

# A look into the Mobile Messaging Black Box

33<sup>rd</sup> Chaos Communication Congress #33c3

---

Roland Schilling   @NerdingByDoing

Frieder Steinmetz   @twillnix

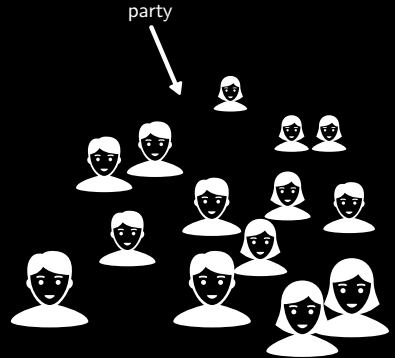
December 23, 2016

Hamburg University of Technology  
Security in Distributed Applications

# Messaging – Identifying Our Expectations

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?

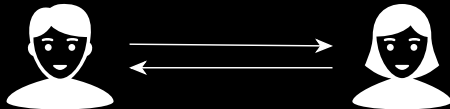


# A Private Room

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**



## You Know Each Other

---

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**

## In Sight of Each Other

---

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**

→ After leaving they would not be able to listen to any future conversations you might have

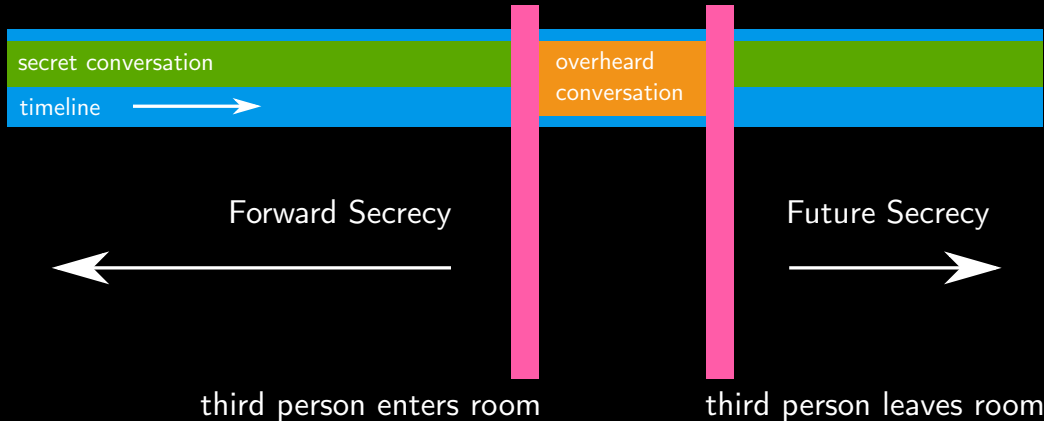
We call this **future secrecy**



# It's a One-Time Talk



## Forward- and Future Secrecy



# It's a One-Time Talk Between Only You Two

---

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**



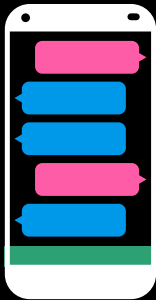
## Messaging – Reality Check



## Messaging – A More Technical Analogy

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**
  - not guarantee any secrecy, so we have neither **forward secrecy** nor **future secrecy**
- We could argue having **deniability** though.

→ Messaging translates badly to our offline communication expectation

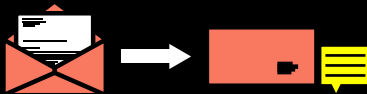


# From Postcards to Letters

---



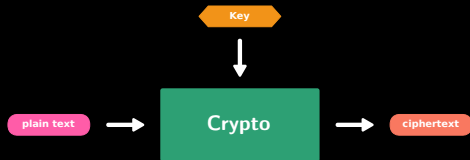
# From Postcards to Letters



# The Shortest Introduction to Encryption You Will Ever Get

Symmetric Encryption:

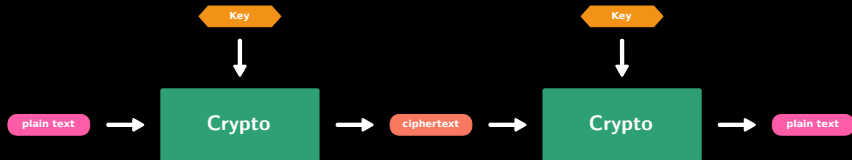
→ Encryption and decryption with the same key



# The Shortest Introduction to Encryption You Will Ever Get

Symmetric Encryption:

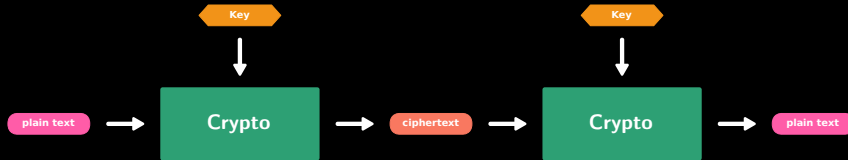
→ Encryption and decryption with the same key



# The Shortest Introduction to Encryption You Will Ever Get

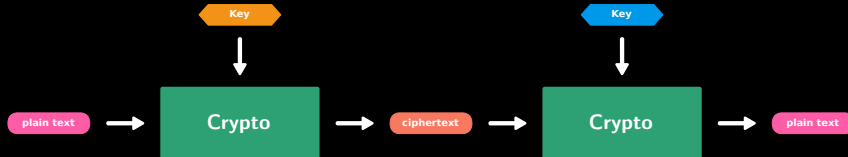
Symmetric Encryption:

→ Encryption and decryption with the same key



Asymmetric Encryption:

→ Encryption and decryption with different keys





# The Shortest Introduction to Encryption You Will Ever Get

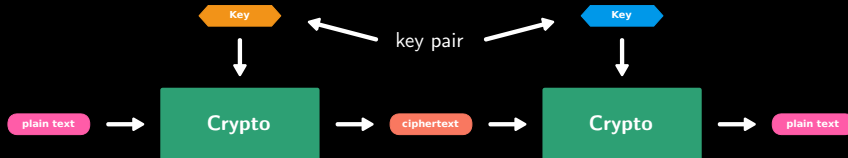
Symmetric Encryption:

→ Encryption and decryption with the same key



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



# 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



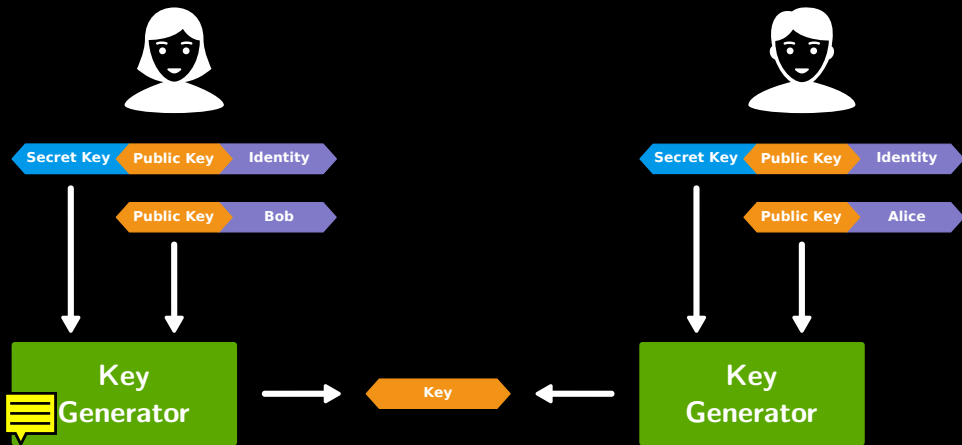
# Public-Key Cryptography – In a Nutshell



- 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

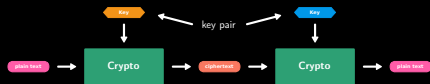


# Key Agreement



# Recap

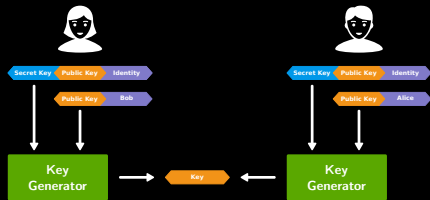
**Asymmetric Encryption** gives us IDs but is very expensive.



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

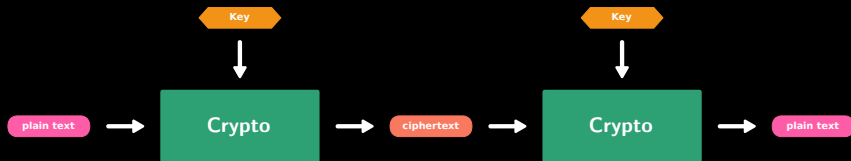


**Key Agreement** allows us to create symmetric keys based on asymmetric key pairs.



But there's more...

# Confidentiality



# Deniability

**From:**  
either of us

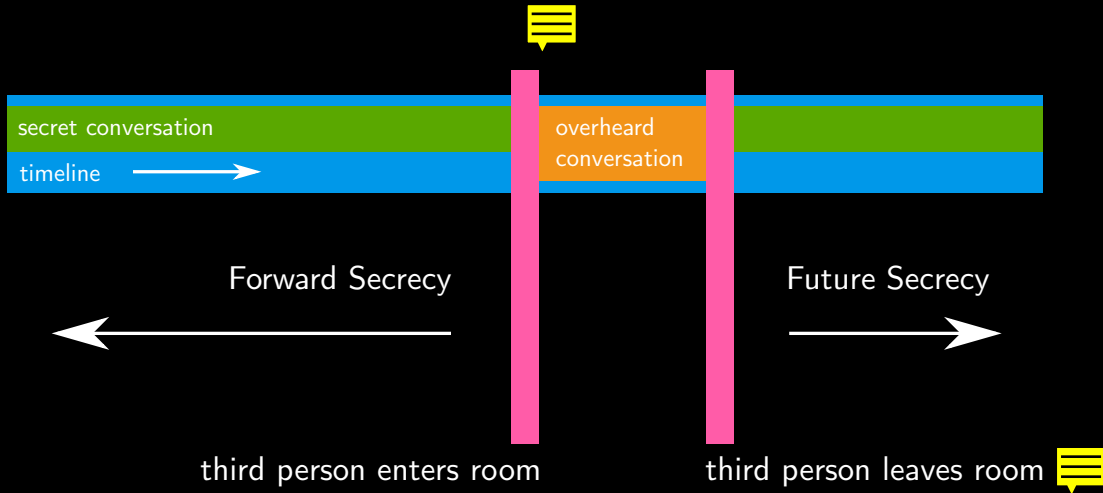


**To:**  
both of us

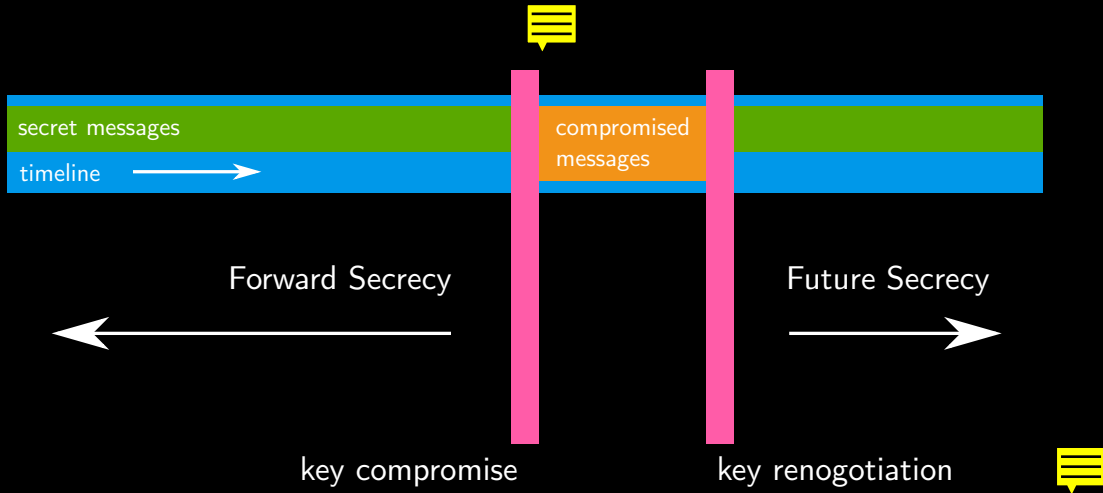




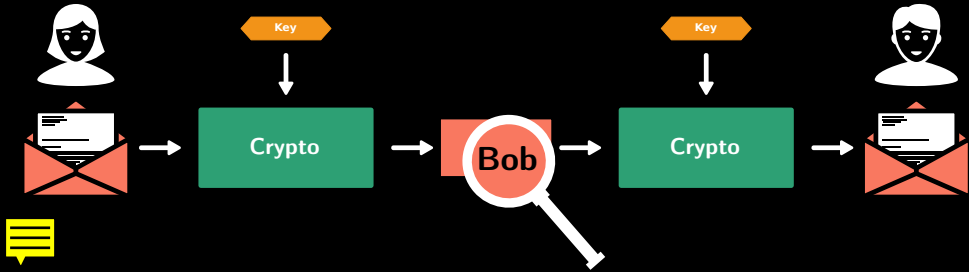
## But What About Forward- and Future Secrecy?



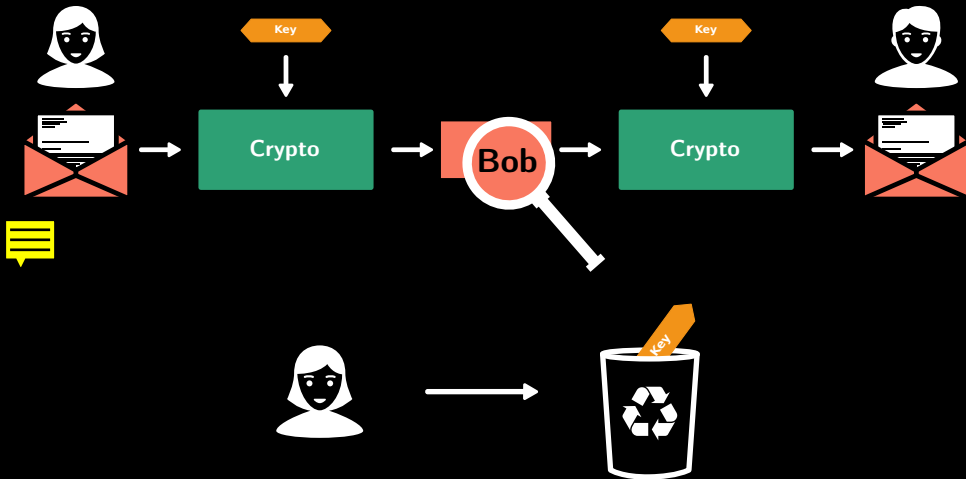
## But What About Forward- and Future Secrecy?



## But What About Forward- and Future Secrecy?



## But What About Forward- and Future Secrecy?



# Recap

---

Our key agreement protocol gives us:

- Confidentiality
- Deniability
- Authenticity

We don't have:

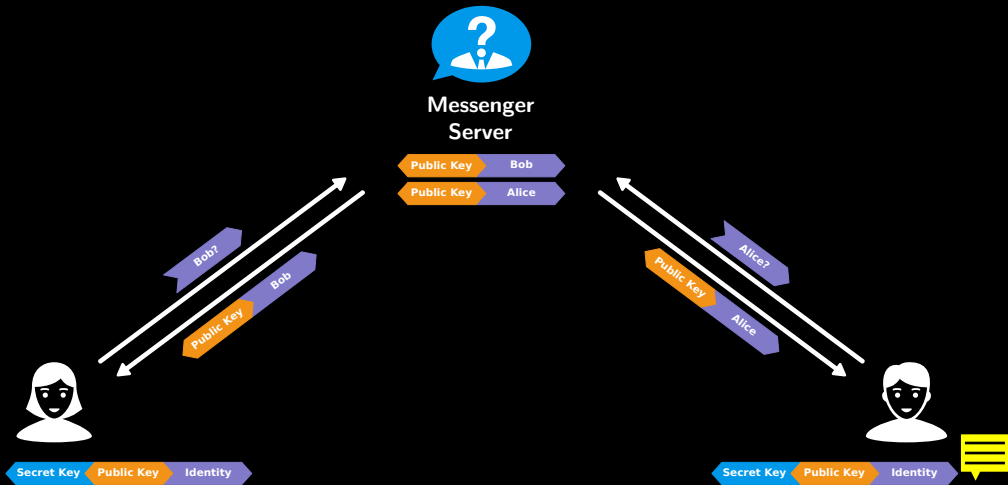
- Forward Secrecy
- Future Secrecy

→ 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

# Key and ID Management



# Key and ID Management

---

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

- A phone number?
- An email address?
- Something else?



# Key and ID Management

---

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?

# Key and ID Management

---

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?

# Key and ID Management

---

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.

# Key and ID Management

---

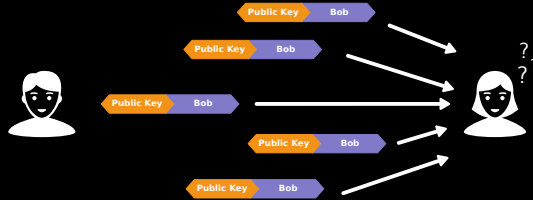
We can ask for IDs, but what is an ID?

- 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.

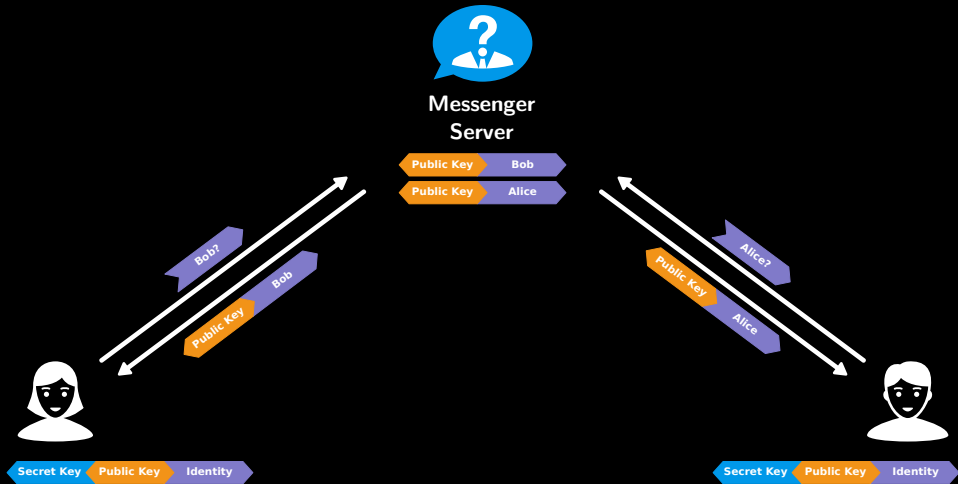
→ Dedicated IDs are preferable. But only if we find a way to verify ID ownership

# Key and ID Management

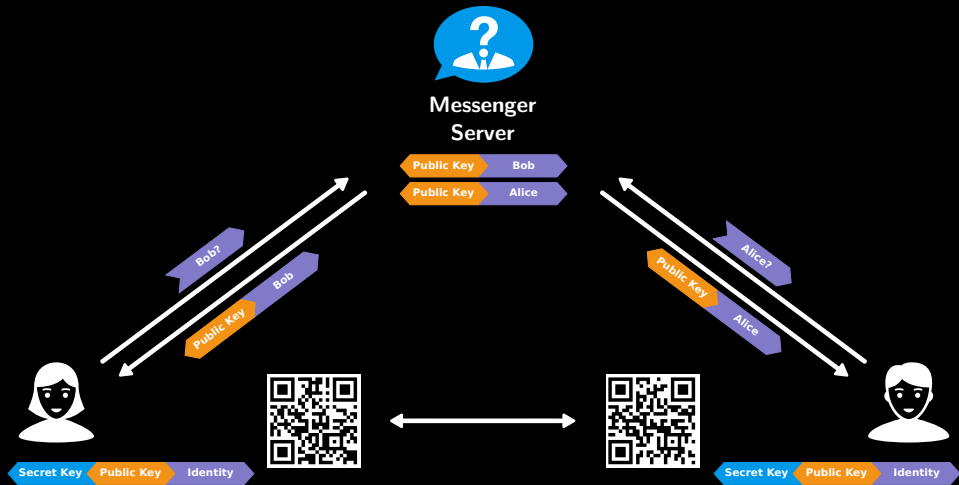
How does Alice know which is Bob's public key?



# Mobile Messaging Key Management



# Mobile Messaging Key Management



# Authenticity

---

We have now solved the Authenticity problem

- User can be identified by their phone number or email address
  - But they have dedicated IDs.
  - Personal verification is possible.

The remaining unsolved problem is a user changing their ID.

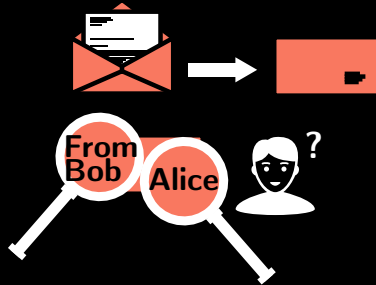
- At this point, the problem starts anew.
- We will get back to that later.



# Metadata Handling

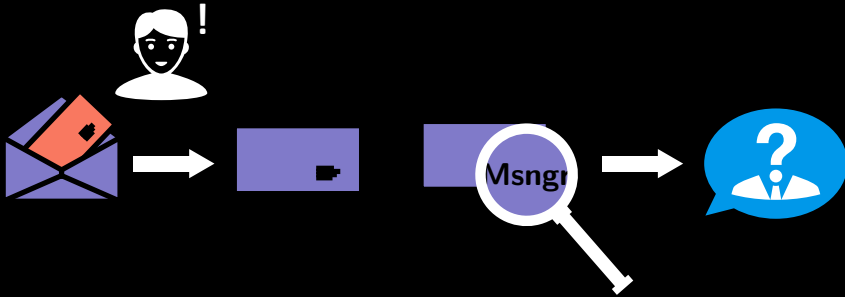
Everybody on the network can see:

- the sender of the message
- the intended receiver of the message

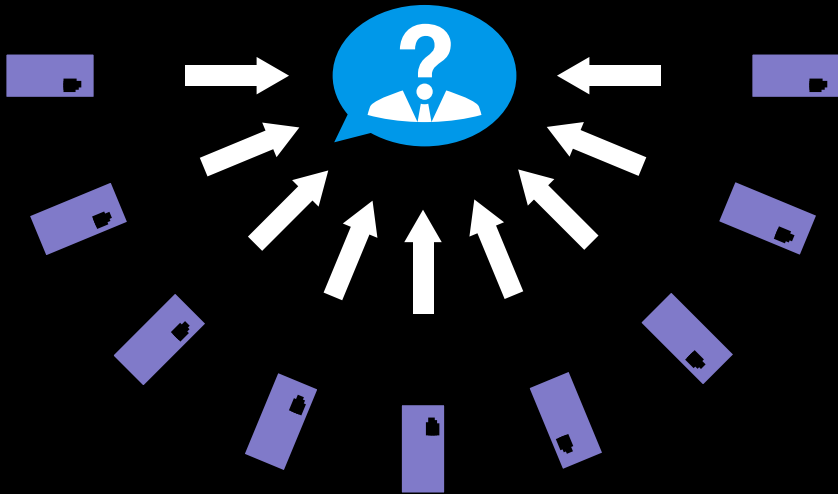


# Metadata Handling

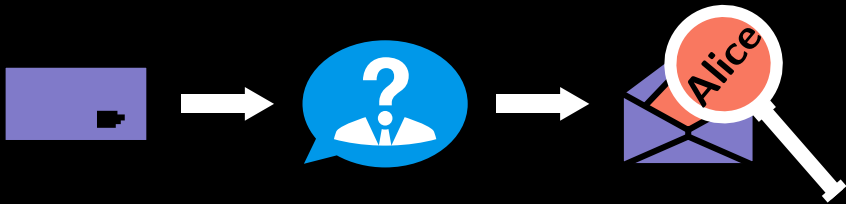
Solution: wrap encrypted message in a second layer of encryption and address it only to the message server.



# Metadata Handling

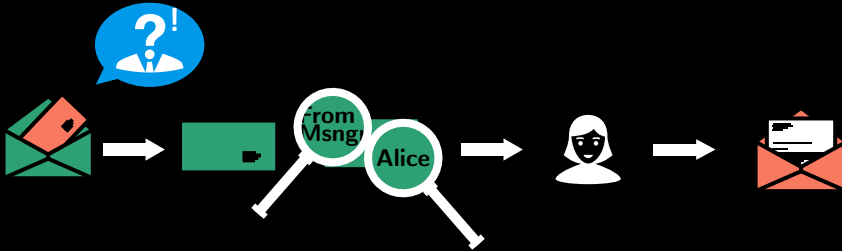


# Metadata Handling



# Metadata Handling

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



# Metadata Handling

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!

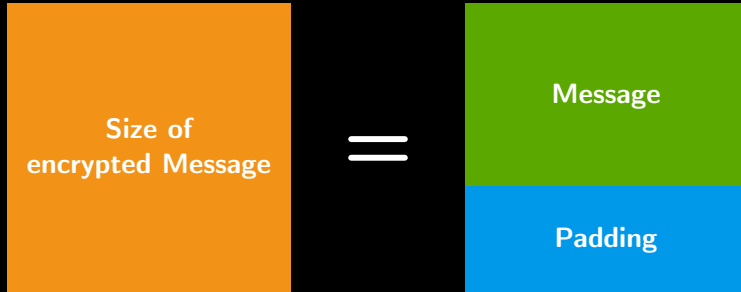
# Metadata Handling

---

We can obfuscate the size of a message with `padding`

# Metadata Handling

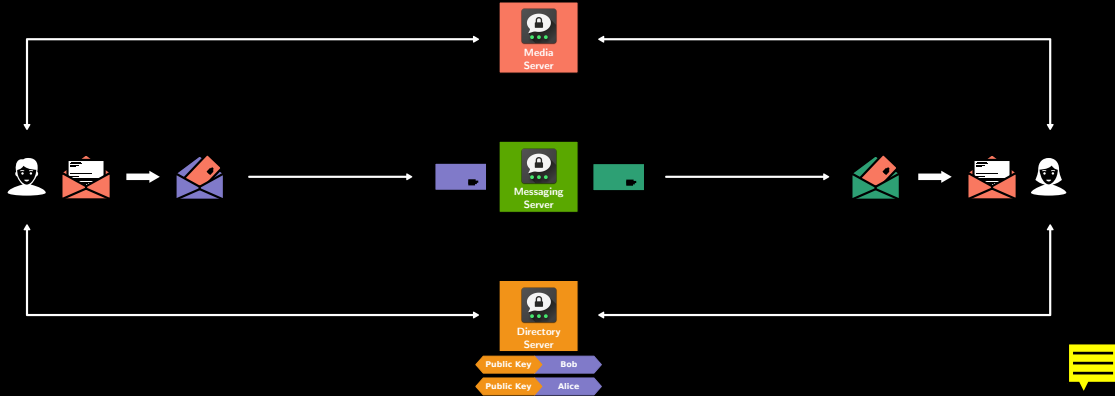
We can obfuscate the size of a message with **padding**



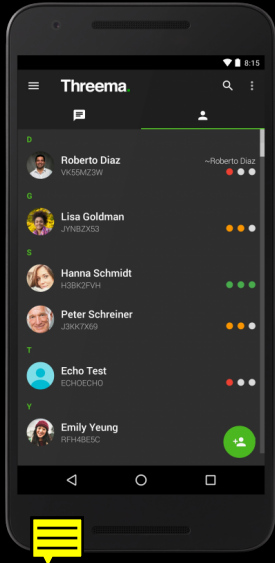




# Threema's Architecture



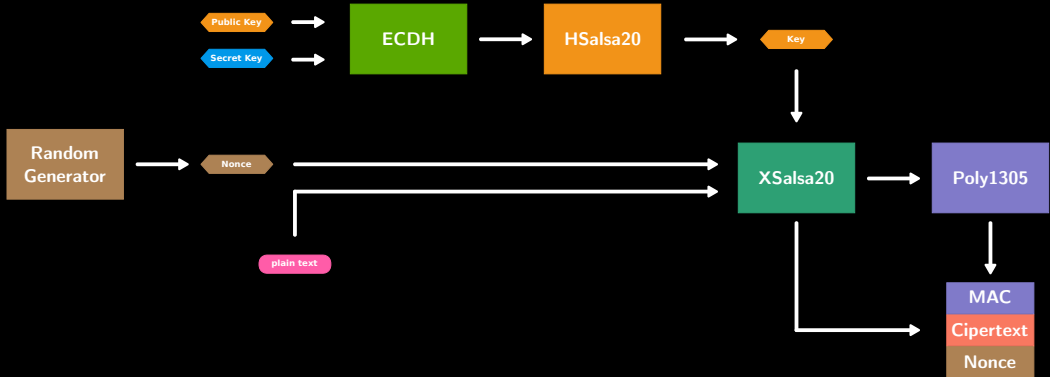
# Threema Fingerprints



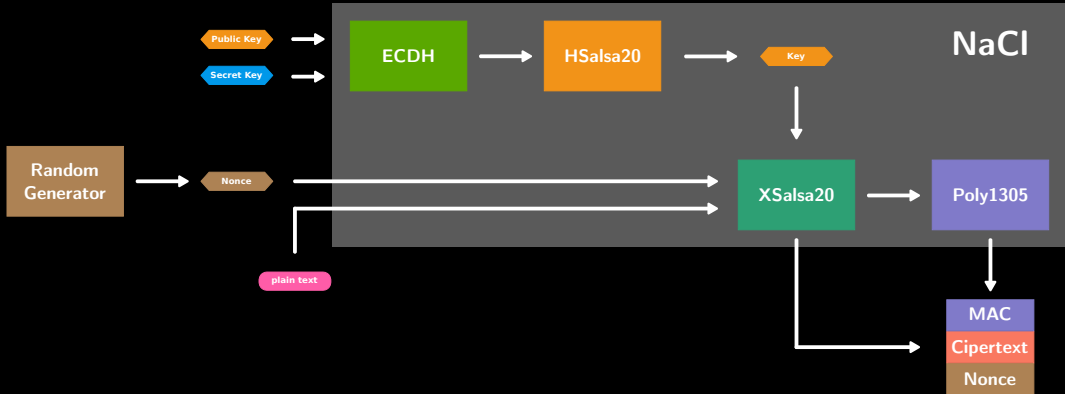
## Threema offers dedicated IDs

- Users **may** provide 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.

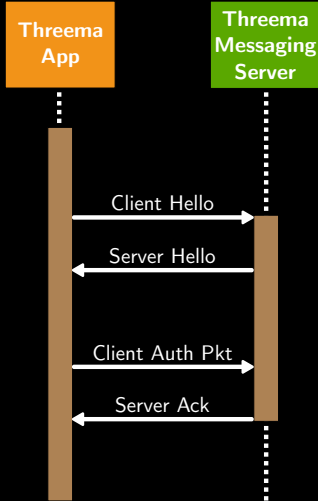
# NaCl and Threema



# NaCl and Threema

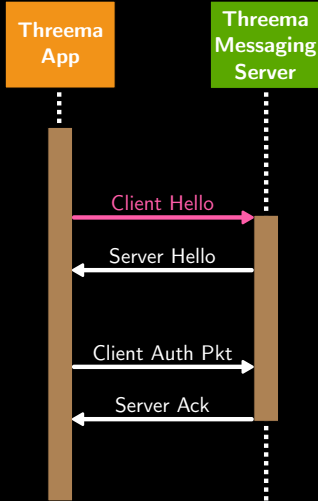


# Threema's Handshake Between the App and the Messaging Server

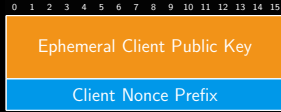


Exchange a set of **ephemeral keys** and verify each others long term identity keys.

# Threema's Handshake Between the App and the Messaging Server

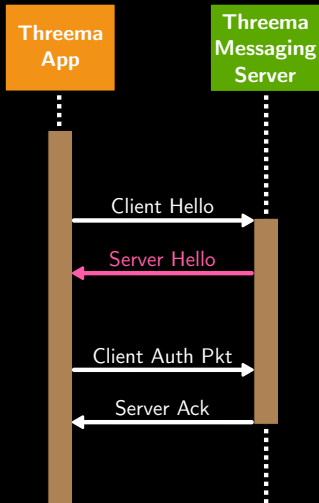


## Client Hello Packet

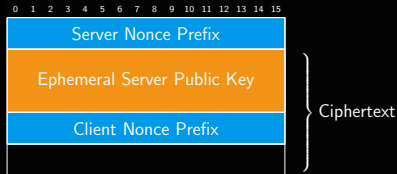


- Client generates a **ephemeral key** pair
- Client generates random nonce prefix

# Threema's Handshake Between the App and the Messaging Server



## Server Hello Packet

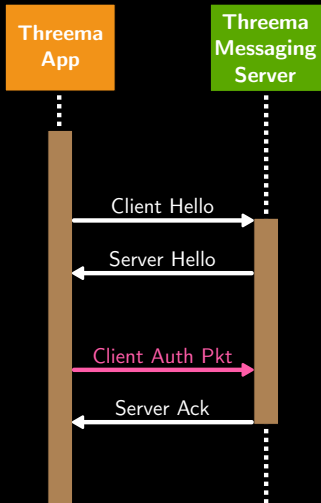


- Server generates ephemeral key pair
- Server generates random nonce
- Ciphertext encrypted with Server Nonce, Client Ephemeral Key and Server Long-Term Key

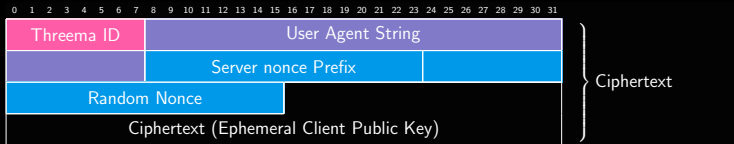




# Threema's Handshake Between the App and the Messaging Server

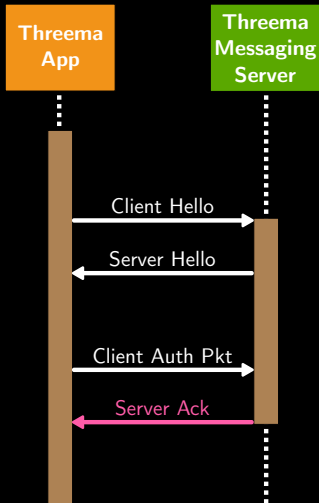


## Client Authentication Packet

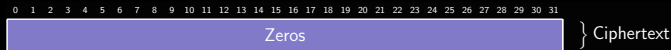


- Outer Encryption with **ephemeral Keys**
- Ciphertext links clients **ephemeral key** pair to it's long term key pair

# Threema's Handshake Between the App and the Messaging Server

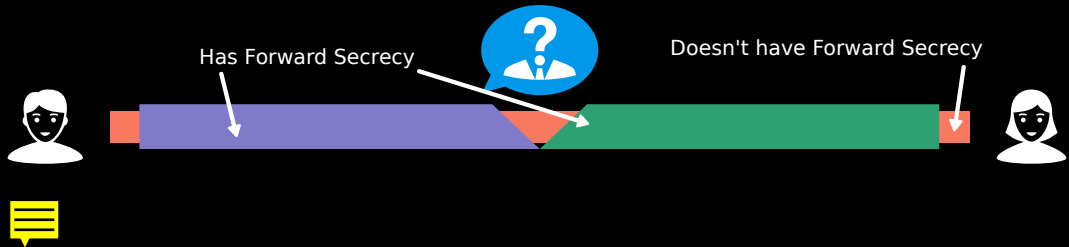


## Server Acknowledgement Packet

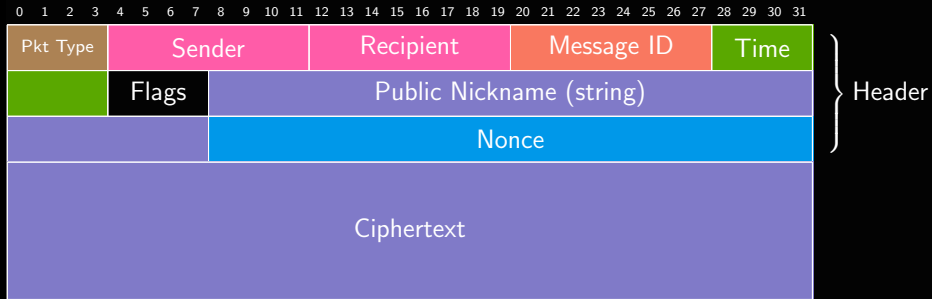


- Server confirms everything worked fine by encrypting something with both **ephemeral keys**
- We have established a forward secure channel between app and messaging server.

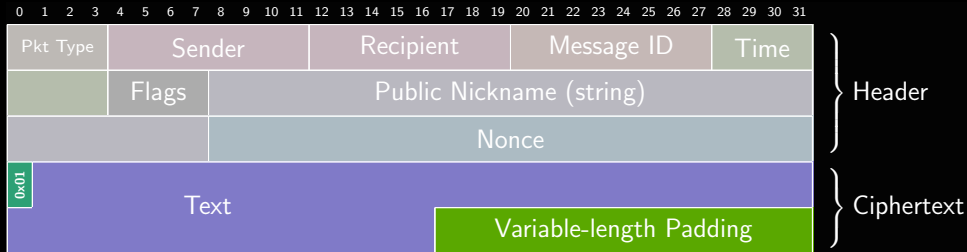
# A 2-Layer Tunnel



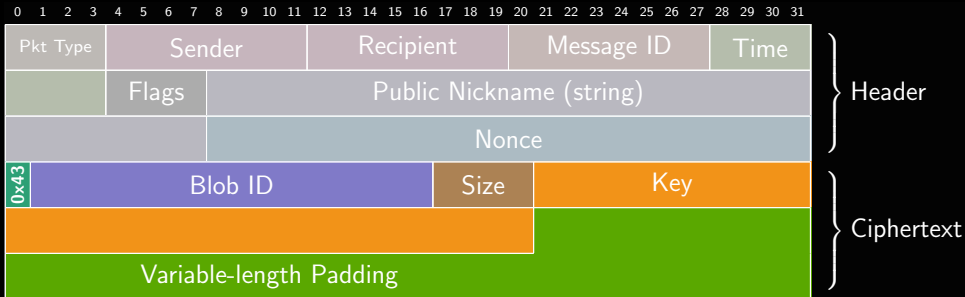
# Threema Packet Format



# Threema Text Messages



# Threema Image Messages



# Sending an Image Message

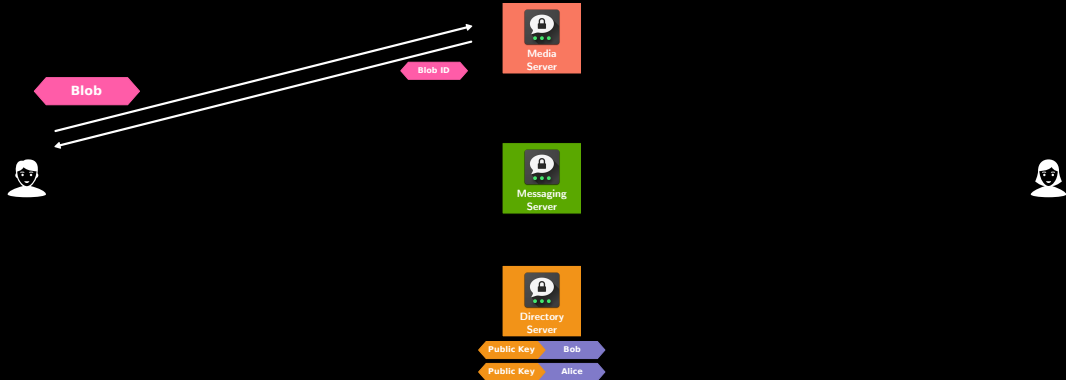


Public Key Bob

Public Key Alice

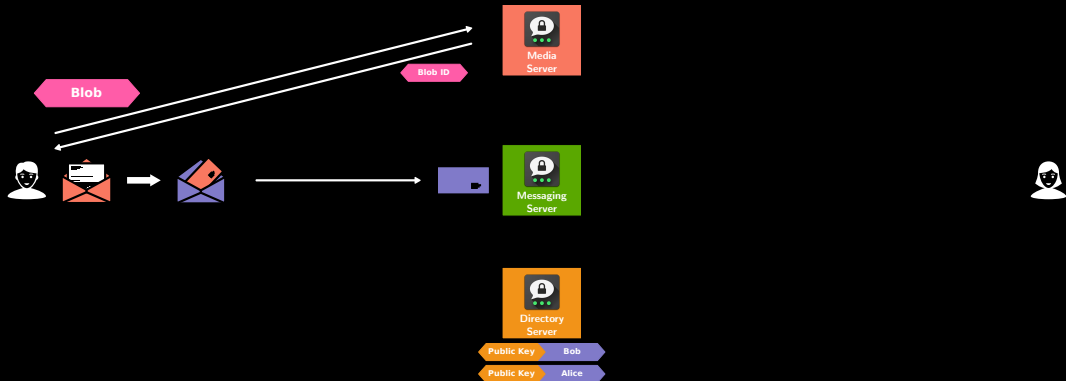


# Sending an Image Message

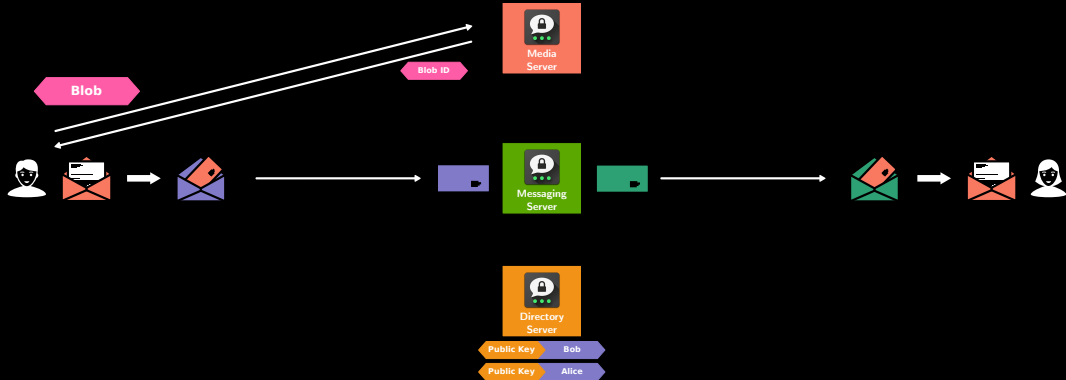




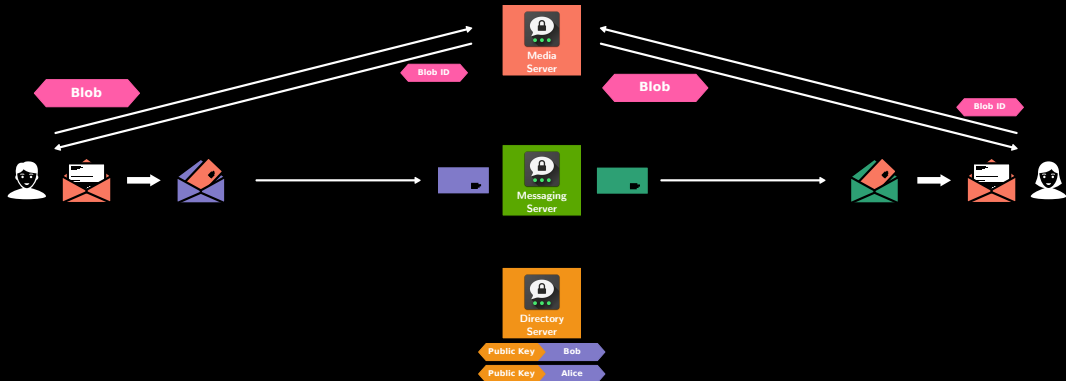
# Sending an Image Message



# Sending an Image Message



# Sending an Image Message

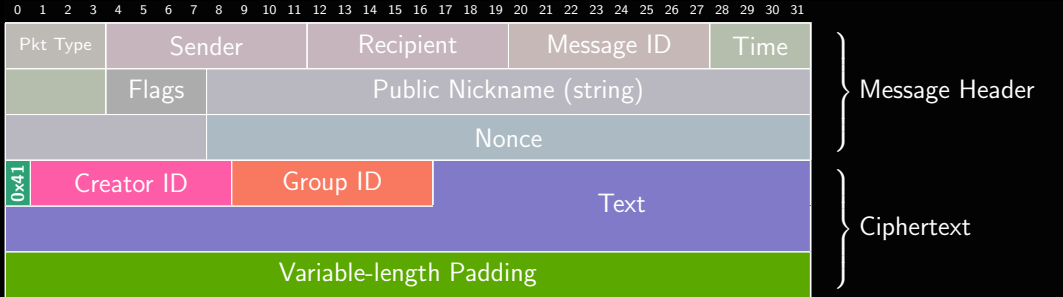


## Recap

---

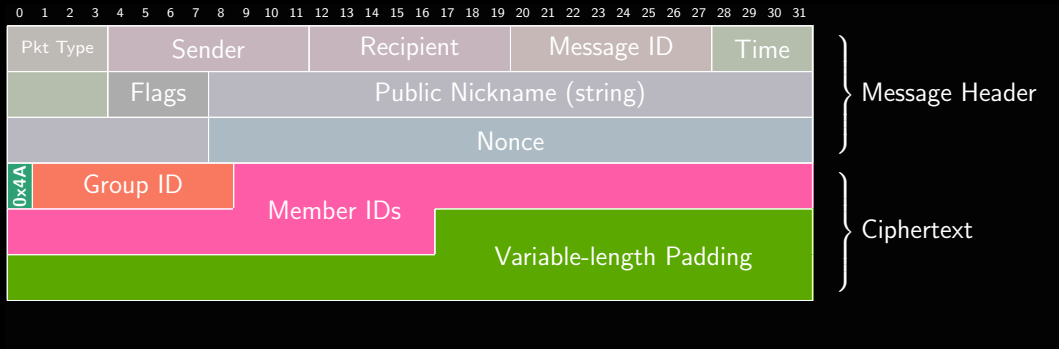
Basic messaging functionality achieved.

# Group Messages



# Group Management Messages

## Group creation message



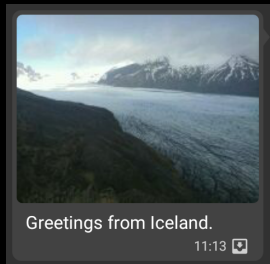
# Group Management Messages

## Group rename message



# Implementation of Addon Features

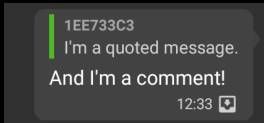
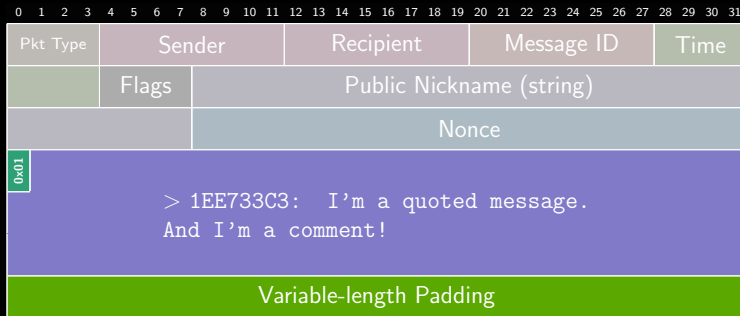
## Captions in Image Messages





# Implementation of Addon Features

## Quoted Messages



# The Devil's in the Detail

---

Sammlung kleinerer Dinge, die uns aufgefallen sind


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


Our reverse-engineering efforts led to a re-implementation of Threema's API.

Done!


# Thank You!


Roland Schilling

 [schilling@tuhh.de](mailto:schilling@tuhh.de)

 [@NerdingByDoing](https://twitter.com/NerdingByDoing)

Frieder Steinmetz

 [frieder.steinmetz@tuhh.de](mailto:frieder.steinmetz@tuhh.de)

 [@twillnix](https://twitter.com/atwillnix)

---

Beamer Theme: [Metropolis](#) by Matthias vorgelsang

Color Theme: [Owl](#) by Ross Churchley

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](#)

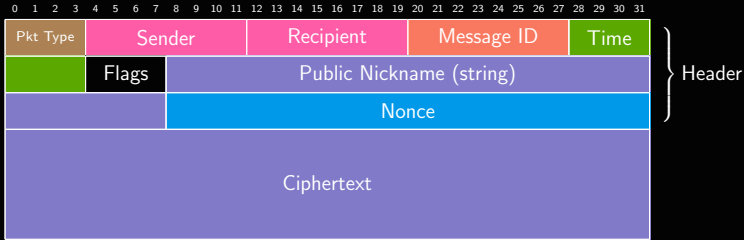
Threema Screenshots taken from the [Threema press package](#)

Thanks to [Jan Ahrens](#) and [Philipp Berger](#) – their work has made ours somewhat easier

Thanks to [Maximilian Köstler](#) for his initial work on Threema

# Appendix

## Message Packet (Threema Protocol Layer)



- Only the MSB of *Flags* is used

## Message Packet on the Wire

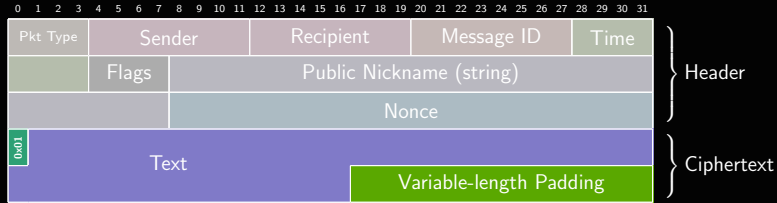
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

Length

Threema Client-to-Server Ciphertext

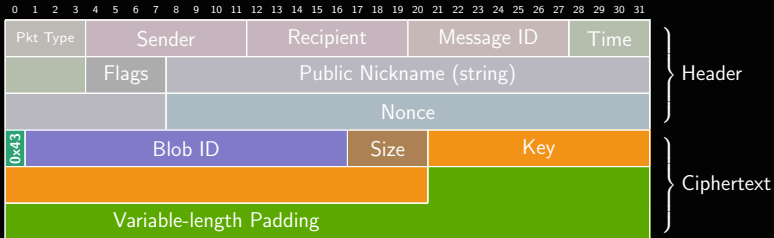
# Appendix

## Text Message



# Appendix

## Image Message

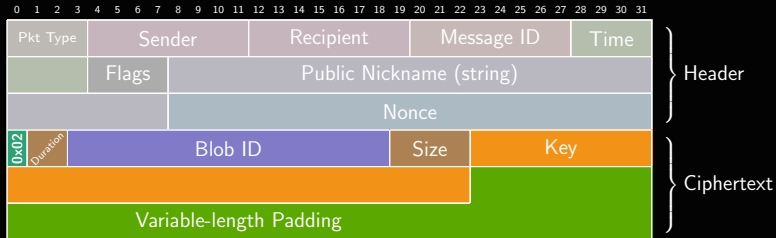


- Blob is symmetrically encrypted using *Key* and uploaded to asset server.
- Image captions are stored inside the image's EXIF data. These data leak upon creating such an image while the "save media to gallery" option is enabled.



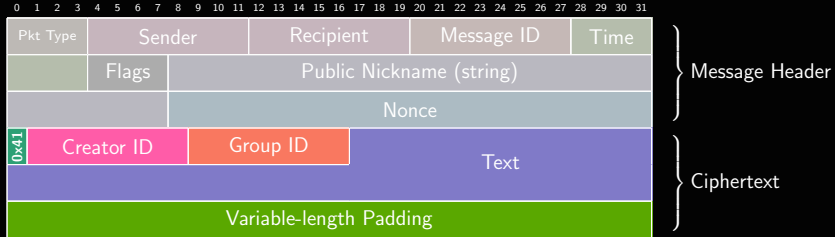
# Appendix

## Audio Message



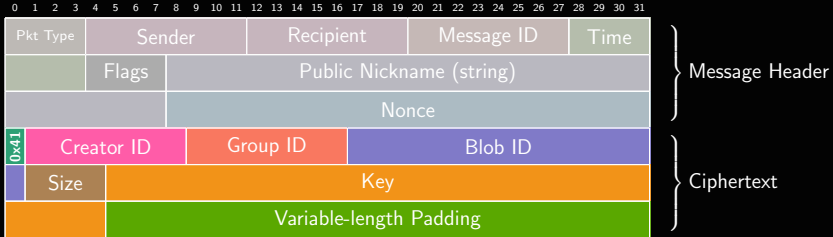
# Appendix

## Group Message Packet



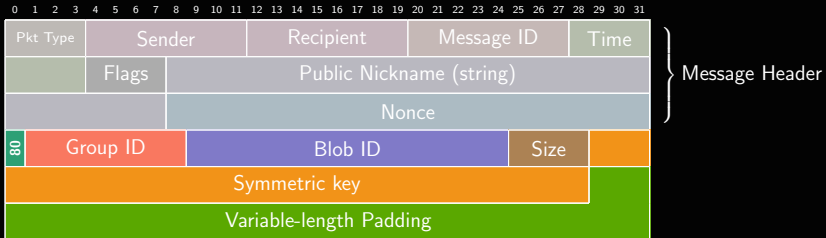
# Appendix

## Group Image Message

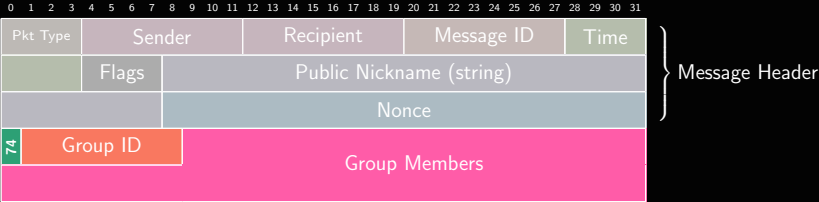


# Appendix

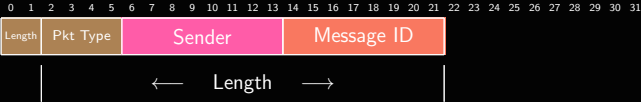
## Group Picture Update



# Create/Update Group (members)



## Acknowledgement Packet to Server



# Appendix

## 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

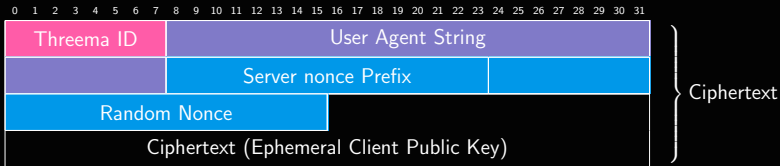
Ephemeral Server Public Key

Client Nonce Prefix

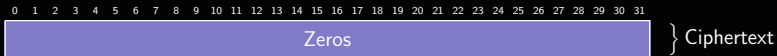
Ciphertext

# Appendix

## Client Authentication Packet



## Server Acknowledgement



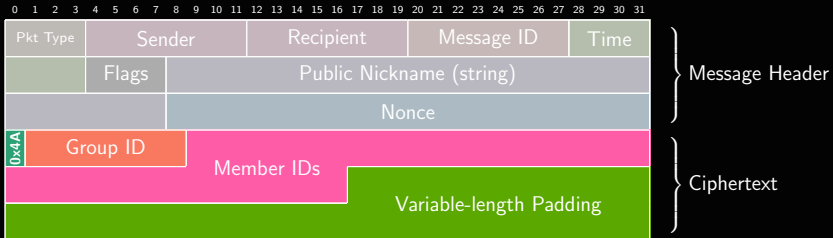


## PKCS7 Padding

												03	03	03
										04	04	04	04	04
								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	06

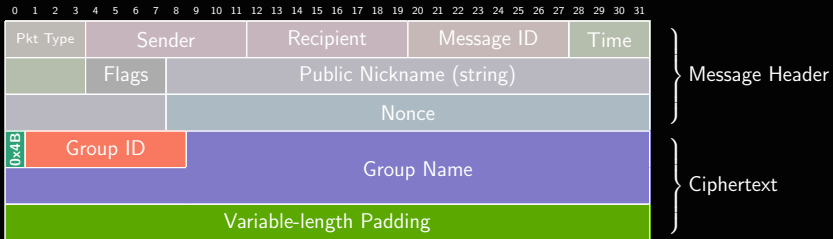
# Appendix

## Group Management Message - Add Users



# Appendix

## Group Management Message - Rename Group



# Appendix

## Quoted Text Message

