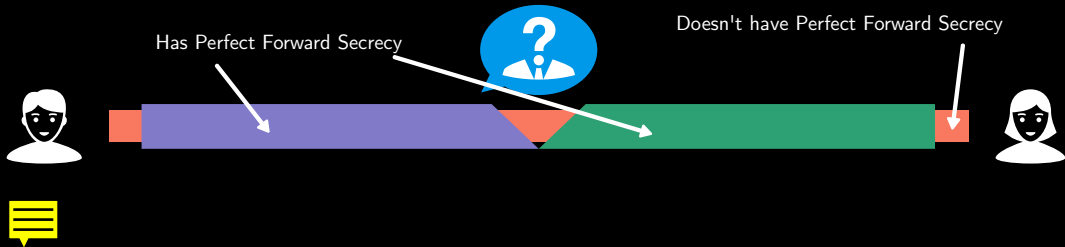
Threema Packet Format

Media Server

Messaging Server

Directory Server
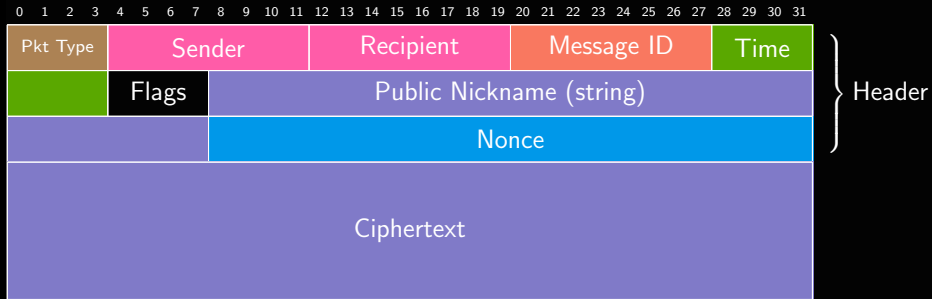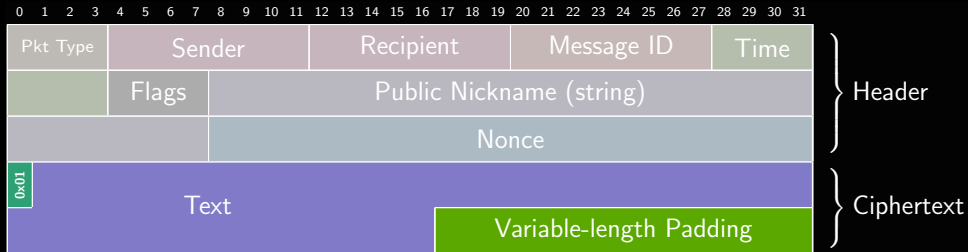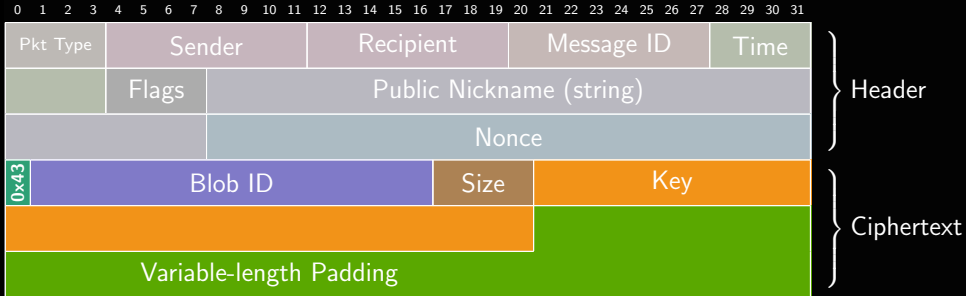
Public Key | Bob
Public Key | Alice
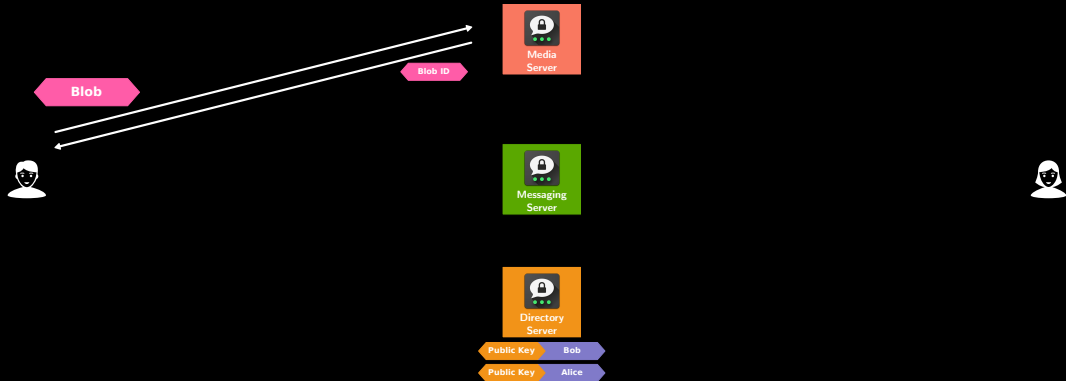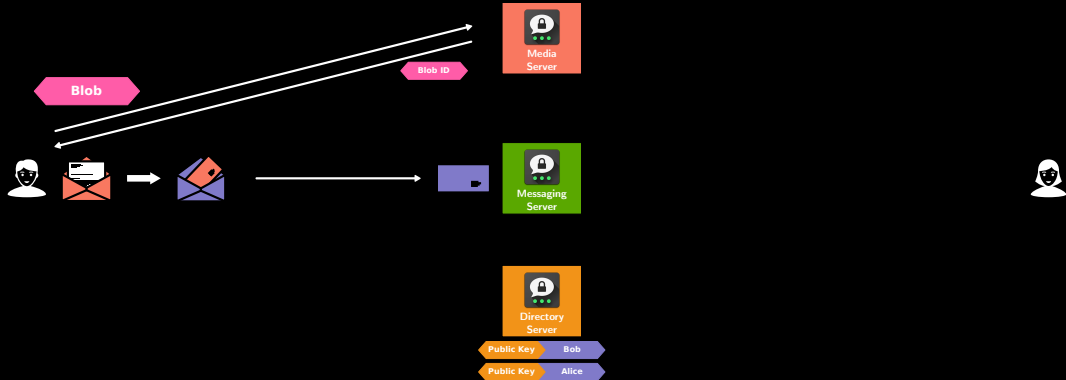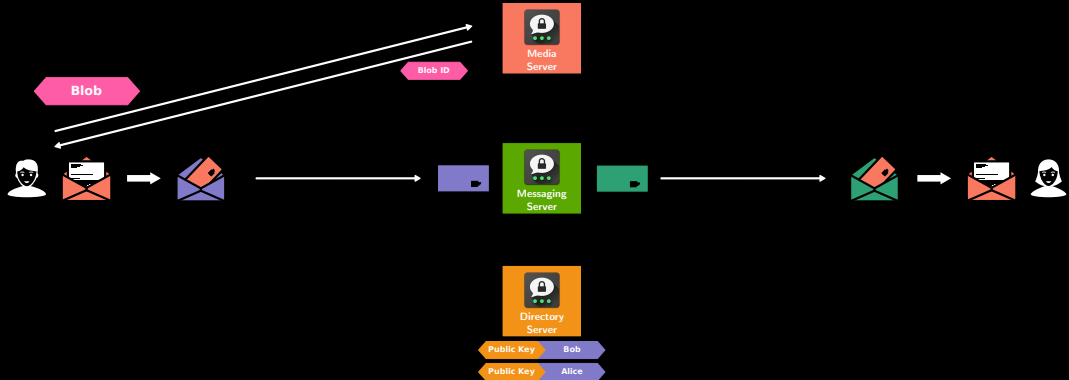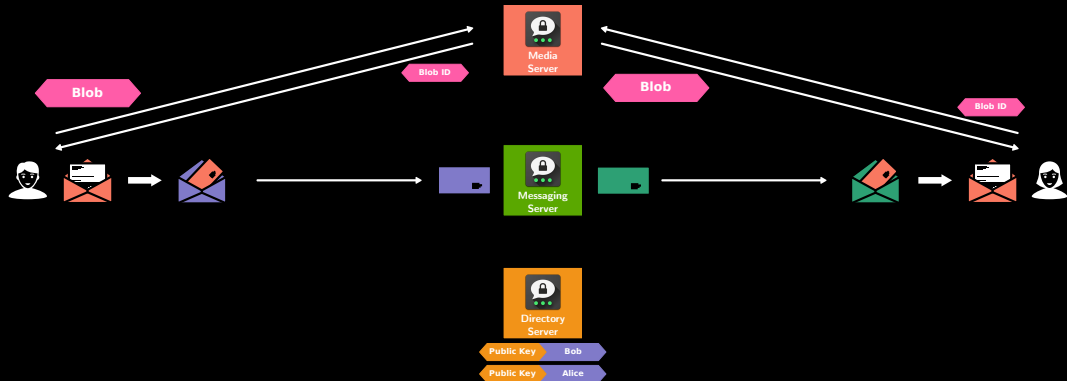
Basic messaging functionality achieved.
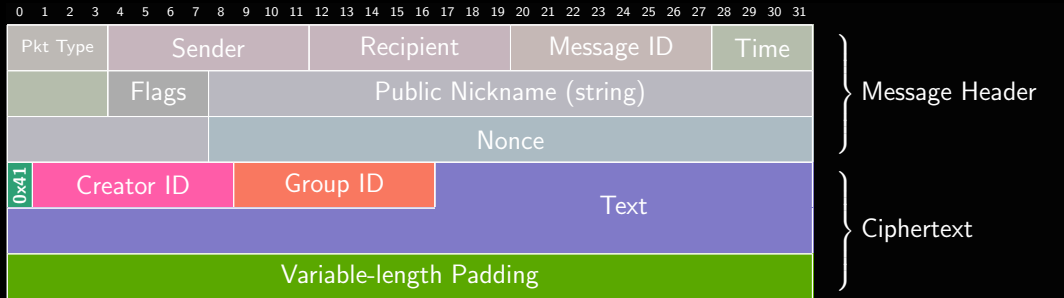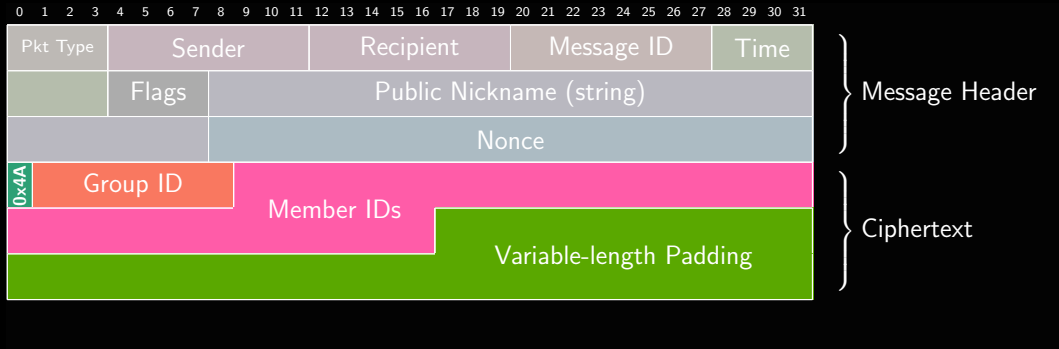
0 1 2 3 4 5 6 7	8 9 10 11 12 13 14 15	16 17 18 19 20 21 22 23	24 25 26 27 28 29 30 31	
Pkt Type	Sender	Recipient	Message ID	Time
	Flags	Public Nickname (string)		Message Header
	Nonce			
0x41	Creator ID	Group ID	Text	
				Ciphertext
Variable-length Padding				

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				
Pkt Type	Sender	Recipient	Message ID	Time
	Flags	Public Nickname (string)		
	Nonce			
0x4A Group ID	Member IDs			
		Variable-length Padding		

Message Header

Ciphertext
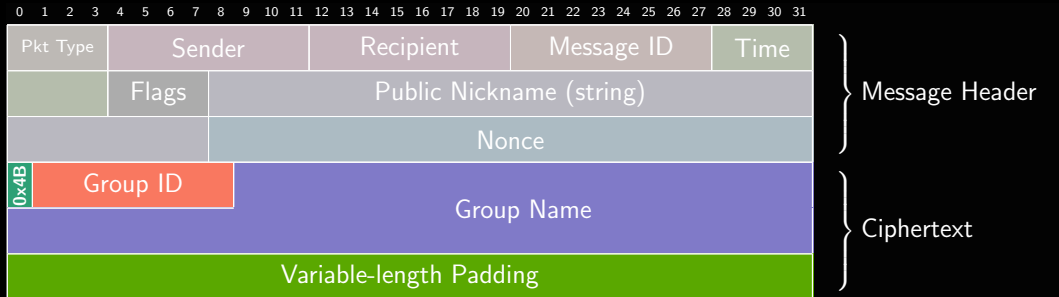
Sammlung kleinerer Dinge, die uns aufgefallen sind

- 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
NaCl slide was adapted from a figure in Threema's Cryptography Whitepaper

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

Message Packet (Threema Protocol Layer)
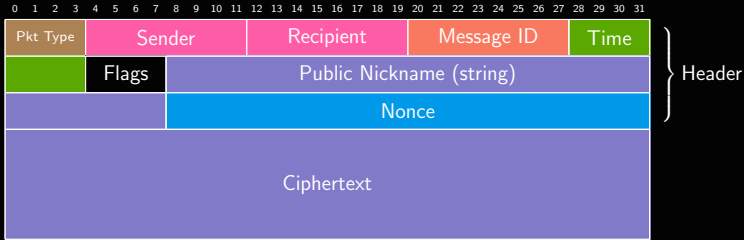
- Only the MSB of *Flags is used*

Message Packet on the Wire

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

Length

Threema Client-to-Server Ciphertext

Text Message

Image Message

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
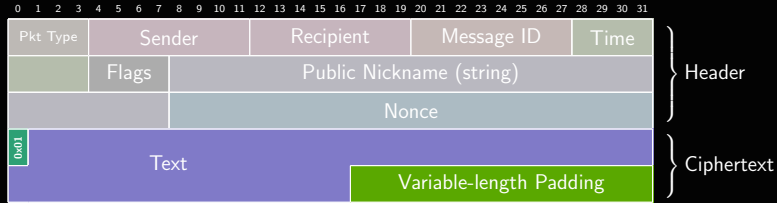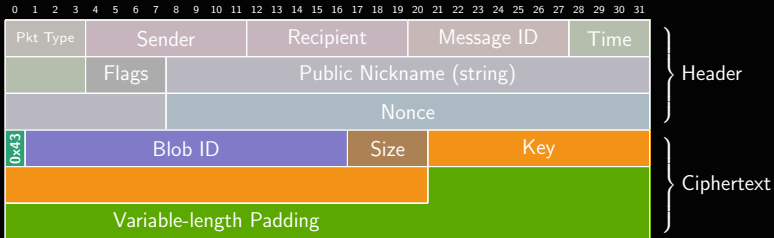
- Blob is symmetrically encrypted using *Key* and uploaded to asset server.

- Image captions are stored inside the image's EXIF data. These data leak upon creating such an image while the "save media to gallery" option is enabled.

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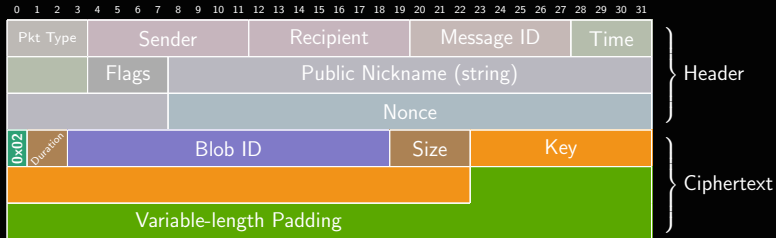
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Group Message 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 |

Pkt Type	Sender	Recipient	Message ID	Time
Flags	Public Nickname (string)		
	Nonce		

Message Header

| 0x41 | Creator ID | Group ID | Text |
| Variable-length Padding | | | |

Ciphertext

Group Image Message

Group Picture Update

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	
Pkt Type	Sender	Recipient	Message ID	Time
	Flags	Public Nickname (string)		
		Nonce		
80	Group ID	Blob ID	Size	
Symmetric key				
Variable-length Padding				

Message Header
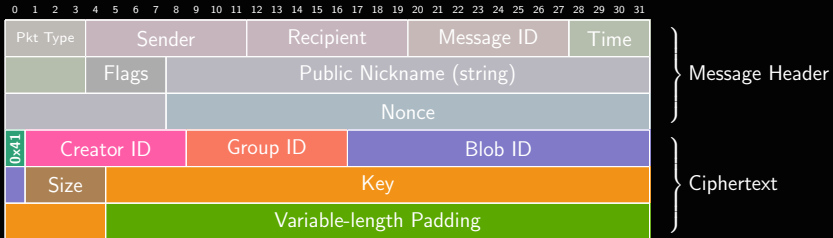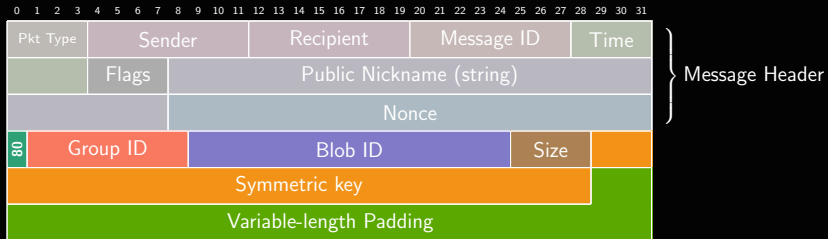
Create/Update Group (members)
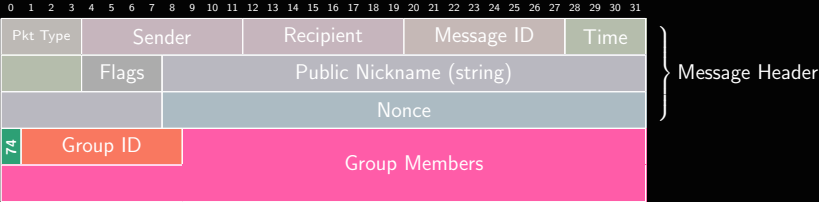
Acknowledgement Packet to Server

```
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
```

Length	Pkt Type	Sender	Message ID

$\longleftarrow$ Length $\longrightarrow$
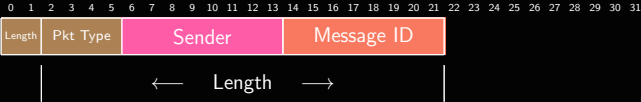
Client-Server Handshake

Client Hello

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

Ephemeral Client Public Key

Client Nonce Prefix

Server Hello

| 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

Appendix

Client Authentication 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

Threema ID	User Agent String	Ciphertext
	Server nonce Prefix	
Random Nonce		
Ciphertext (Ephemeral Client Public Key)		

Server Acknowledgement

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
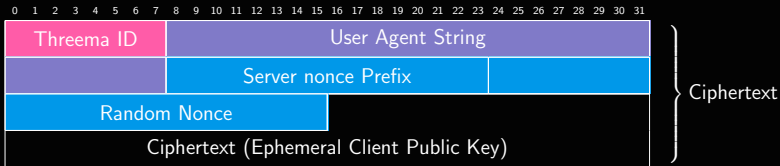
Zeros	Ciphertext
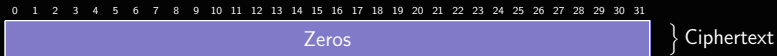
PKCS7 Padding

												03	03	03
											04	04	04	04
						08	08	08	08	08	08	08	08	
16	16	16	16	16	16	16	16	16	16	16	16	16	16	16
									05	05	05	05	05	
							06	06	06	06	06	06		

Group Management Message - Add Users
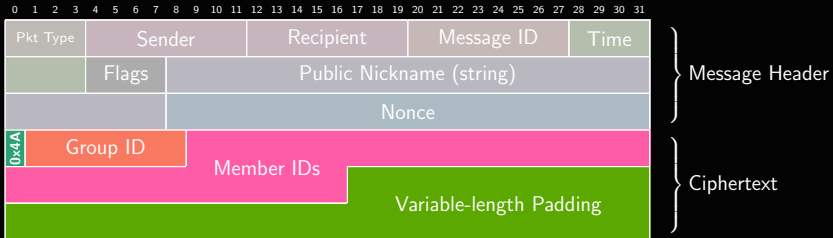
Group Management Message - Rename Group