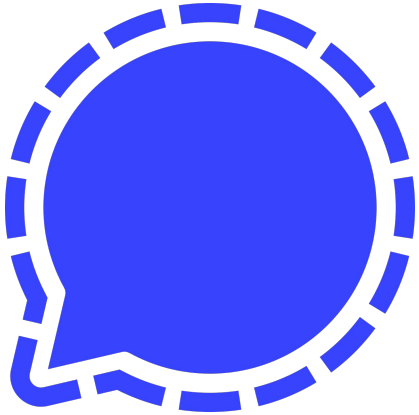
Message Injection Attacks Against Signal

Work published at USENIX Security 2026

Kien Tuong Truong[1], **Noemi Terzo**[2], Kenny Paterson[1]

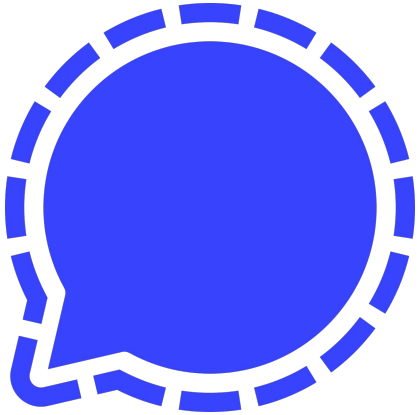
[1] ETH Zurich
[2] Max Planck Institute for Security and Privacy

What’s Signal?



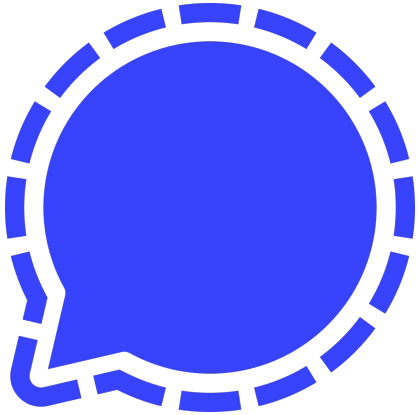
What’s Signal?

“The messaging app”



What's Signal?

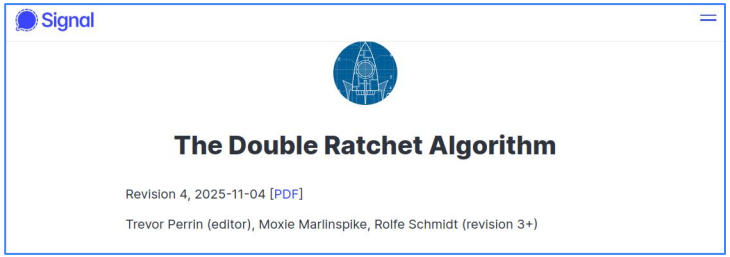
"The messaging app"

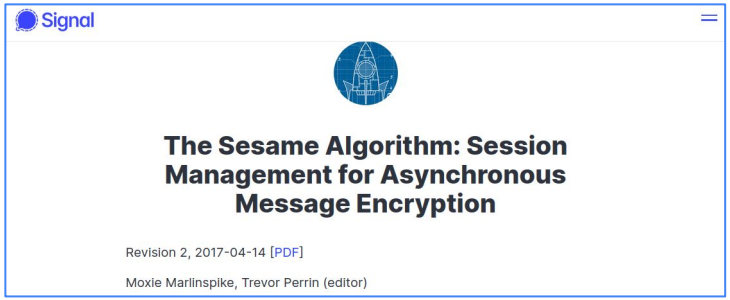


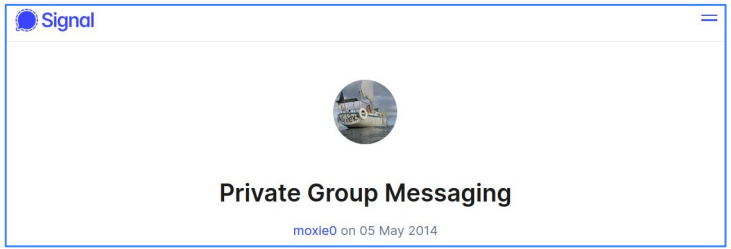
"The set of protocols"

What's Signal?

"The set of protocols"







What's Signal?

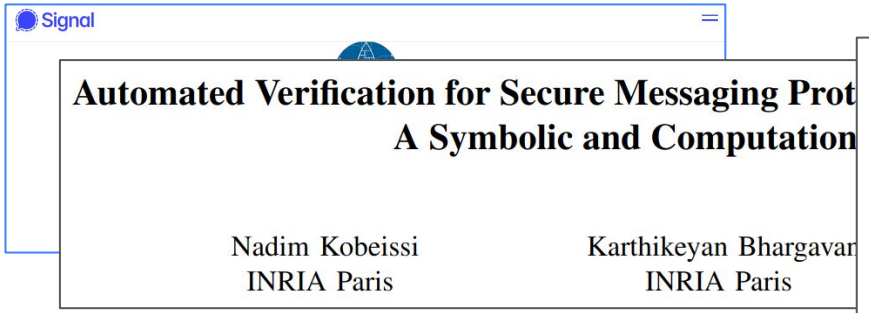
"The set of protocols"


Signal

The PQXDH Key Agreement Protocol

Revision 3, 2023-05-24, Last Updated: 2024-01-23 [PDF]

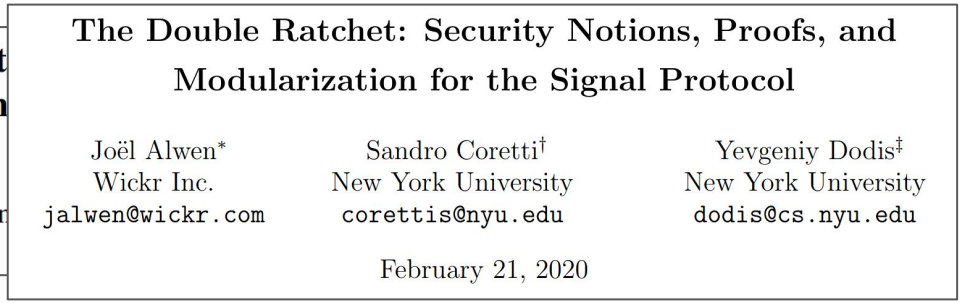
Ehren Kret, Rolfe Schmidt


Signal

Automated Verification for Secure Messaging Prot
A Symbolic and Computation

Nadim Kobeissi
INRIA Paris

Karthikeyan Bhargavan
INRIA Paris

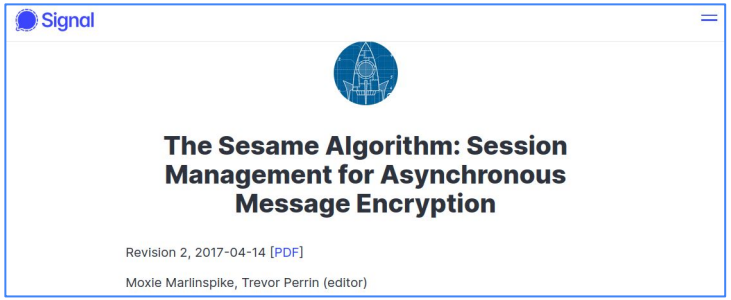

The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol

Joël Alwen*
Wickr Inc.
jalwen@wickr.com

Sandro Coretti†
New York University
corettis@nyu.edu

Yevgeniy Dodis‡
New York University
dodis@cs.nyu.edu

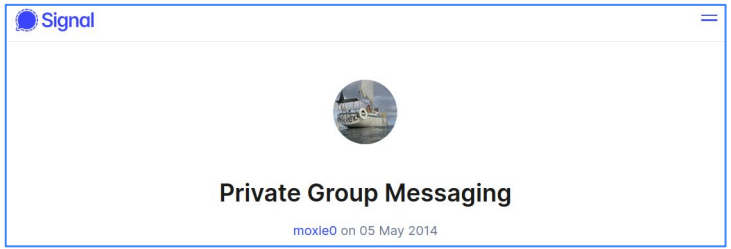
February 21, 2020


Signal

The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [PDF]

Moxie Marlinspike, Trevor Perrin (editor)

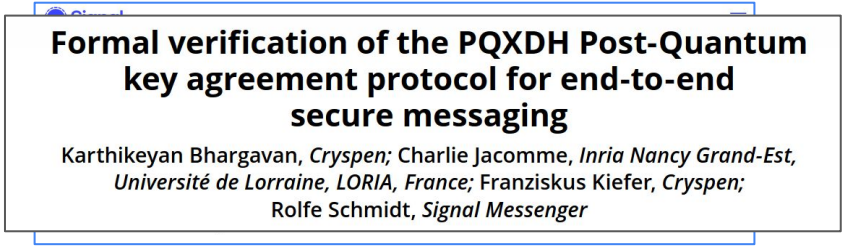

Signal

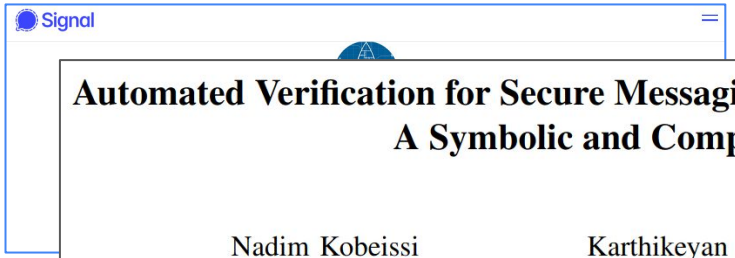
Private Group Messaging

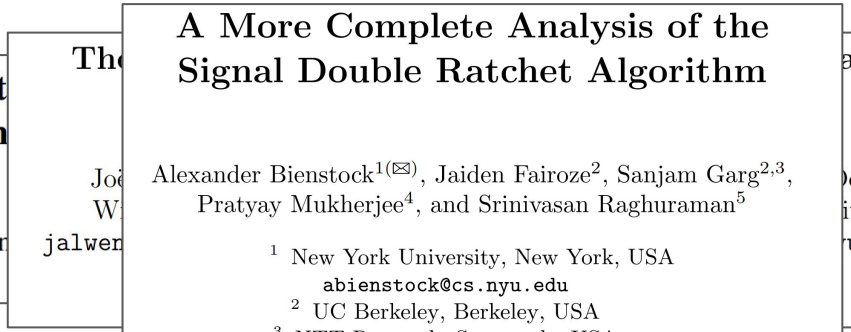
moxie0 on 05 May 2014

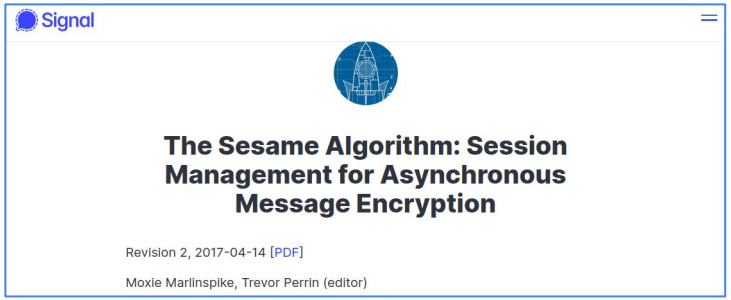
What's Signal?

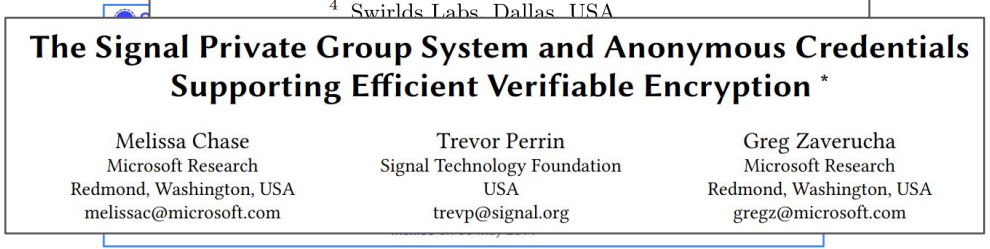
"The set of protocols"









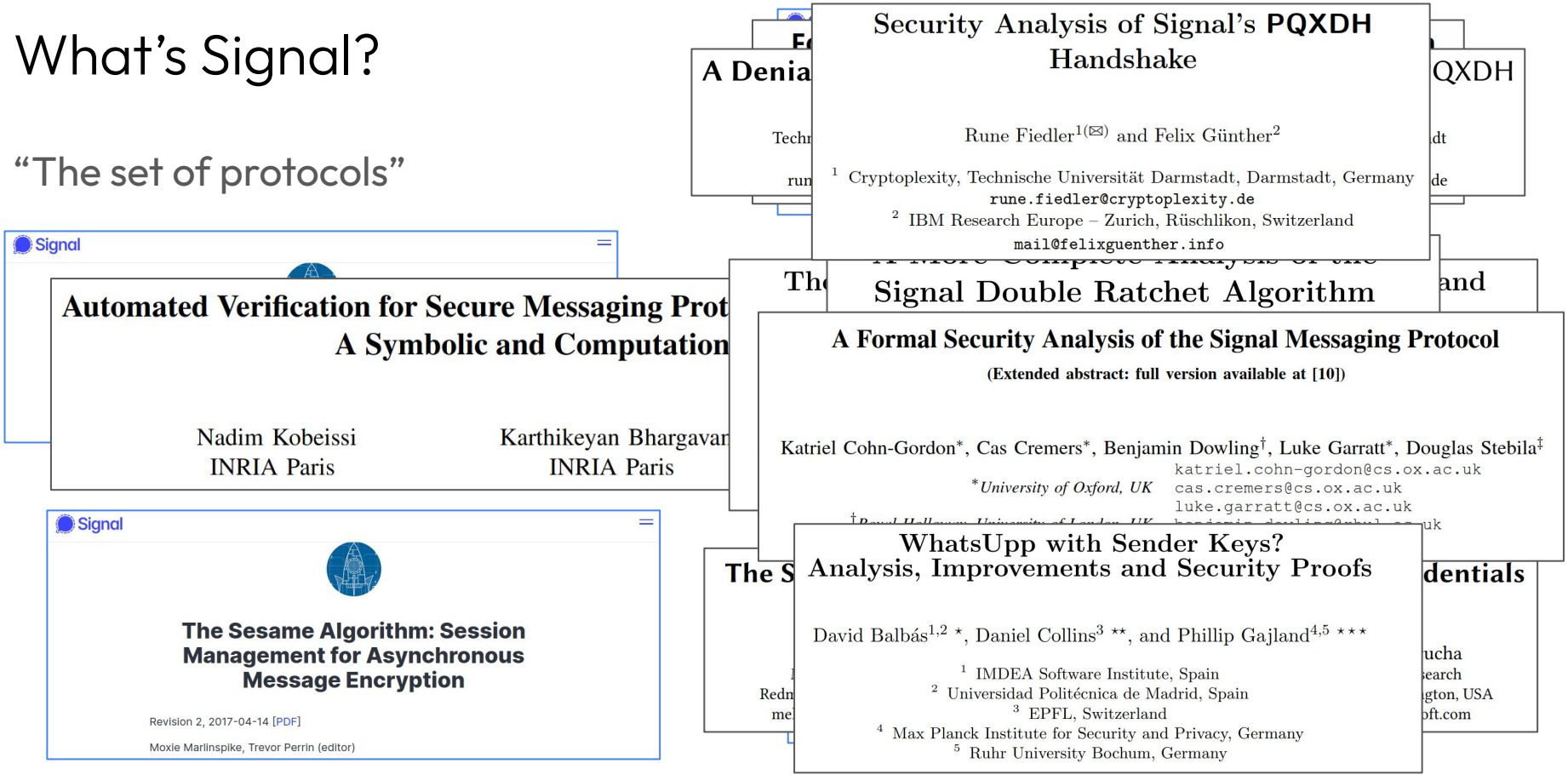


What's Signal?

"The set of protocols"

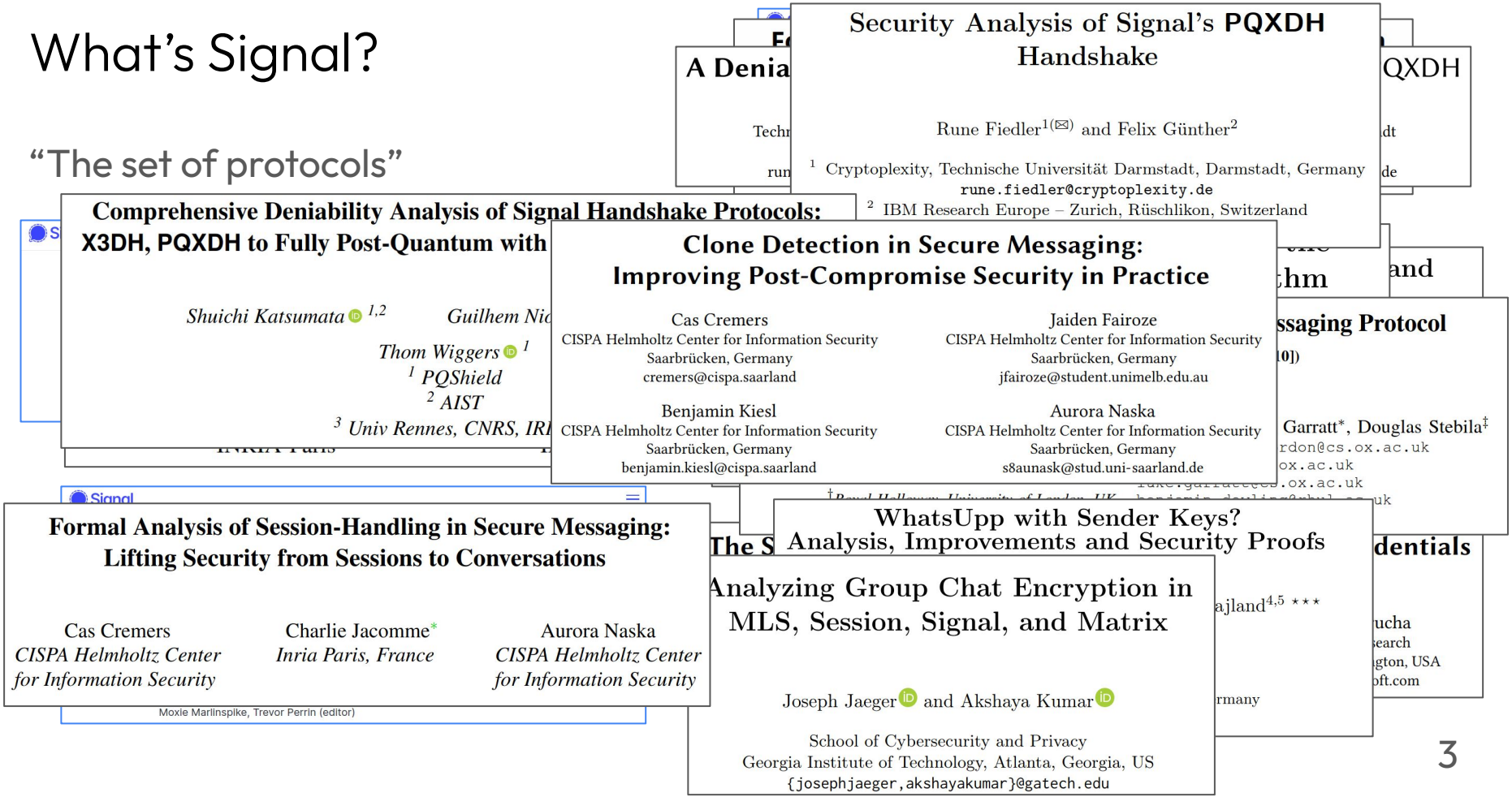
What's Signal?

"The set of protocols"

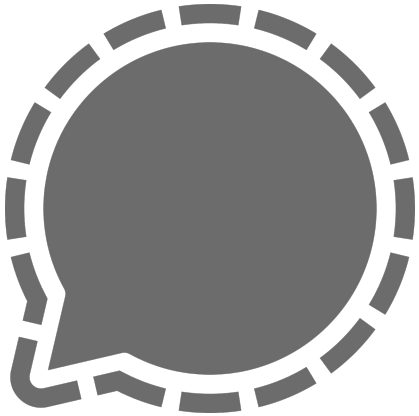


What's Signal?

"The set of protocols"

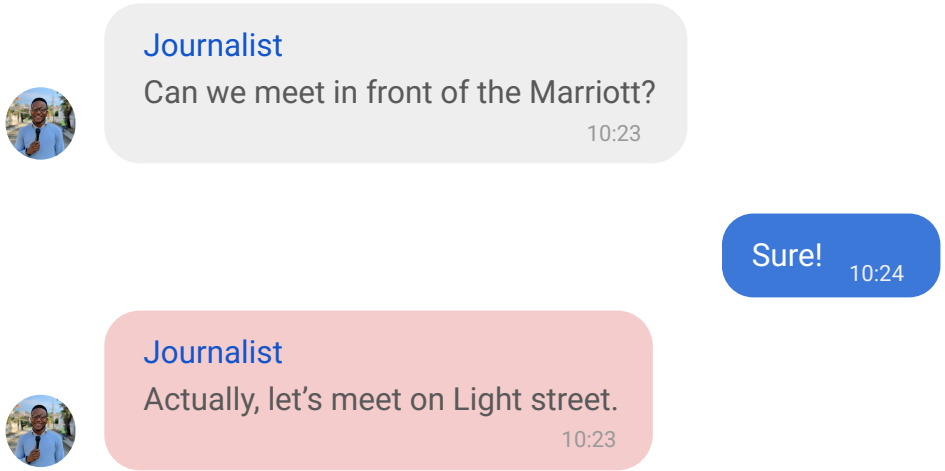


What about the app?



Our Contributions

Two attacks that allow a malicious server to inject arbitrary messages in a conversation, undetected

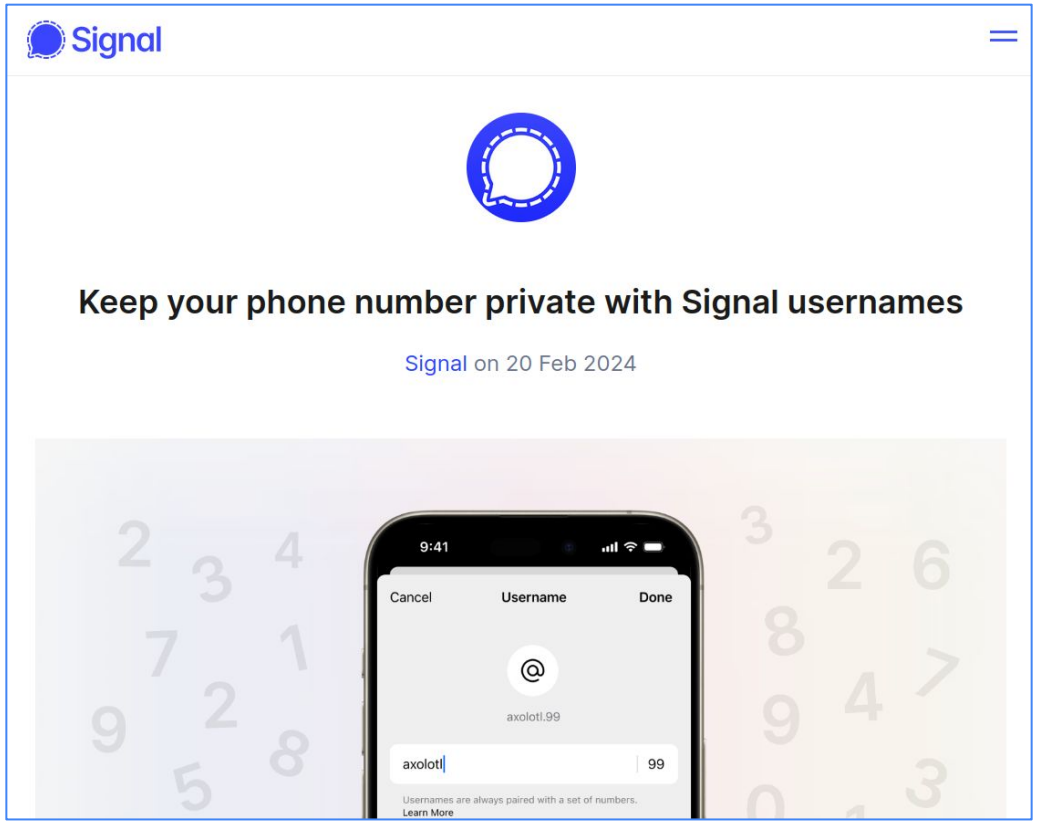


and

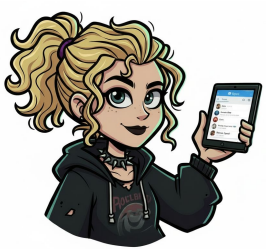
Detailed descriptions of protocols that were never analyzed and that affect the security of conversations in Signal.

Attack 1: Usernames and Session Handling

Introduction of usernames in Signal



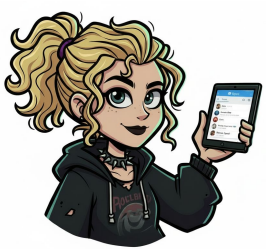
Phone numbers vs usernames

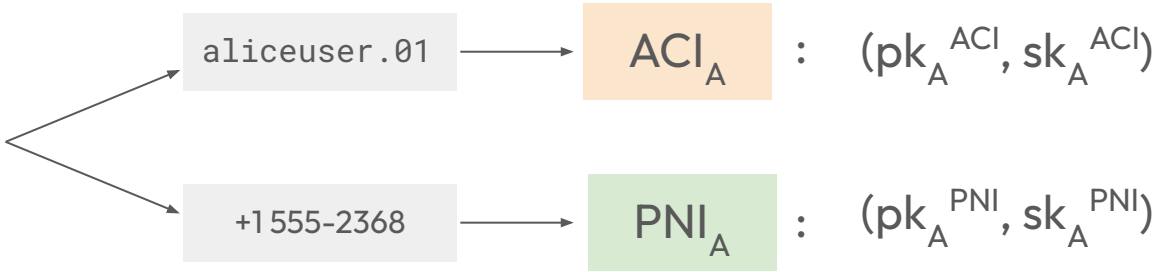


+1 555-2368 → PNI_A : (pk_A^{PNI}, sk_A^{PNI})

PNI: Phone Number Identifier

Phone numbers vs usernames

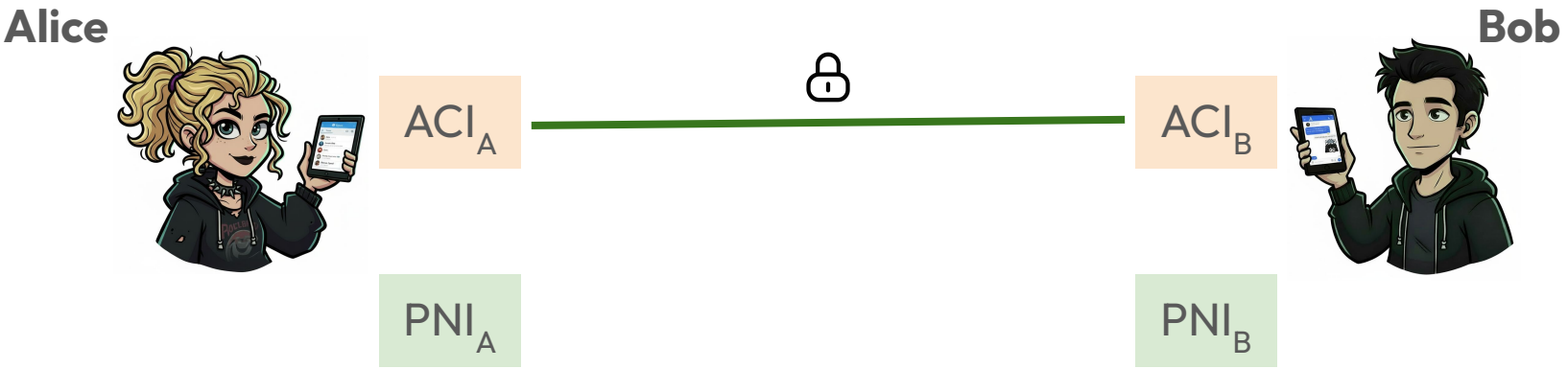




PNI: Phone Number Identifier

ACI: Account Identifier

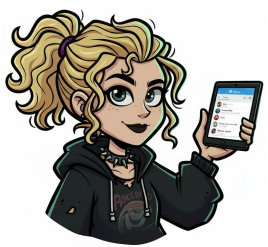
ACIs, PNIs and sessions



ACI_A — ACI_B session: by username

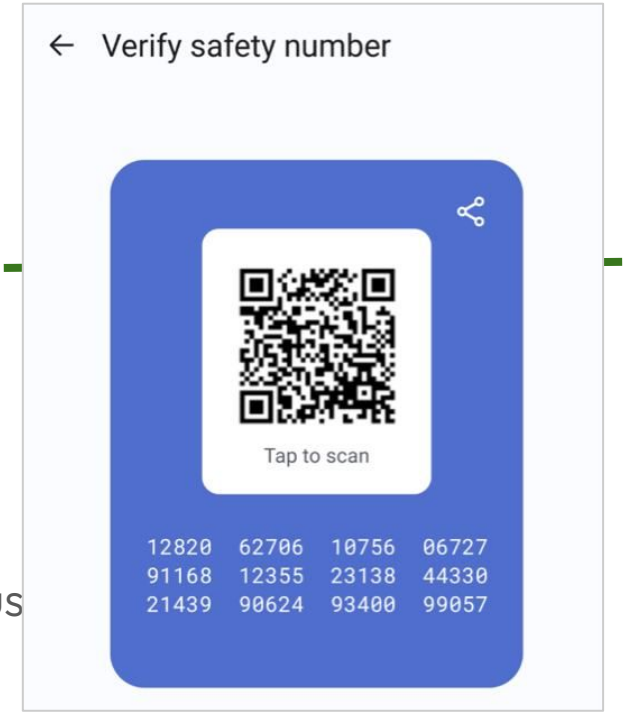
ACIs, PNIs and sessions

Alice



ACI_A

PNI_A



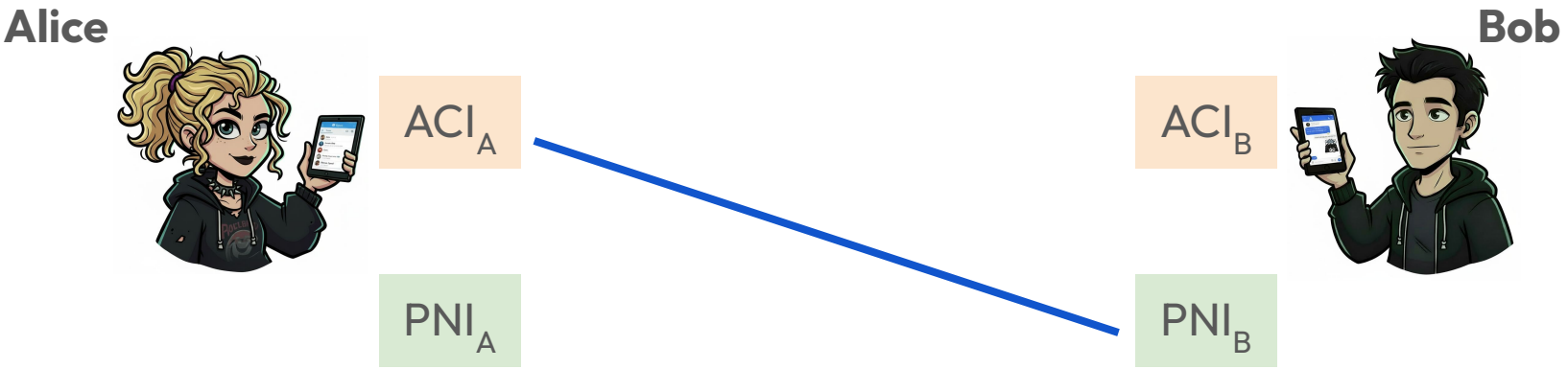
ACI_B

PNI_B

Bob

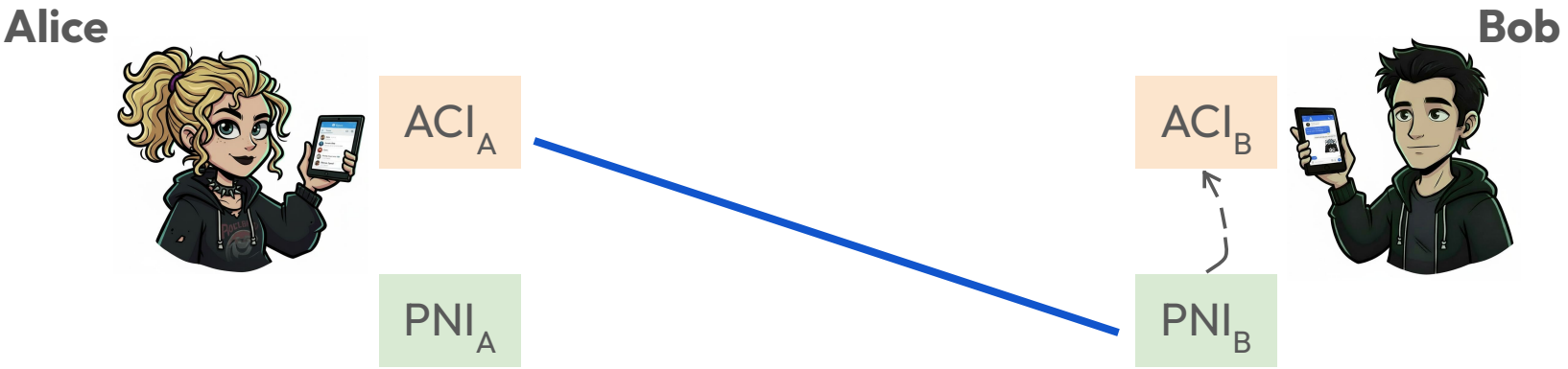
ACI_A — ACI_B session: by us

ACIs, PNIs and sessions



ACI_A — PNI_B session: by phone number

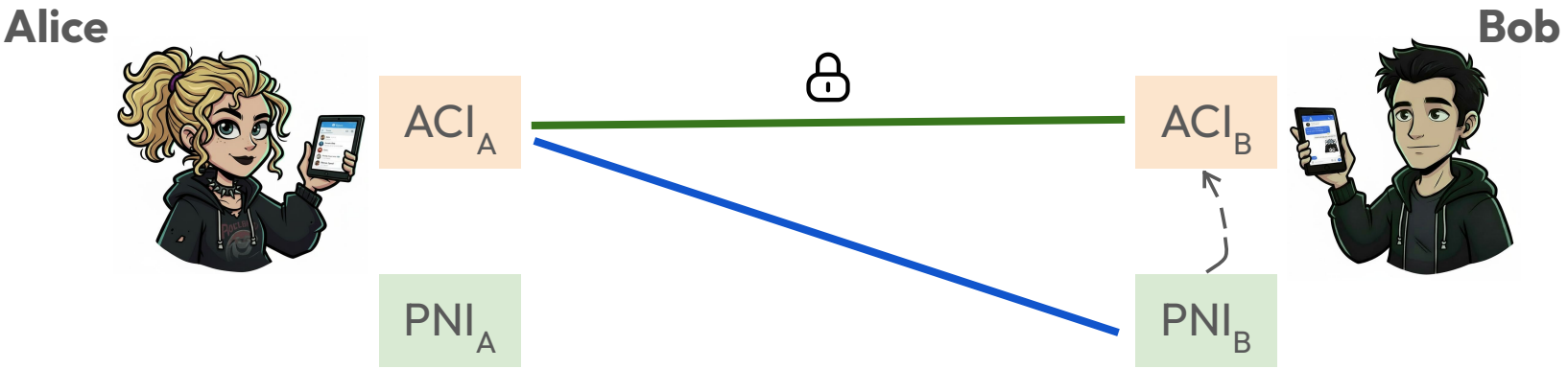
ACIs, PNIs and sessions



ACI$_A$ — PNI$_B$ session: by phone number

upgrade to ACI...

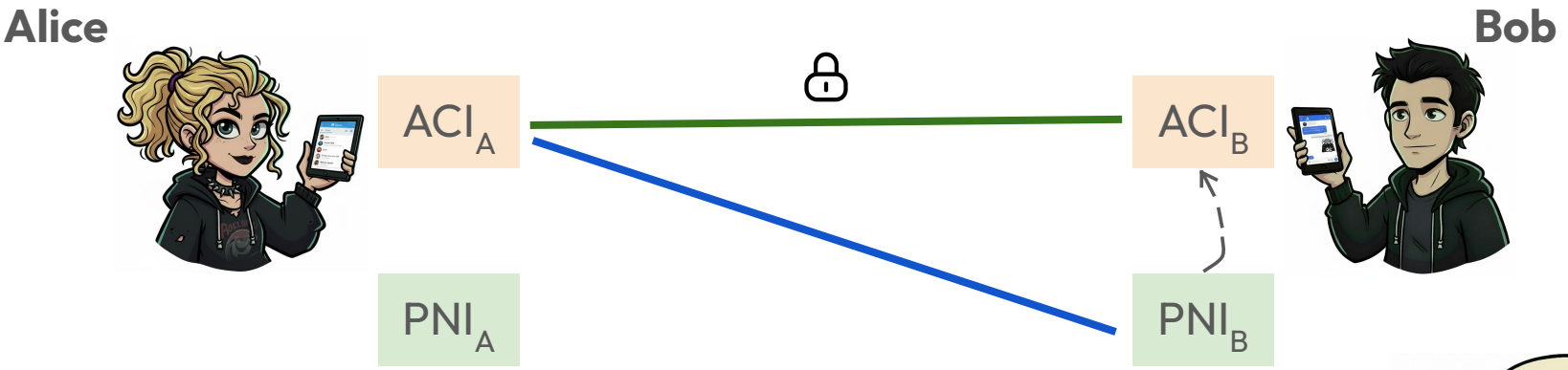
ACIs, PNIs and sessions



ACI_A — PNI_B session: by phone number

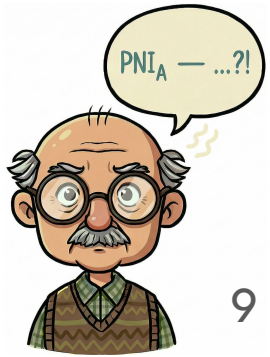
upgrade to ACI... new ACI_A — ACI_B session!

ACIs, PNIs and sessions

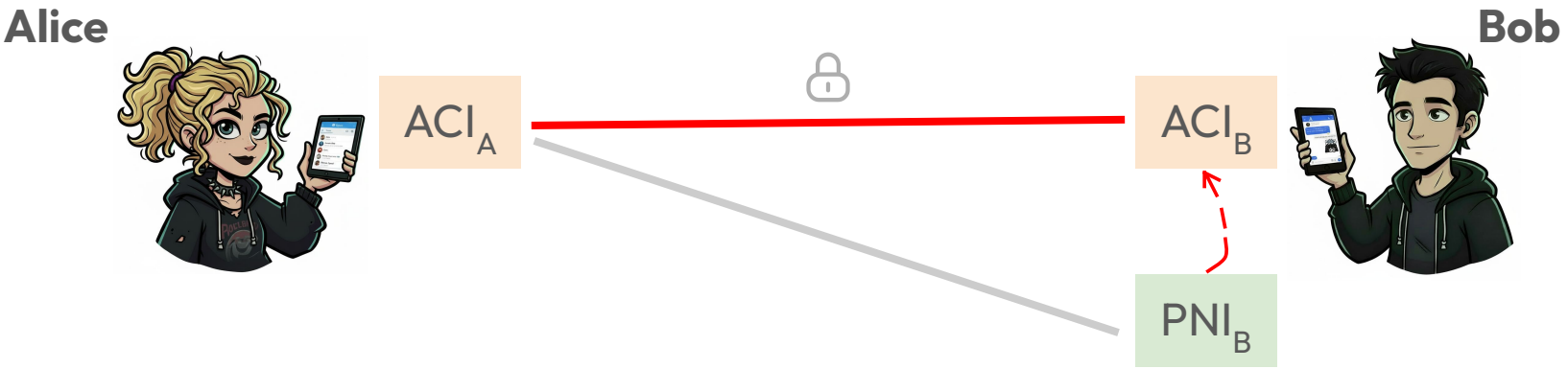


ACI$_A$ — PNI$_B$ session: by phone number

upgrade to ACI... new ACI$_A$ — ACI$_B$ session!



ACIs, PNIs and sessions

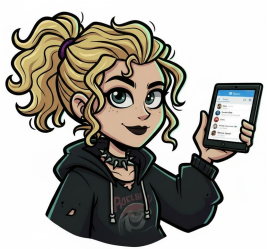


ACI$_A$ — PNI$_B$ session: by phone number

upgrade to ACI... new ACI$_A$ — ACI$_B$ session!

“PNI-to-ACI” protocol

Alice



ACI_A

Bob

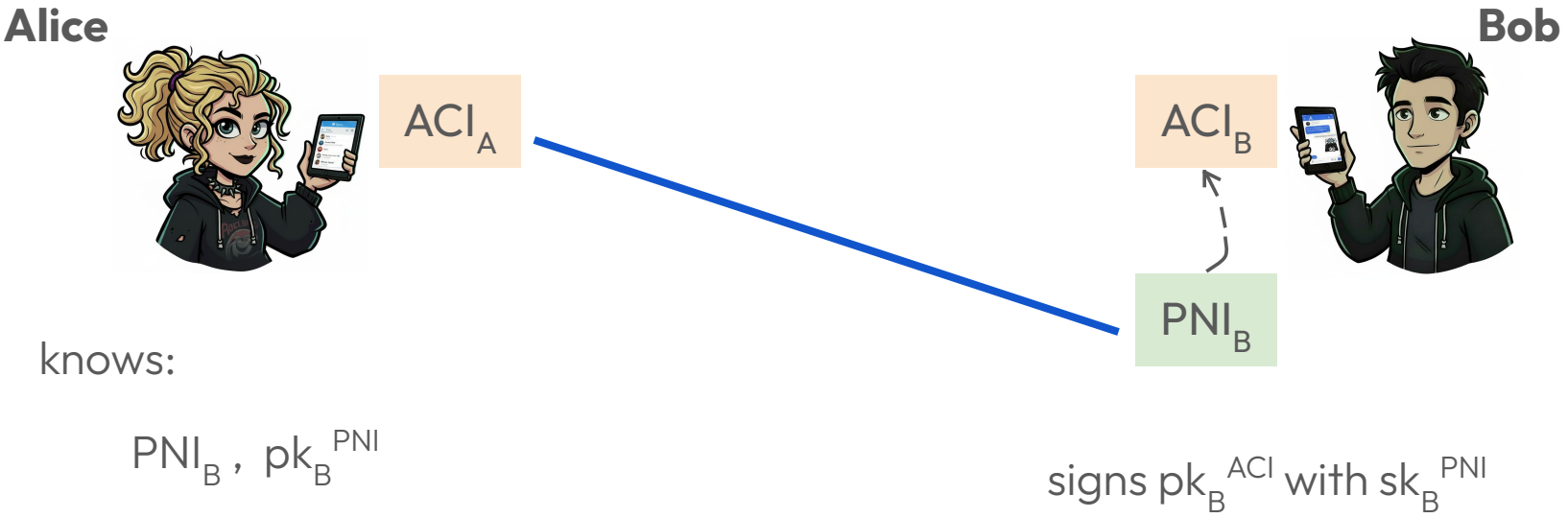
ACI_B

PNI_B

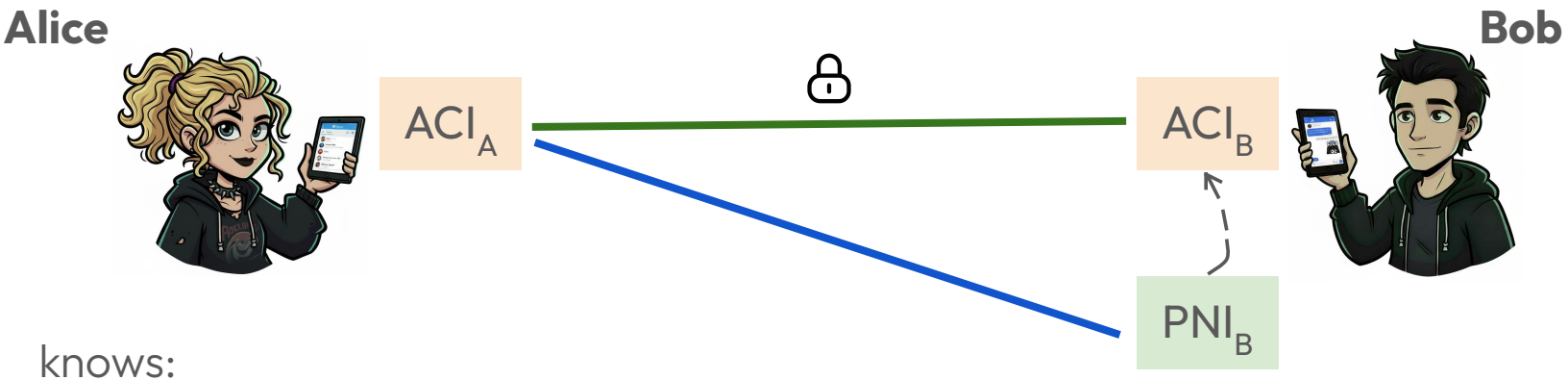
knows:

PNI_B , pk_B^{PNI}

“PNI-to-ACI” protocol



“PNI-to-ACI” protocol



knows:

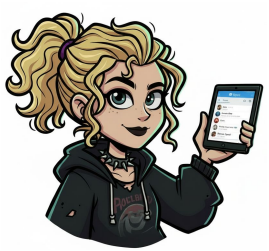
PNI_B , pk_B^{PNI}

merges ACI_B, pk_B^{ACI}, PNI_B, pk_B^{PNI}

signs pk_B^{ACI} with sk_B^{PNI}

Involved protocols: X3DH, Double Ratchet, Sesame

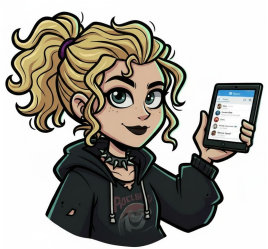
“PNI-to-ACI” protocol

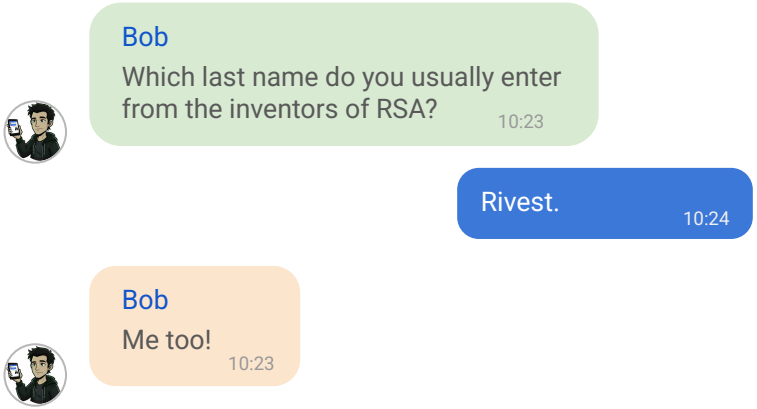


…	…	…	…	…	…
+1 555-1259	bobsignal.03	PNI_B	pk_B^{PNI}	ACI_B	pk_B^{ACI}
NULL	charlie.42	NULL	NULL	ACI_C	pk_C^{ACI}
…	…	…	…	…	…

The two identities are not cryptographically linked:
they are merged into the same database row.

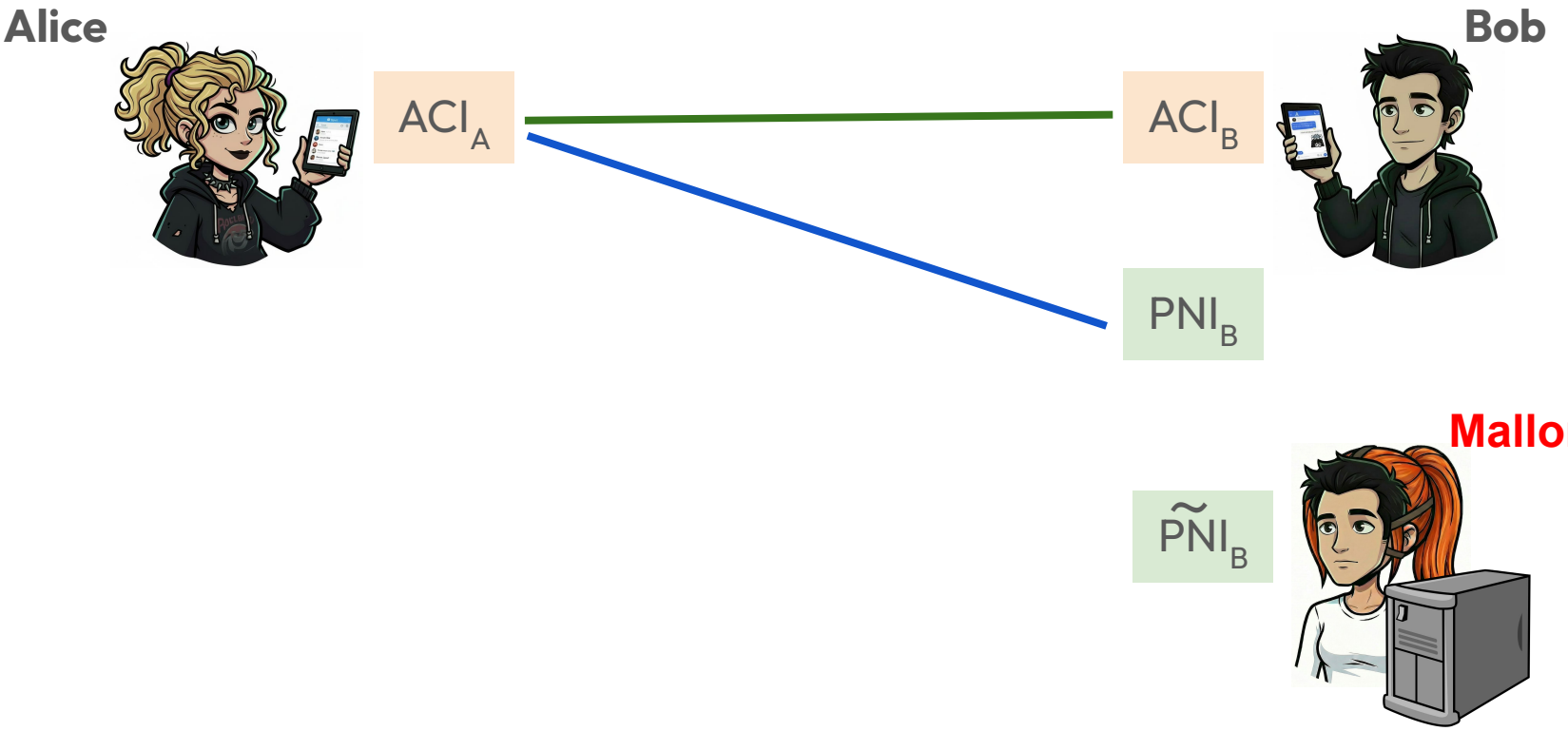
“PNI-to-ACI” protocol



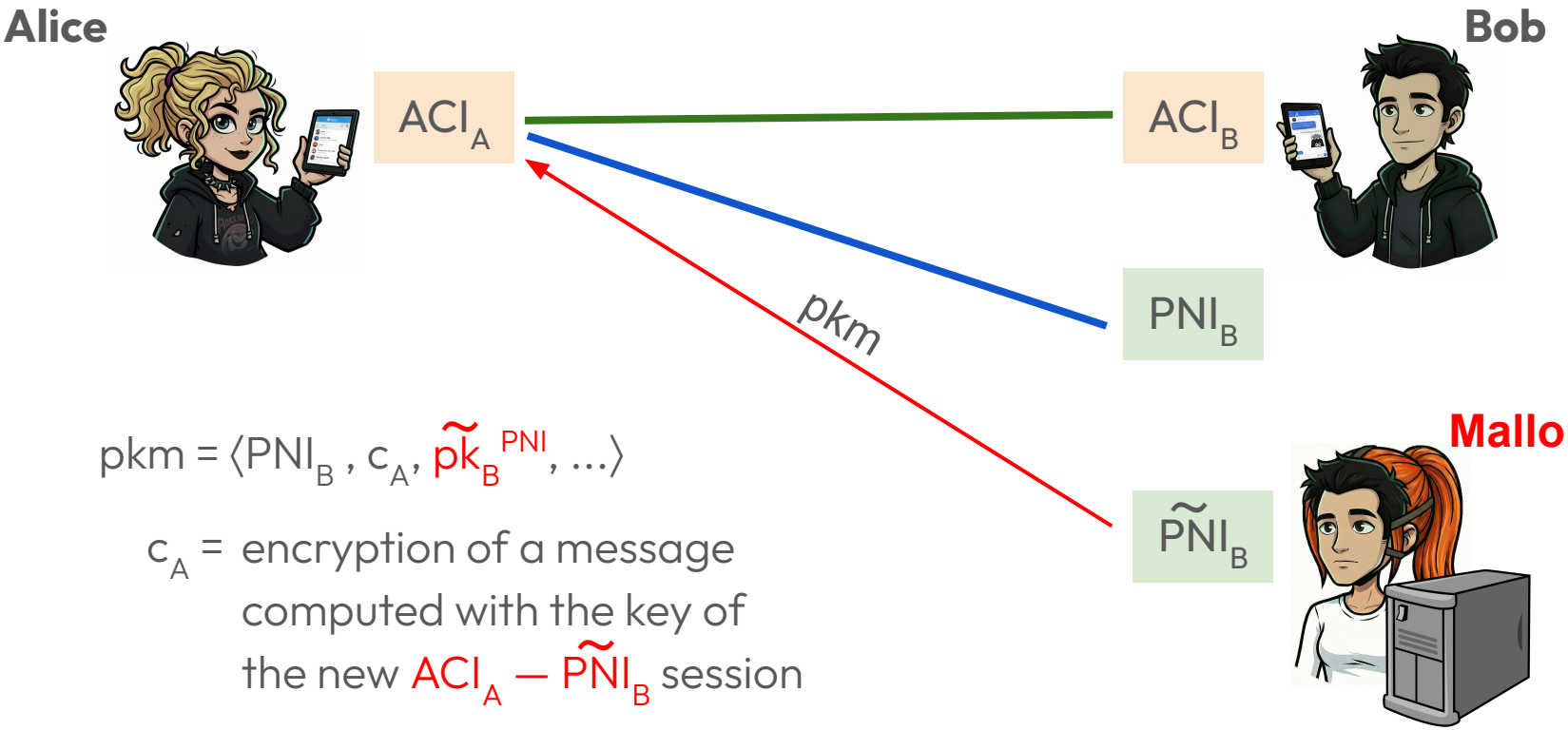


After linking, any message with PNI_B or ACI_B as a source will be **displayed in the same conversation.**

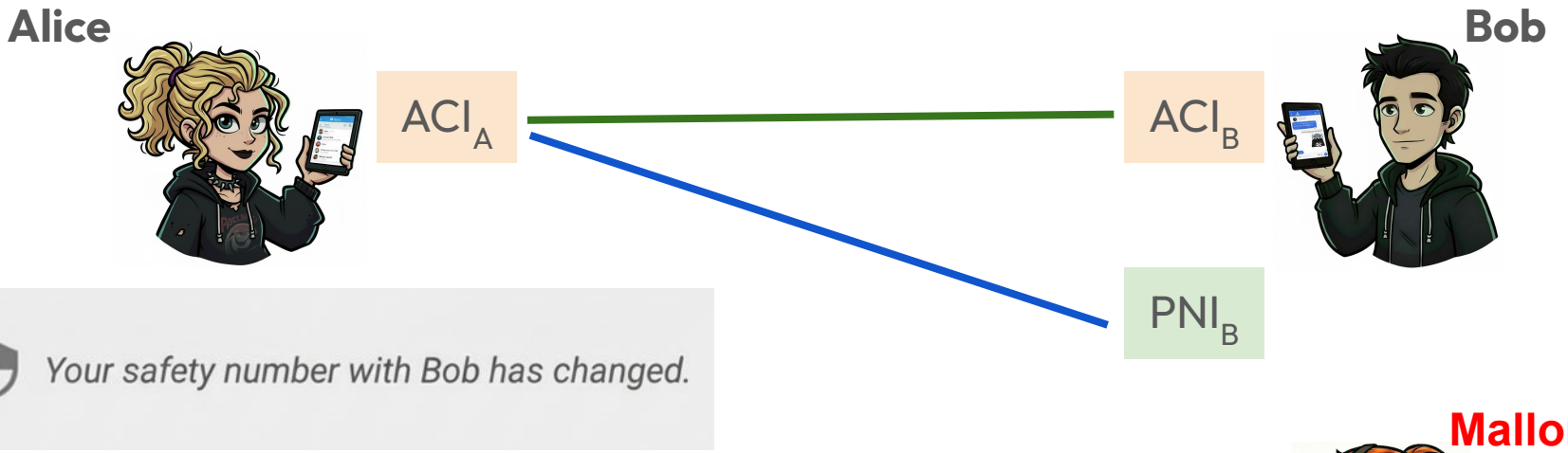
The First Attack



The First Attack

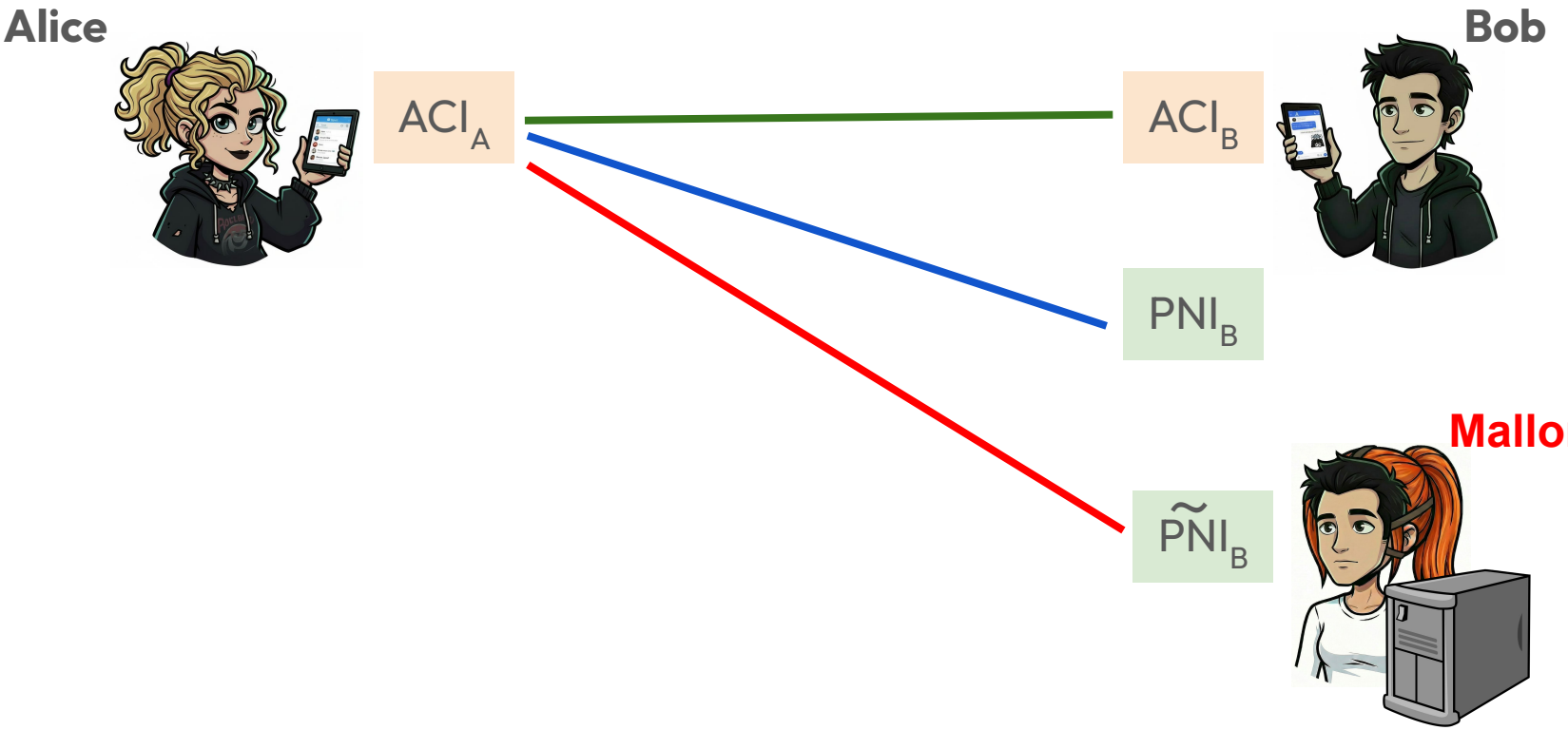


The First Attack

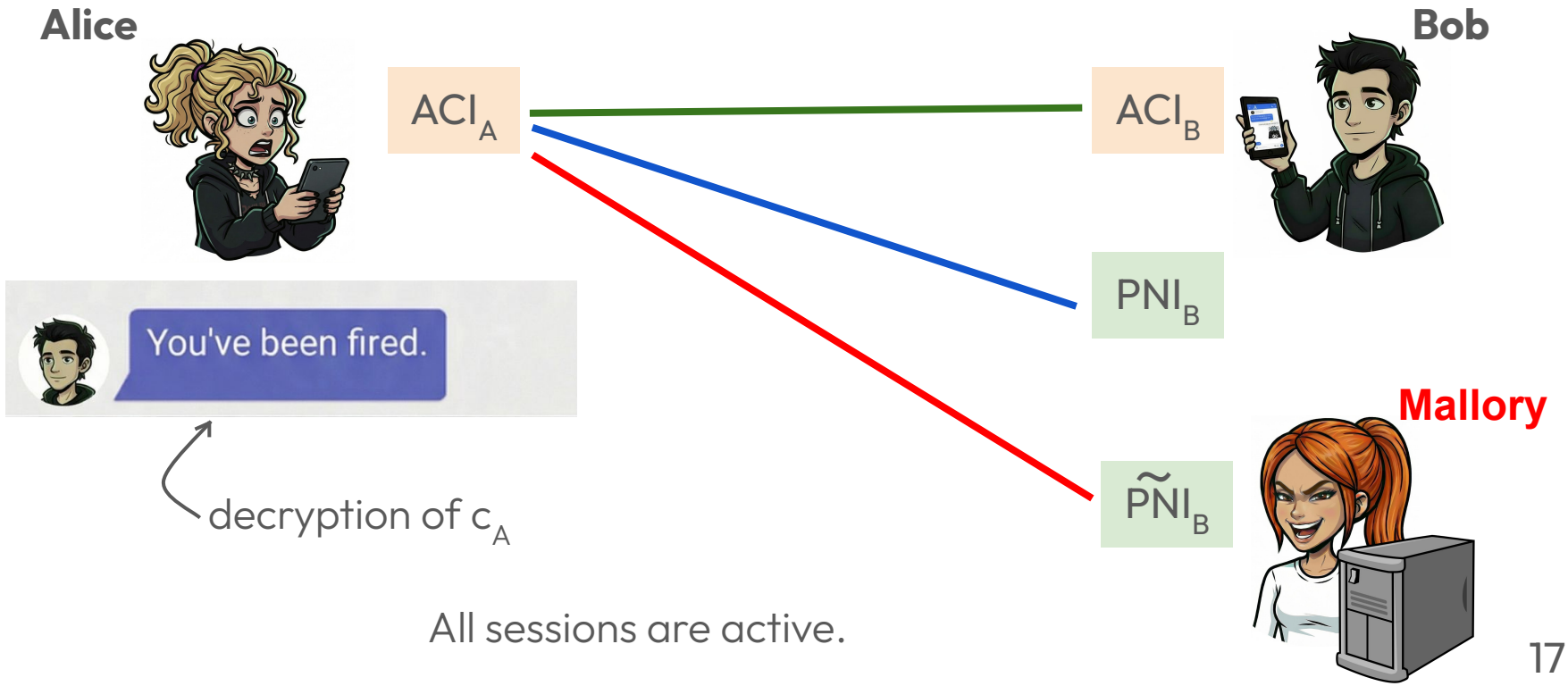


If Alice and Bob check the safety numbers out-of-band, they are the same.

The First Attack



The First Attack



More about the attack

More about the attack

- The attack can be repeated injecting new messages
  - No “safety number changed” notification anymore

More about the attack

- The attack can be repeated injecting new messages
  - No “safety number changed” notification anymore
- The attack can be bidirectional
  - Bob has to accept a message request from an unknown user and send a message

More about the attack

- The attack can be repeated injecting new messages
  - No “safety number changed” notification anymore
- The attack can be bidirectional
  - Bob has to accept a message request from an unknown user and send a message
- Affected platforms
  - Android → up to v7.57.2
  - Desktop → up to v7.73.0

More about the attack

- The attack can be repeated injecting new messages
  - No “safety number changed” notification anymore
- The attack can be bidirectional
  - Bob has to accept a message request from an unknown user and send a message
- Affected platforms
  - Android → up to v7.57.2
  - Desktop → up to v7.73.0
- Fixed in 1-2 days in September 2025 after responsible disclosure
  - Patch: do not accept messages coming from a PNI

More about the attack

- The attack can be repeated injecting new messages
  - No “safety number changed” notification anymore
- The attack can be bidirectional
  - Bob has to accept a message request from an unknown user and send a message
- Affected platforms
  - Android → up to v7.57.2
  - Desktop → up to v7.73.0
- Fixed in 1-2 days in September 2025 after responsible disclosure
  - Patch: do not accept messages coming from a PNI
- Devastating impact in real scenarios

Attack 2: Sealed Sender and Plaintext Messages

Injecting Messages

In Signal, a malicious server can send a message on behalf of anyone, as the metadata fields are not authenticated...

Injecting Messages

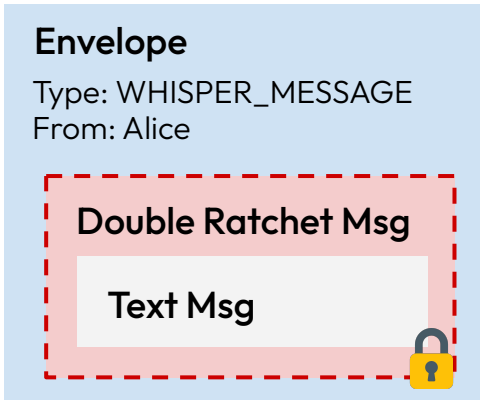
In Signal, a malicious server can send a message on behalf of anyone, as the metadata fields are not authenticated...

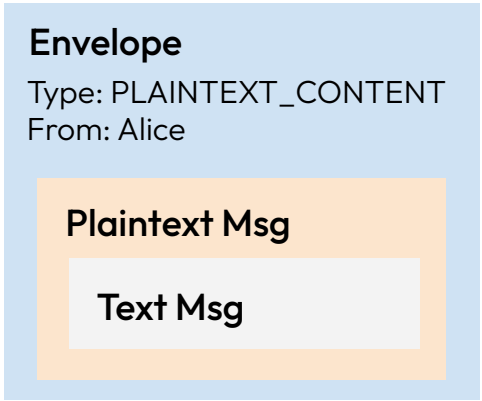
...but it cannot forge valid ciphertexts!

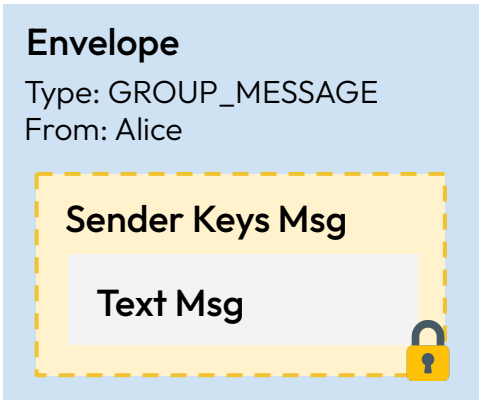
Injecting Messages

Plaintext content is allowed, but only to **encode errors.**

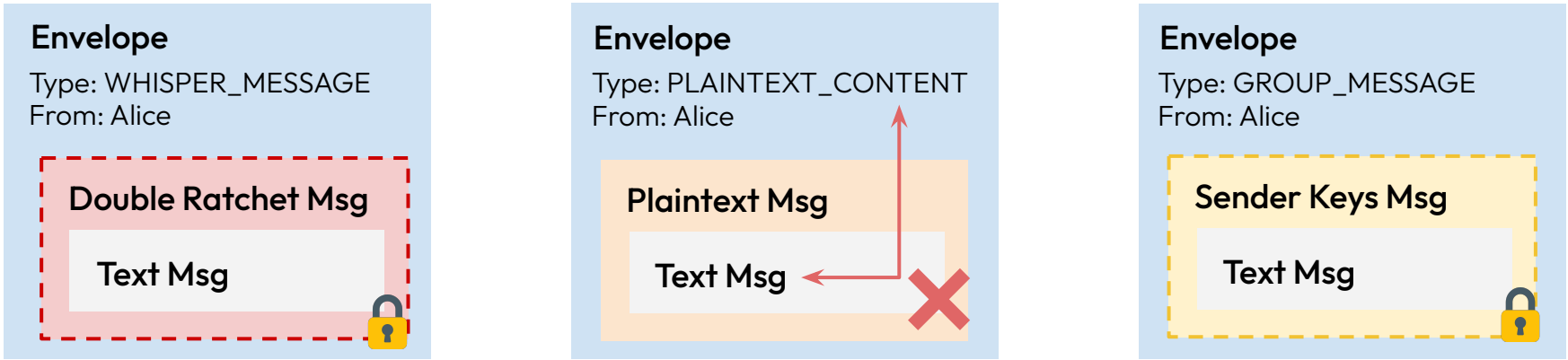
To understand the validation process, we need to look into the message structure.
Standard messages consist of an Envelope, wrapping some Content





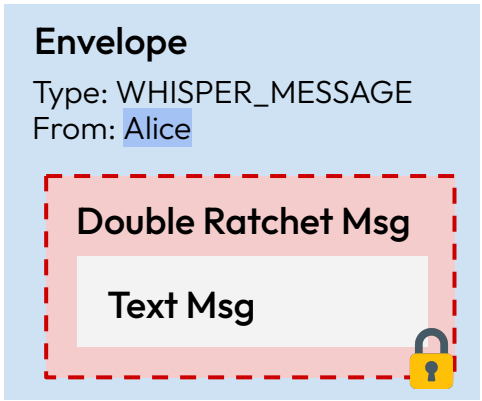


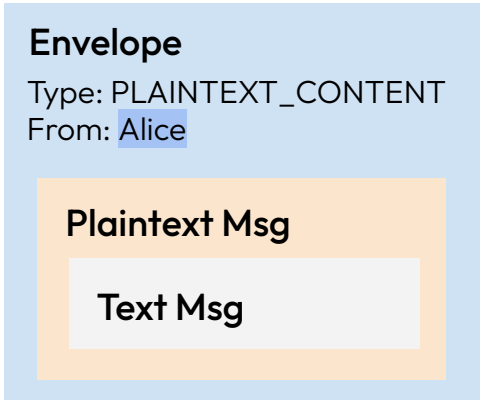
To understand the validation process, we need to look into the message structure.
Standard messages consist of an Envelope, wrapping some Content

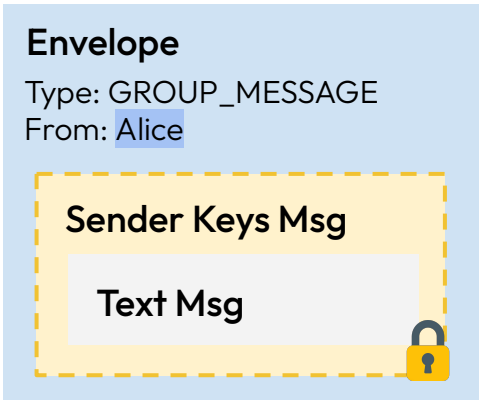


Plaintext messages containing Text messages are not accepted.

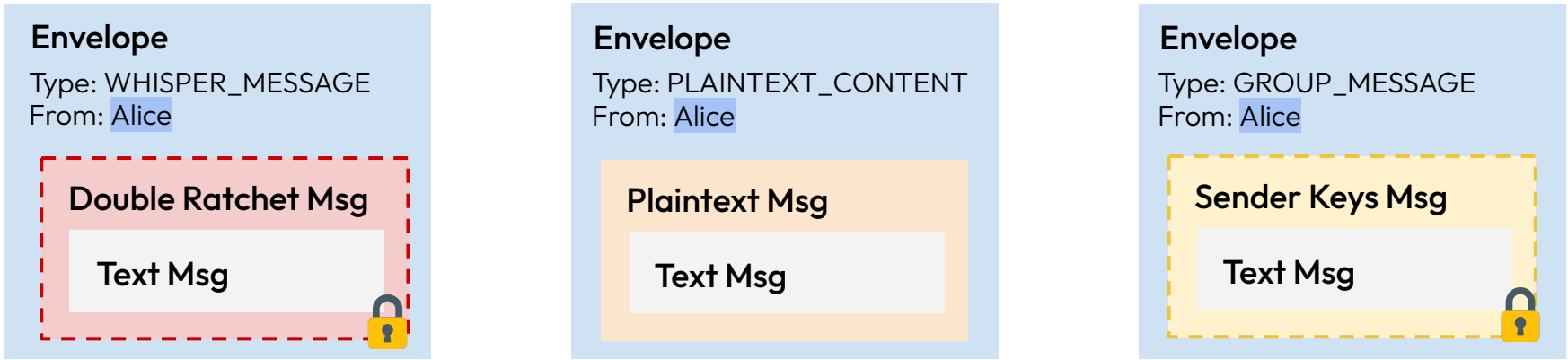
But these messages also leak the sender identity...





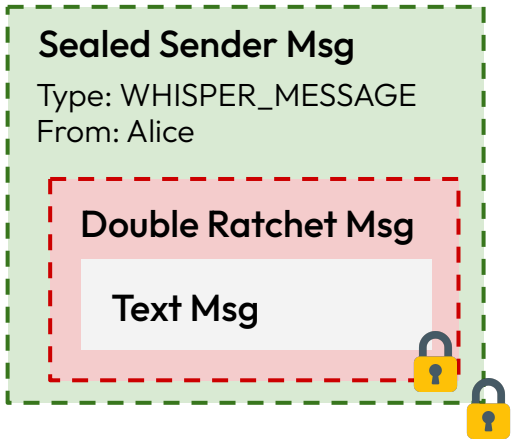


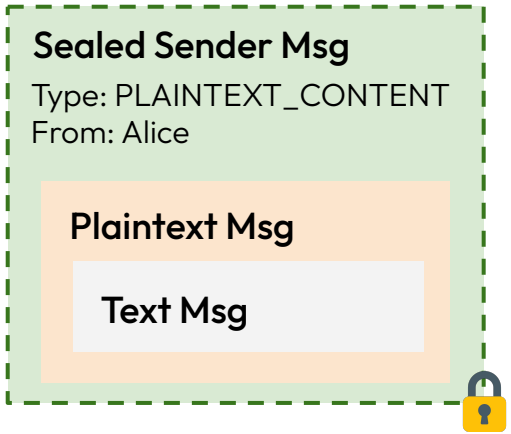
But these messages also leak the sender identity...

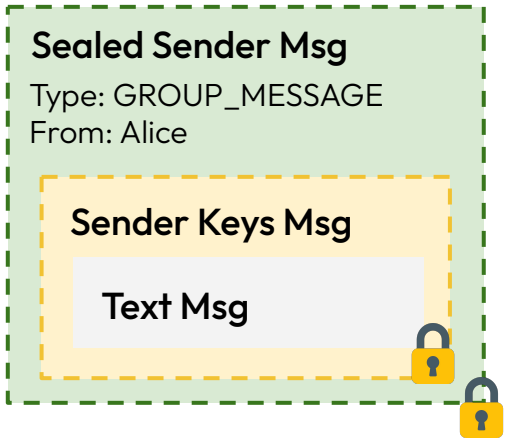


Sealed Sender is a protocol which aims to hide this metadata

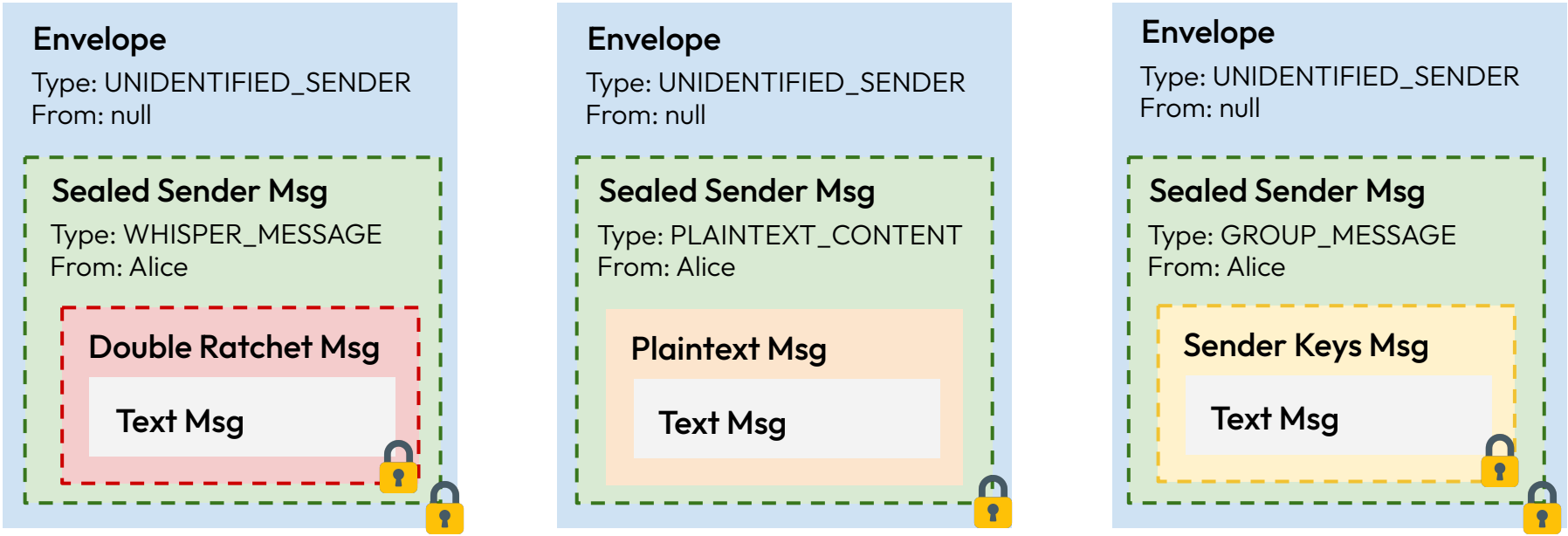
Sealed Sender encrypts and authenticates the envelope...



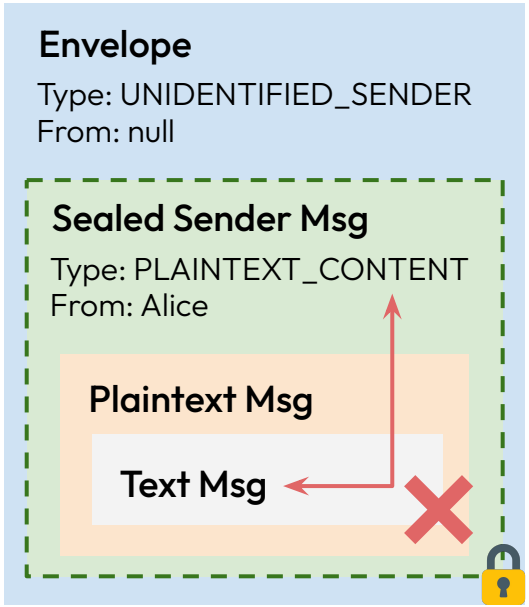




...and places a new, anonymized, envelope outside



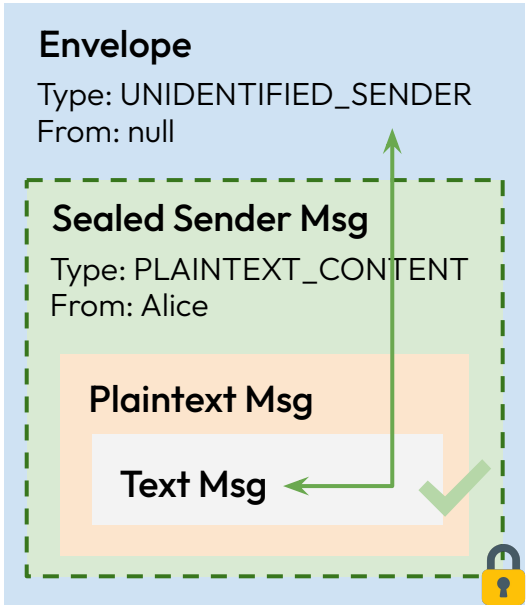
To perform message validation, one should look in the Sealed Sender envelope



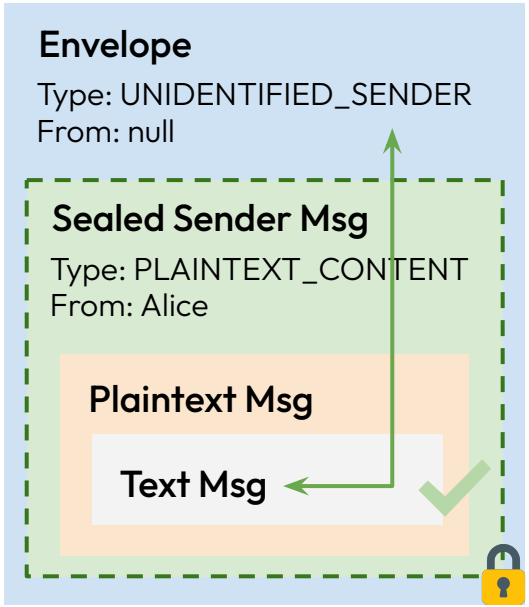

SO SIGNAL CHECKS THE SEALED SENDER ENVELOPE, RIGHT?


...RIGHT?

Signal (on Android) performs validation using the outer envelope type

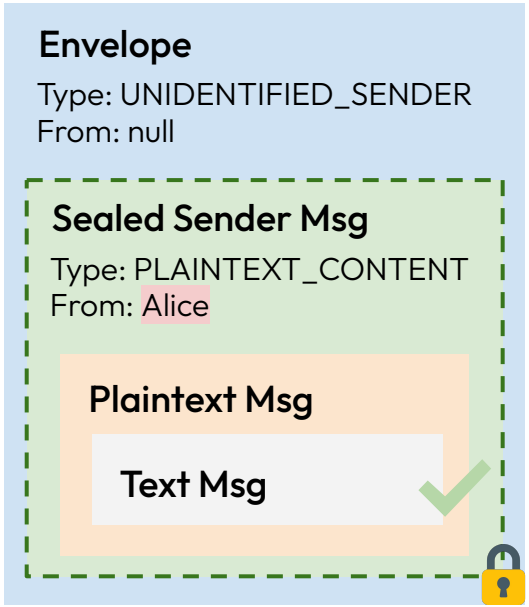


Signal (on Android) performs validation using the outer envelope type

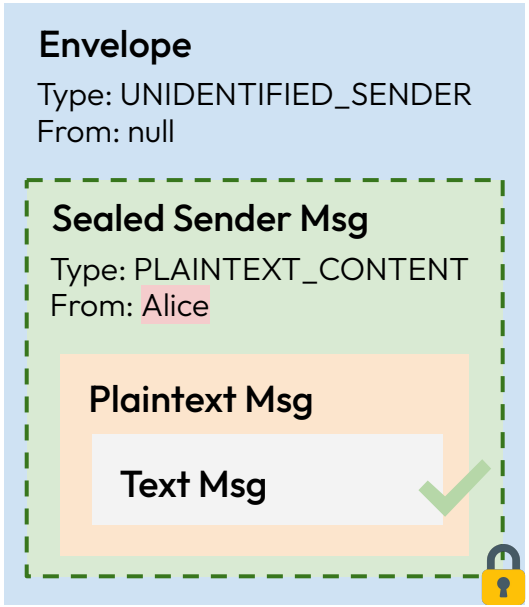


This passes validation and is therefore accepted as a text message!

But how can a malicious server forge the sender address?



But how can a malicious server forge the sender address?



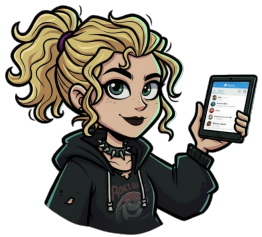
What properties does sealed sender actually provide?

Signal Sealed Sender

pk_A

Reveals the sender identity to Bob

pk_B



$PKE.Enc(pk_B, msg=pk_A||ACI_A||cert)$

Wants to wrap msg m

Signal Sealed Sender

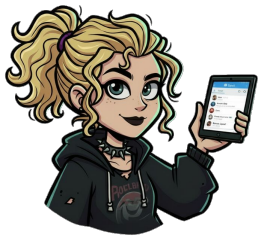
The Signal server signs (pk_A, ACI_A) to certify the association

pk_A

Reveals the sender identity to Bob

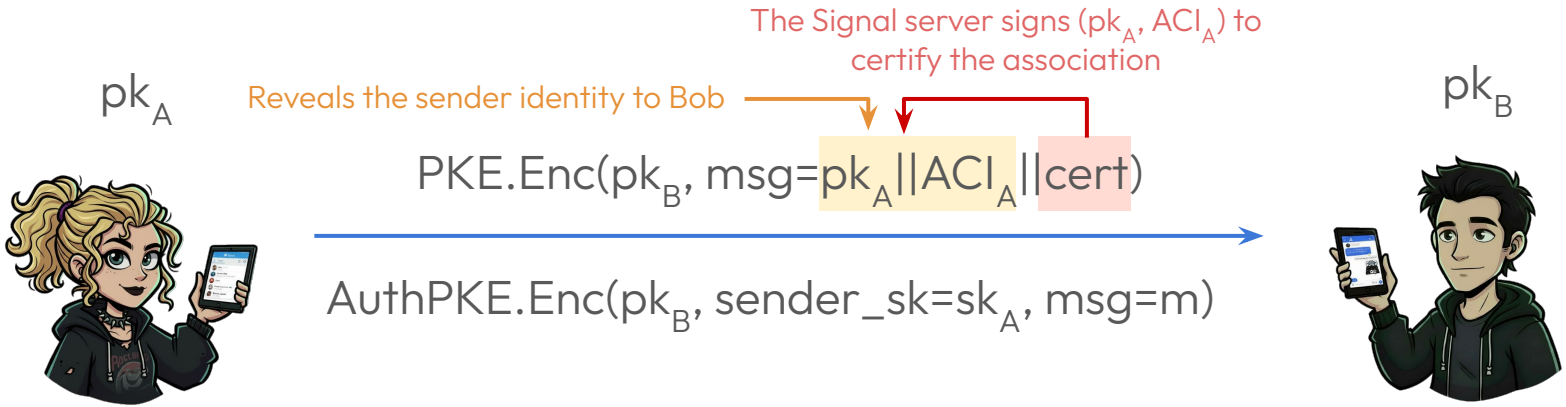
pk_B

$PKE.Enc(pk_B, msg=pk_A||ACI_A||cert)$



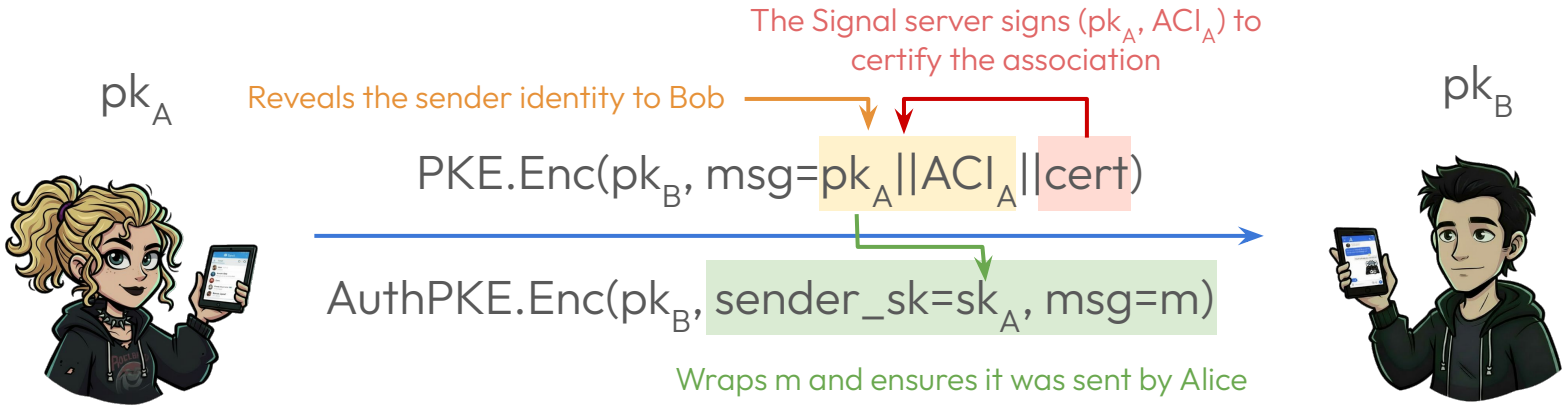
Wants to wrap msg m

Signal Sealed Sender



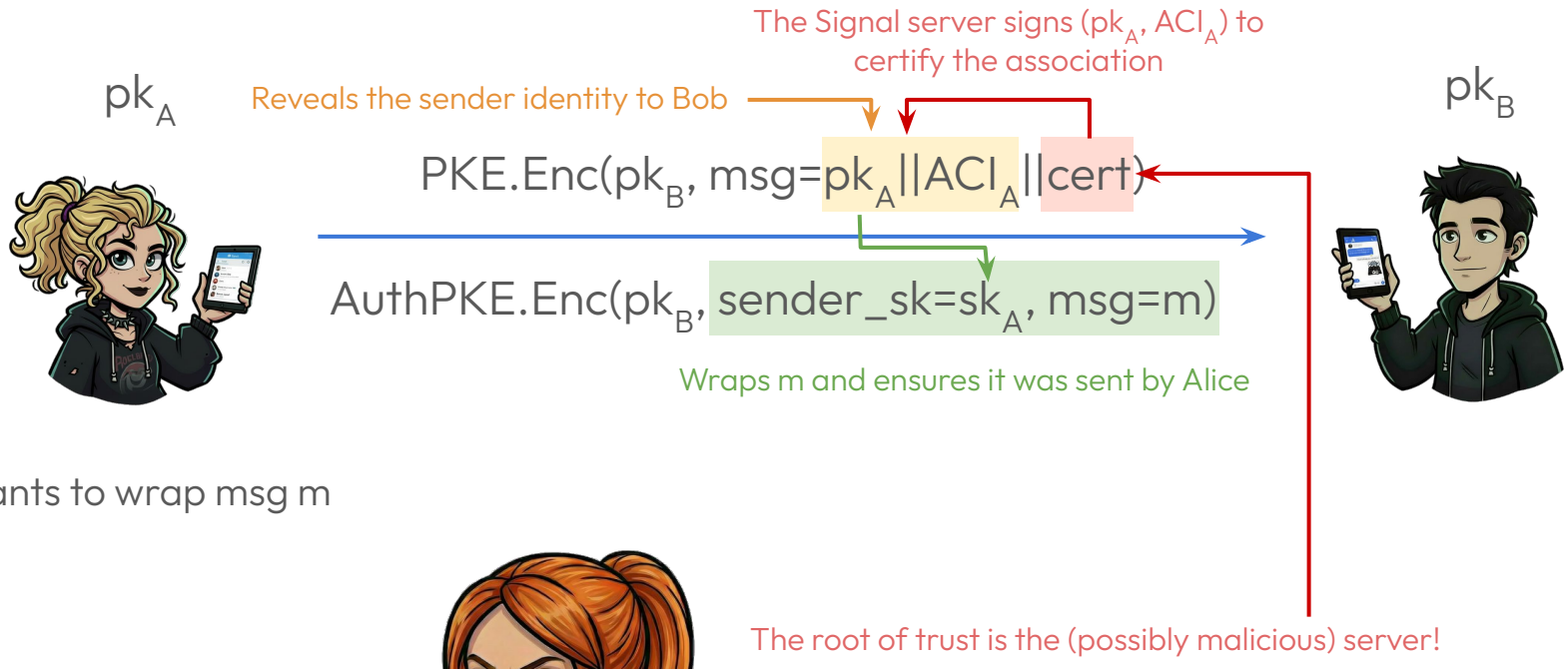
Wants to wrap msg m

Signal Sealed Sender



Wants to wrap msg m

Signal Sealed Sender



Sealed Sender does not authenticate

(sk_M, pk_M)

Wants to wrap msg m

PKE.Enc(pk_B, msg=pk_M||ACI_A||cert')

AuthPKE.Enc(pk_B, sender_sk=sk_M, msg=m)

pk_B

Sealed Sender does not authenticate

(sk_M, pk_M)

Wants to wrap msg m

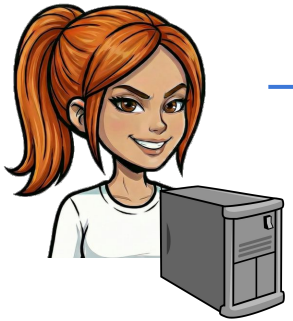
PKE.Enc(pk_B, msg=pk_M||ACI_A||cert')

AuthPKE.Enc(pk_B, sender_sk=sk_M, msg=m)

pk_B

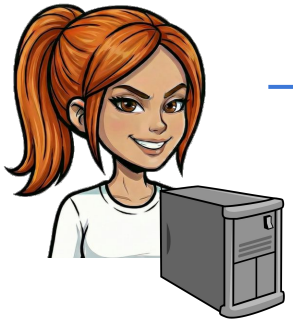
But Bob knows pk_A if he contacted Alice beforehand, so surely he can check that pk_M!=pk_A...

Sealed Sender does not authenticate



I'm Alice, and pk_M is my long-term key!

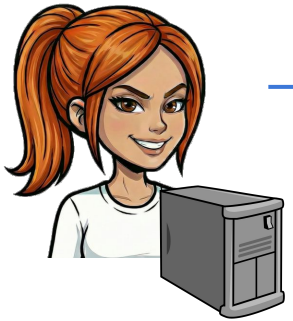
Sealed Sender does not authenticate



I'm Alice, and pk_M is my long-term key!

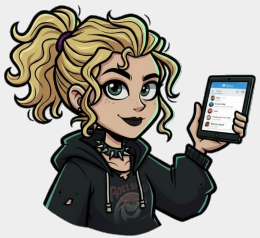
Here's a message encrypted using pk_M!

Sealed Sender does not authenticate



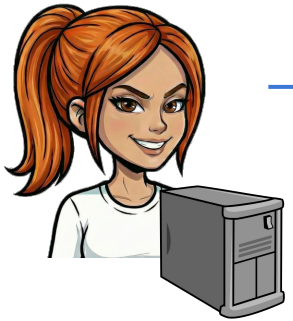
I'm Alice, and pk_M is my long-term key!

Here's a message encrypted using pk_M!



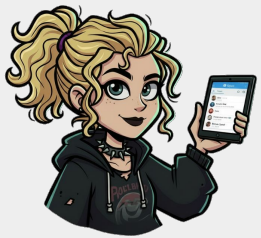
pk_A?

Sealed Sender does not authenticate



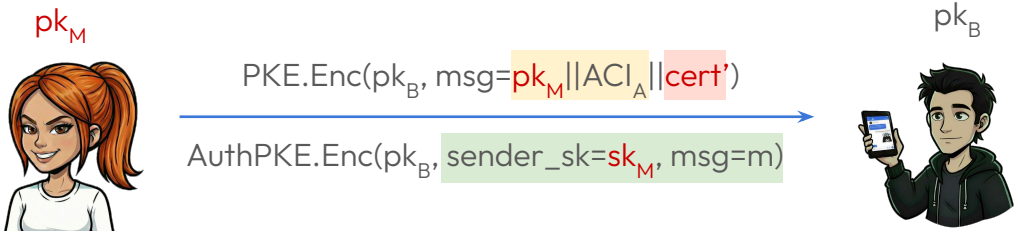
I'm Alice, and pk_M is my long-term key!

Here's a message encrypted using pk_M!



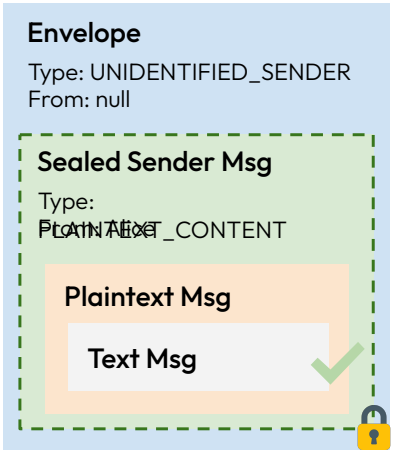
pk_A?

Combining the Two Issues

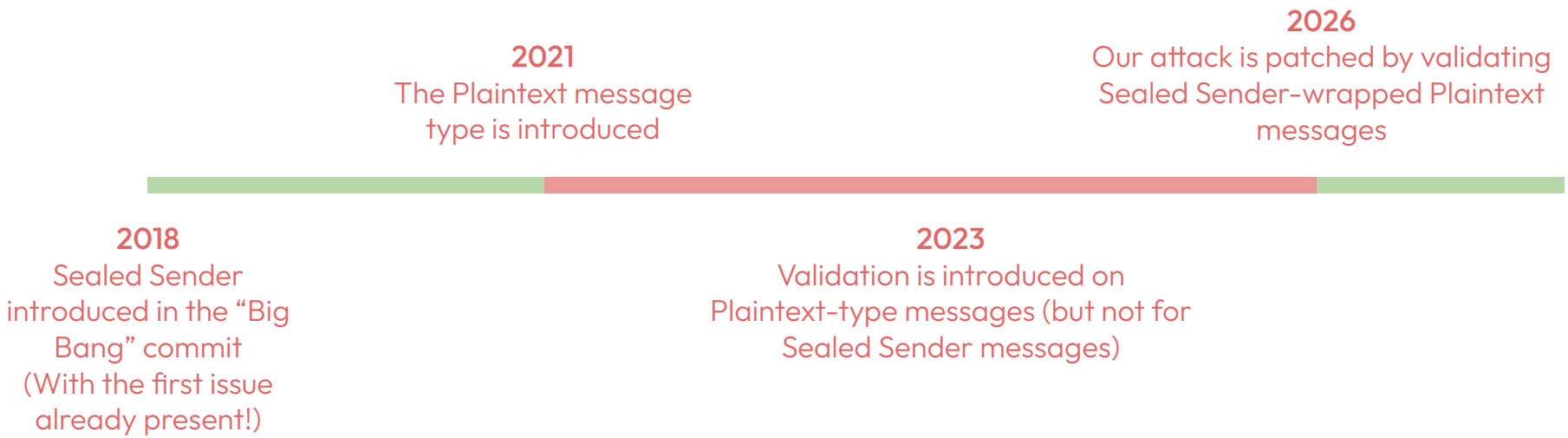


The server can send messages as any user…

… and send plaintext content, bypassing authentication



Vulnerability Timeline



This vulnerability existed for 5 years!

Conclusions

Signal is an app...

Its cryptography should be studied at the system level

Signal

The PQXDH Key Agreement Protocol

Revision 3, 2023-05-24, Last Updated: 2024-01-23 [PDF]

Ehren Kret, Rolfe Schmidt

The Double Ratchet Algorithm

Revision 4, 2025-11-04 [PDF]

Trevor Perrin (editor), Moxie Marlinspike, Rolfe Schmidt (revision 3+)

...AND a set of protocols

Signal

Private Group Messaging

moxie0 on 05 May 2014

The X3DH Key Agreement Protocol

Revision 1, 2016-11-04 [PDF]

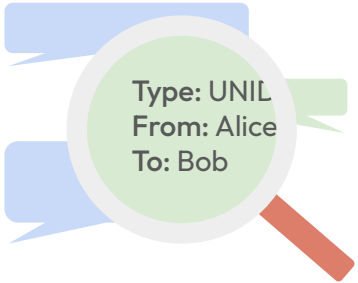
Moxie Marlinspike, Trevor Perrin (editor)

Signal

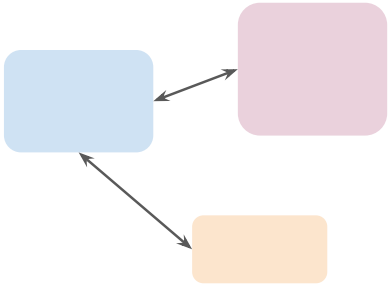
The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Could we have spotted this earlier?

Formal analyses *could* have spotted these attacks...



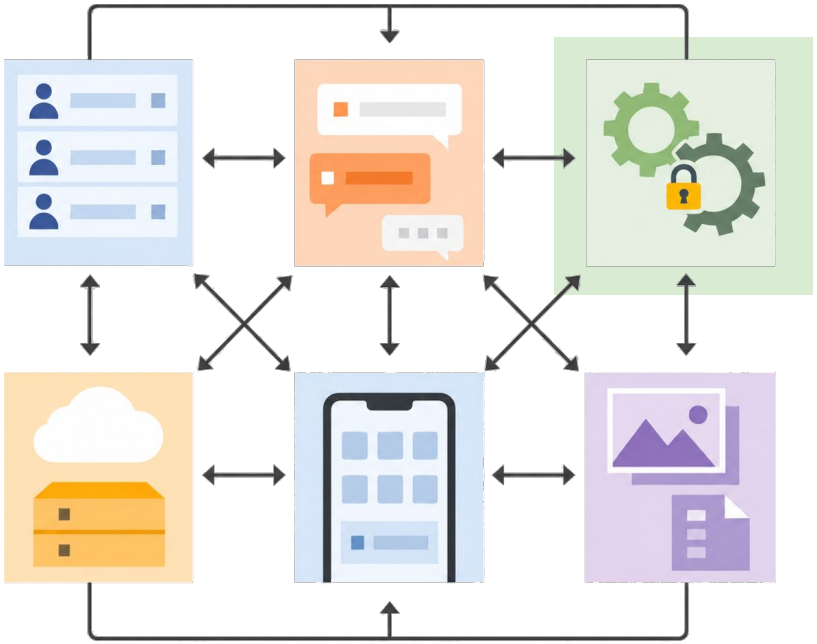
...but only if they captured the system in enough detail



...and only if they captured the interaction between all protocols

...but our techniques do not seem mature enough to capture this level of complexity yet.

For vendors



There's no such thing as a cryptographic core. Everything in the application is security-relevant.

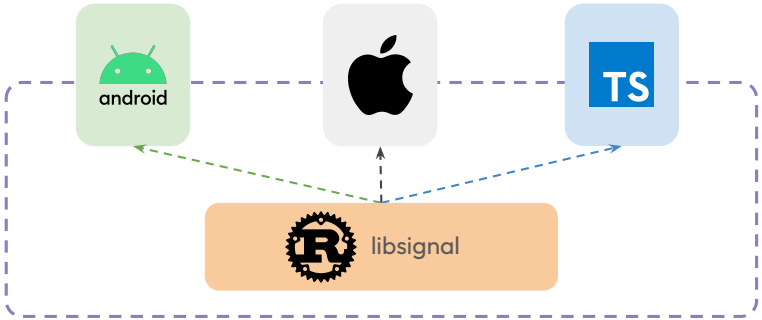
Cryptography should be audited along the rest of the code.

Platform-Dependent Logic

Attack 1 affects Android and Desktop.

Attack 2 affects only Android.

Avoid scattering cryptographic logic across modules and platforms.

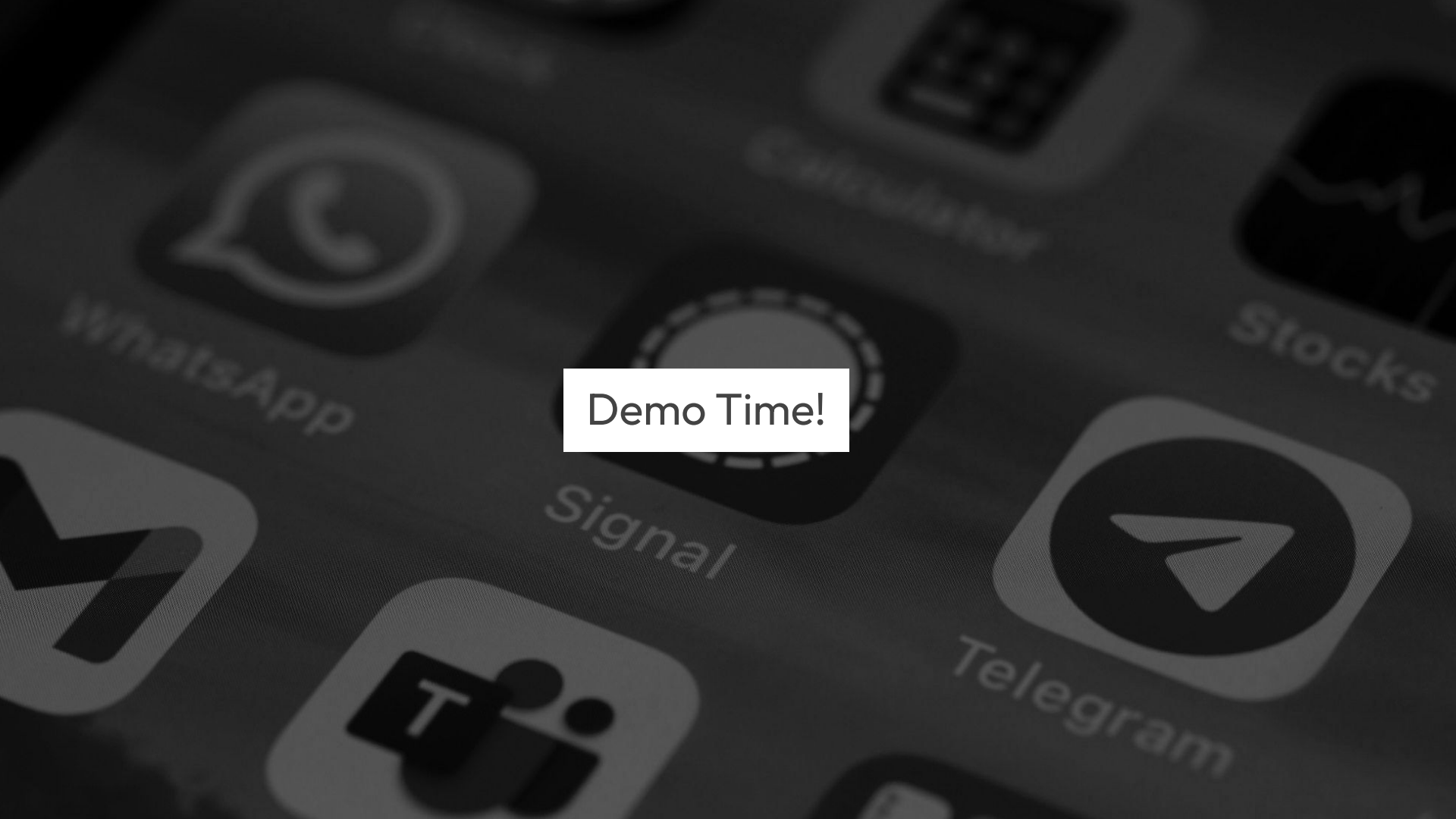


- We analysed Signal in the **malicious server setting.**
- We found **two attacks which violate conversation integrity.**
- Signal for iOS was unaffected by both attacks.
  Signal Desktop was unaffected by the second attack.
- Both attacks were **patched very quickly by the Signal team**, whom we thank for the smooth disclosure procedure.

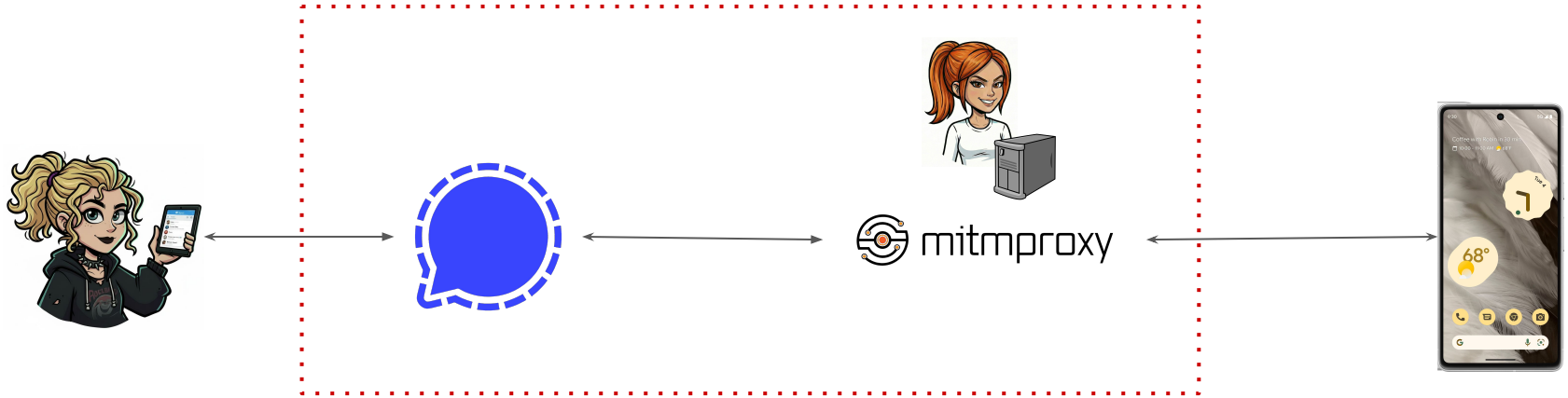
kitruong@ethz.ch
https://ktruong.dev

noemi.terzo@mpi-sp.org
https://nterzo.sh

ePrint Link


Demo Time!
WhatsApp
Signal
Telegram
Stocks

Injecting Messages



For ease of demonstration, we read the sender ACI from the device DB.
No cryptographic keys are read.