

Message Injection Attacks Against Signal

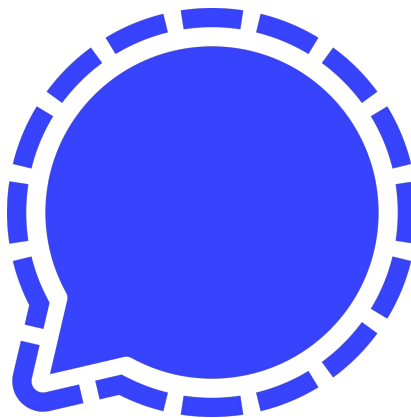
Work published at USENIX Security 2026

Kien Tuong Truong¹, Noemi Terzo², Kenny Paterson¹

¹ETH Zurich

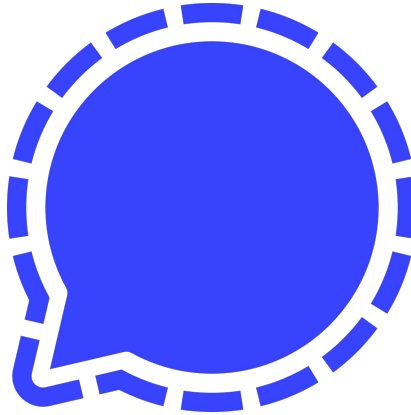
²Max Planck Institute for Security and Privacy

What's Signal?



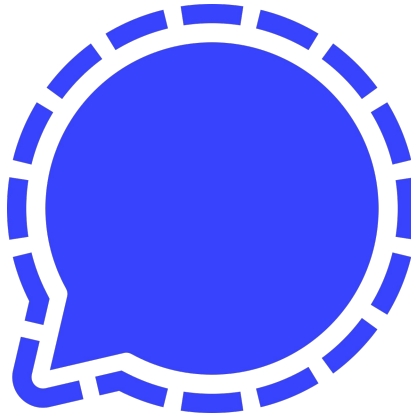
What's Signal?

“The messaging app”



What's Signal?

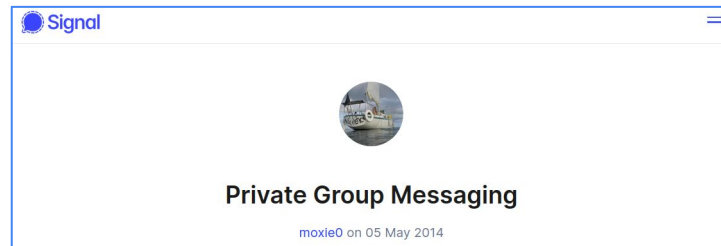
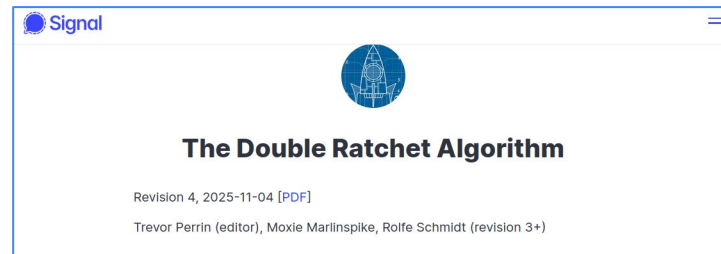
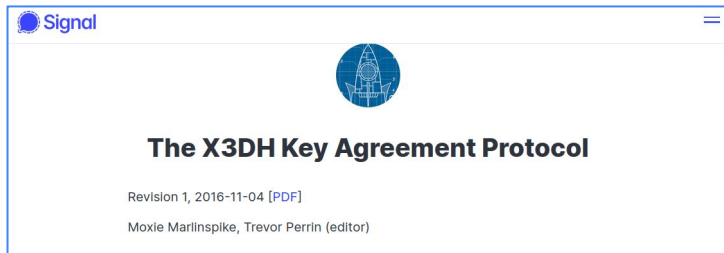
“The messaging app”



“The set of protocols”

What's Signal?

“The set of protocols”



What's Signal?

“The set of protocols”



Automated Verification for Secure Messaging Protocols A Symbolic and Computation

Nadim Kobeissi
INRIA Paris

Karthikeyan Bhargava
INRIA Paris



The PQXDH Key Agreement Protocol

Revision 3, 2023-05-24, Last Updated: 2024-01-23 [\[PDF\]](#)
Ehren Kret, Rolfe Schmidt

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



The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [\[PDF\]](#)
Moxie Marlinspike, Trevor Perrin (editor)



Private Group Messaging

moxie0 on 05 May 2014

What's Signal?

“The set of protocols”

 Signal

Automated Verification for Secure Messaging Protocols
A Symbolic and Computation

Nadim Kobeissi
INRIA Paris

Karthikeyan Bhargavan
INRIA Paris

 Signal



The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [PDF]

Moxie Marlinspike, Trevor Perrin (editor)

Formal verification of the PQXDH Post-Quantum key agreement protocol for end-to-end secure messaging

Karthikeyan Bhargavan, *Cryspen*; Charlie Jacomme, *Inria Nancy Grand-Est, Université de Lorraine, LORIA, France*; Franziskus Kiefer, *Cryspen*; Rolfe Schmidt, *Signal Messenger*

A More Complete Analysis of the Signal Double Ratchet Algorithm

Alexander Bienstock¹(✉), Jaiden Fairoze², Sanjam Garg^{2,3}, Pratyay Mukherjee⁴, and Srinivasan Raghuraman⁵

¹ New York University, New York, USA

abienstock@cs.nyu.edu

² UC Berkeley, Berkeley, USA

³ NTT Research, Sunnyvale, USA

⁴ Swirlds Labs, Dallas, USA

The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption *

Melissa Chase
Microsoft Research
Redmond, Washington, USA
melissac@microsoft.com

Trevor Perrin
Signal Technology Foundation
USA
trevp@signal.org

Greg Zaverucha
Microsoft Research
Redmond, Washington, USA
gregz@microsoft.com

What's Signal?

“The set of protocols”



Automated Verification for Secure Messaging Protocols A Symbolic and Computation

Nadim Kobeissi
INRIA Paris

Karthikeyan Bhargava
INRIA Paris



The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [PDF]

Moxie Marlinspike, Trevor Perrin (editor)

Formal verification of the POXDH Post-Quantum A Deniability Analysis of Signal's Initial Handshake PQXDH

Rune Fiedler
Technische Universität Darmstadt
Darmstadt, Germany
rune.fiedler@cryptoptexity.de

Christian Janson
Technische Universität Darmstadt
Darmstadt, Germany
christian.janson@cryptoptexity.de

A More Complete Analysis of the Signal Double Ratchet Algorithm

A Formal Security Analysis of the Signal Messaging Protocol

(Extended abstract: full version available at [10])

Katriel Cohn-Gordon*, Cas Cremers*, Benjamin Dowling†, Luke Garratt*, Douglas Stebila‡

*University of Oxford, UK

†Royal Holloway, University of London, UK

‡McMaster University, Canada

katriel.cohn-gordon@cs.ox.ac.uk
cas.cremers@cs.ox.ac.uk
luke.garratt@cs.ox.ac.uk
benjamin.dowling@rhul.ac.uk
stebila@mcmaster.ca

The Signal Private Group System and Anonymous Credentials Supporting Efficient Verifiable Encryption *

Melissa Chase
Microsoft Research
Redmond, Washington, USA
melissac@microsoft.com

Trevor Perrin
Signal Technology Foundation
USA
trevp@signal.org

Greg Zaverucha
Microsoft Research
Redmond, Washington, USA
gregz@microsoft.com

What's Signal?

“The set of protocols”



Automated Verification for Secure Messaging Protocols A Symbolic and Computation

Nadim Kobeissi
INRIA Paris

Karthikeyan Bhargava
INRIA Paris



The Sesame Algorithm: Session Management for Asynchronous Message Encryption

Revision 2, 2017-04-14 [PDF]

Moxie Marlinspike, Trevor Perrin (editor)

A Denial

Techn

run

Security Analysis of Signal's PQXDH Handshake

Rune Fiedler¹ and Felix Günther²

¹ Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany

rune.fiedler@cryptoplexity.de

² IBM Research Europe – Zurich, Rüschlikon, Switzerland

mail@felixguenther.info

The

Signal Double Ratchet Algorithm

A Formal Security Analysis of the Signal Messaging Protocol

(Extended abstract: full version available at [10])

Katriel Cohn-Gordon*, Cas Cremers*, Benjamin Dowling†, Luke Garratt*, Douglas Stebila†

*University of Oxford, UK

katriel.cohn-gordon@cs.ox.ac.uk

cas.cremers@cs.ox.ac.uk

luke.garratt@cs.ox.ac.uk

†David H. Lehman, University of London, UK

benjamin.dowling@ucl.ac.uk

WhatsApp with Sender Keys? Analysis, Improvements and Security Proofs

David Balbás^{1,2} *, Daniel Collins³ **, and Phillip Gajland^{4,5} ***

¹ IMDEA Software Institute, Spain

² Universidad Politécnica de Madrid, Spain

³ EPFL, Switzerland

⁴ Max Planck Institute for Security and Privacy, Germany

⁵ Ruhr University Bochum, Germany

PQXDH

dt

de

and

credentials

ucha
search
ngton, USA
oft.com

What's Signal?

“The set of protocols”

Comprehensive Deniability Analysis of Signal Handshake Protocols:

X3DH, PQXDH to Fully Post-Quantum with

Shuichi Katsumata ^{1,2}

Guilhem Nic

Thom Wiggers ¹

¹ PQShield

² AIST

³ Univ Rennes, CNRS, IR

A Denial

Techn

run

Security Analysis of Signal's PQXDH Handshake

Rune Fiedler¹  and Felix Günther²

¹ Cryptoplexity, Technische Universität Darmstadt, Darmstadt, Germany

rune.fiedler@cryptoplexity.de

² IBM Research Europe – Zurich, Rüschlikon, Switzerland

Clone Detection in Secure Messaging: Improving Post-Compromise Security in Practice

Cas Cremers

CISPA Helmholtz Center for Information Security

Saarbrücken, Germany

cremers@cispa.saarland

Benjamin Kiesel

CISPA Helmholtz Center for Information Security

Saarbrücken, Germany

benjamin.kiesel@cispa.saarland

Jaiden Fairoze

CISPA Helmholtz Center for Information Security

Saarbrücken, Germany

jfairuze@student.unimelb.edu.au

Aurora Naska

CISPA Helmholtz Center for Information Security

Saarbrücken, Germany

s8aunask@stud.uni-saarland.de

Secure Messaging Protocol

(0))

Garratt*, Douglas Stebila†

rdon@cs.ox.ac.uk

ox.ac.uk

ox.ac.uk

Formal Analysis of Session-Handling in Secure Messaging: Lifting Security from Sessions to Conversations

Cas Cremers

CISPA Helmholtz Center
for Information Security

Charlie Jacomme*

Inria Paris, France

Aurora Naska

CISPA Helmholtz Center
for Information Security

Moxie Marlinspike, Trevor Perrin (editor)

WhatsApp with Sender Keys?

The S Analysis, Improvements and Security Proofs

Analyzing Group Chat Encryption in MLS, Session, Signal, and Matrix

Joseph Jaeger  and Akshaya Kumar 

School of Cybersecurity and Privacy

Georgia Institute of Technology, Atlanta, Georgia, US

{josephjaeger, akshayakumar}@gatech.edu

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.



Journalist

Can we meet in front of the Marriott?

10:23



Journalist

Actually, let's meet on Light street.

10:23

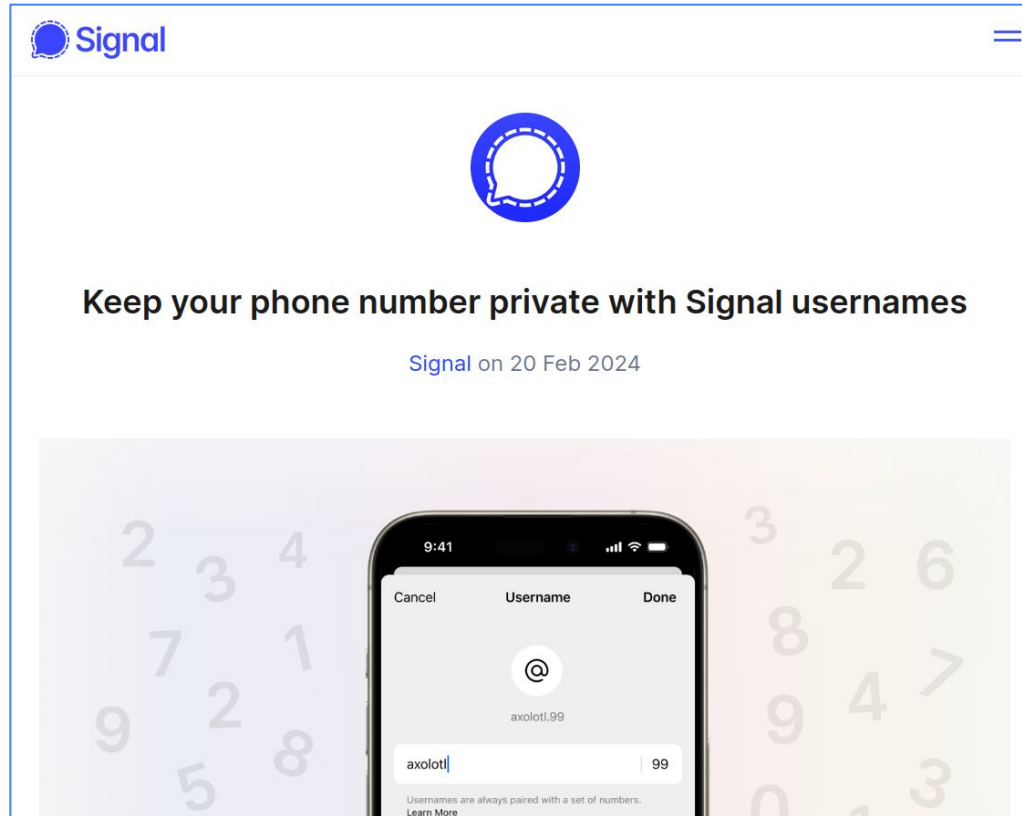
Sure!

10:24

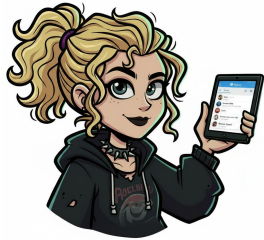


Attack 1: Usernames and Session Handling

Introduction of usernames in Signal



Phone numbers vs usernames



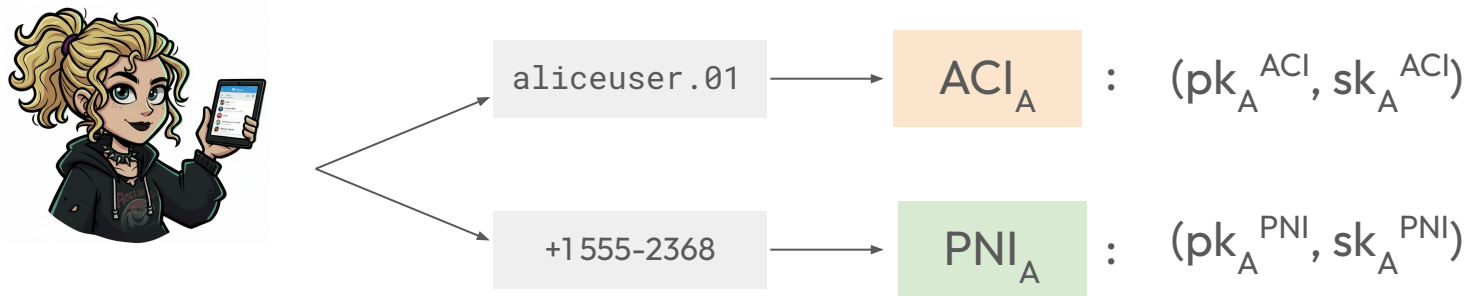
+1 555-2368



$\text{PNI}_A : (\text{pk}_A^{\text{PNI}}, \text{sk}_A^{\text{PNI}})$

PNI: Phone Number Identifier

Phone numbers vs usernames

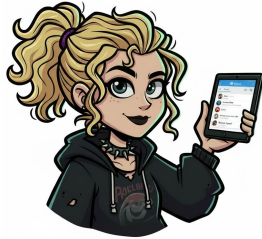


PNI: Phone Number Identifier

ACI: Account Identifier

ACIs, PNIs and sessions

Alice



ACI_A

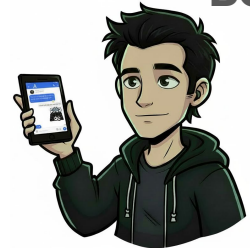
PNI_A



ACI_B

PNI_B

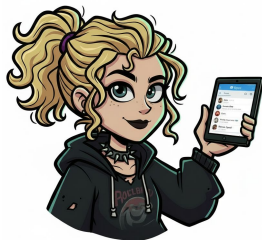
Bob



ACI_A — ACI_B session: by username

ACIs, PNIs and sessions

Alice



ACI_A

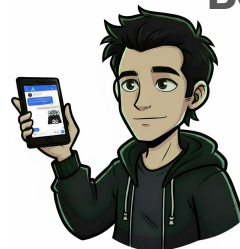
PNI_A

ACI_A — ACI_B session: by us

← Verify safety number



Bob

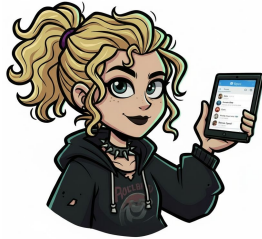


ACI_B

PNI_B

ACIs, PNIs and sessions

Alice



ACI_A

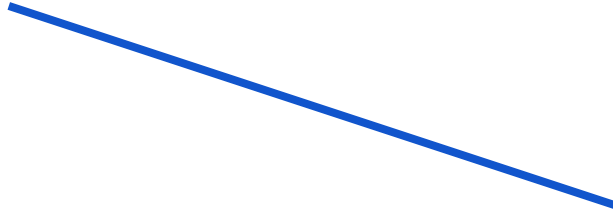
PNI_A

Bob



ACI_B

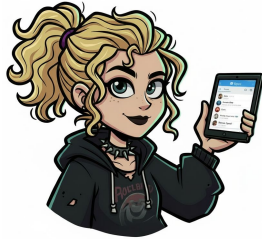
PNI_B



ACI_A — PNI_B session: by phone number

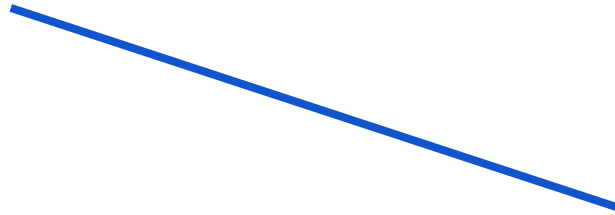
ACIs, PNIs and sessions

Alice



ACI_A

PNI_A



Bob



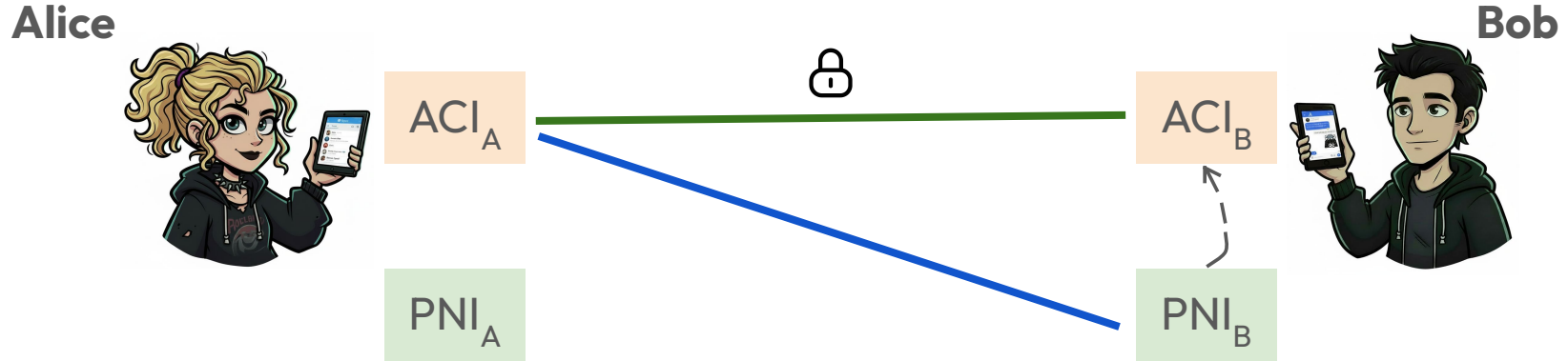
ACI_B

PNI_B



ACI_A — PNI_B session: by phone number
upgrade to ACI...

ACIs, PNIs and sessions

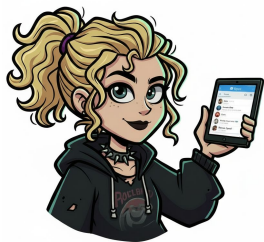


$ACI_A - PNI_B$ session: by phone number

upgrade to ACI... new $ACI_A - ACI_B$ session!

ACIs, PNIs and sessions

Alice



ACI_A

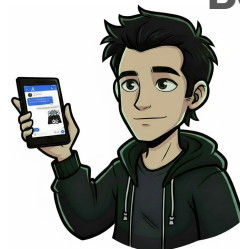
PNI_A



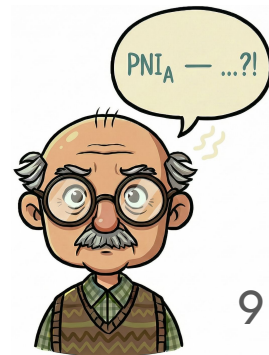
ACI_B

PNI_B

Bob

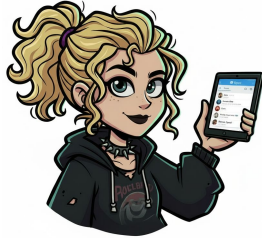


ACI_A — PNI_B session: by phone number
upgrade to ACI... new ACI_A — ACI_B session!



ACIs, PNIs and sessions

Alice

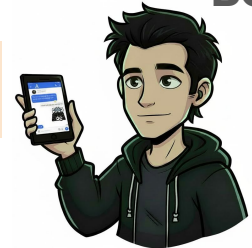


ACI_A



ACI_B

Bob



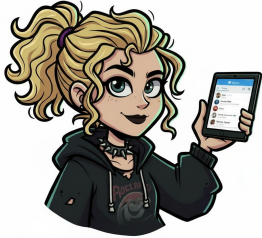
PNI_B

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

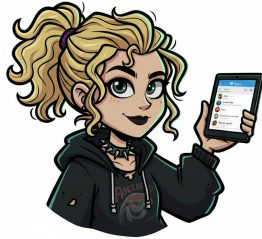
PNI_B

knows:

PNI_B, pk_B^{PNI}

“PNI-to-ACI” protocol

Alice



ACI_A

knows:

$\text{PNI}_B, \text{pk}_B^{\text{PNI}}$

Bob



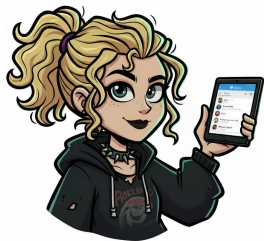
ACI_B

PNI_B

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

“PNI-to-ACI” protocol

Alice



ACI_A



ACI_B

Bob



PNI_B

knows:

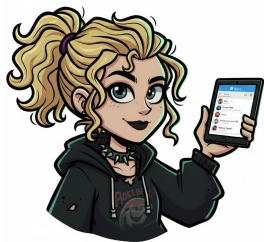
PNI_B, pk_B^{PNI}

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

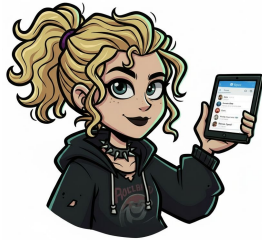
“PNI-to-ACI” protocol



...	...	...	...	...	...
+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



Bob

Which last name do you usually enter
from the inventors of RSA?

10:23

Rivest.

10:24



Bob

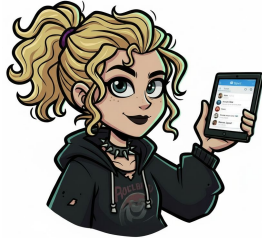
Me too!

10:23

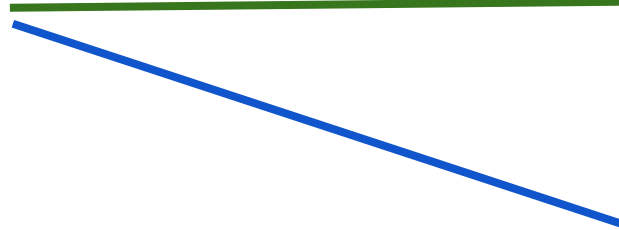
After linking, any message with PNI_B or ACI_B as a source
will be displayed in the same conversation.

The First Attack

Alice

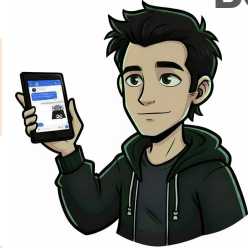


ACI_A



ACI_B

Bob



PNI_B

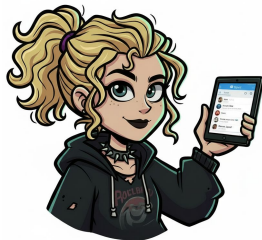
Mallory

$\tilde{PNI}_B$



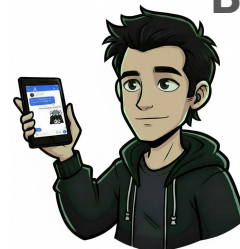
The First Attack

Alice



ACI_A

Bob



ACI_B

PNI_B

pkm

$$pkm = \langle PNI_B, c_A, \tilde{pk}_B^{PNI}, \dots \rangle$$

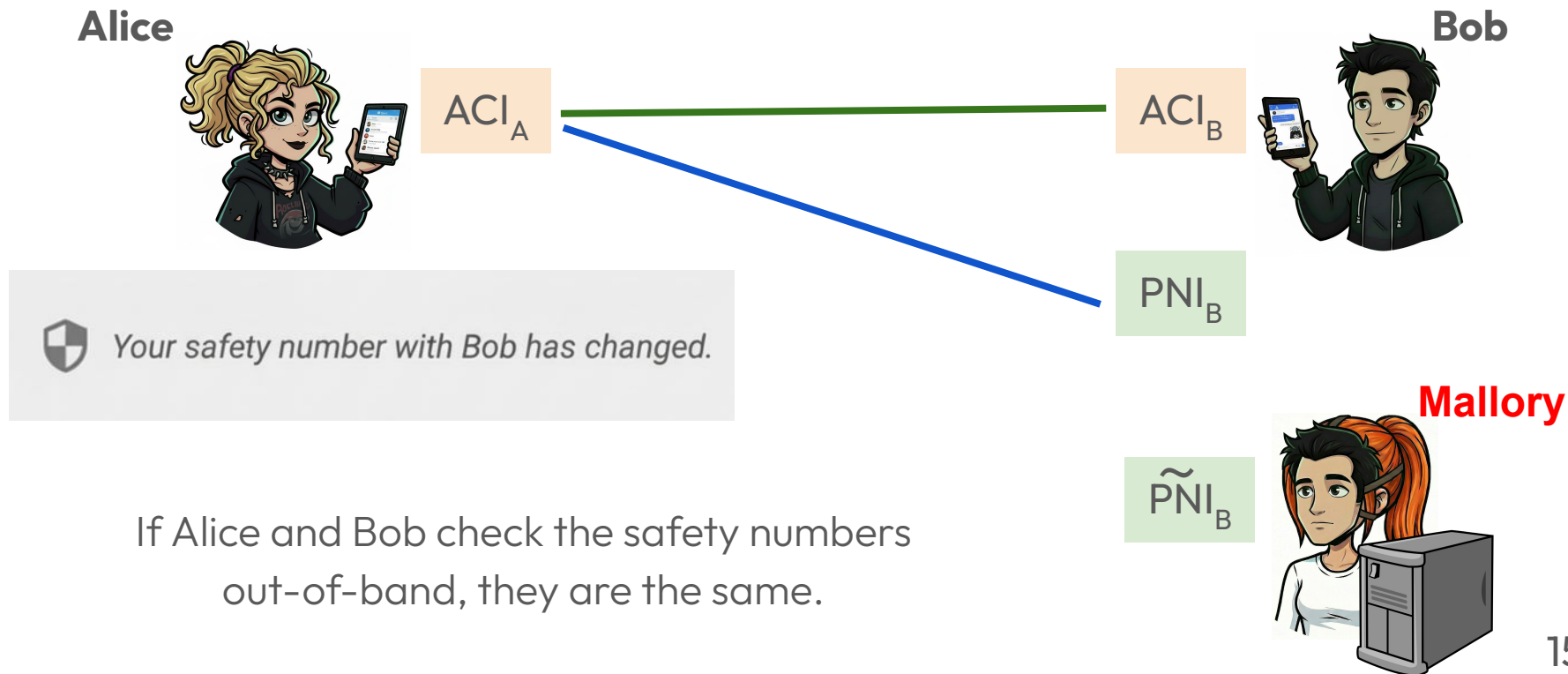
c_A = encryption of a message
computed with the key of
the new $ACI_A - \tilde{PNI}_B$ session

Mallory



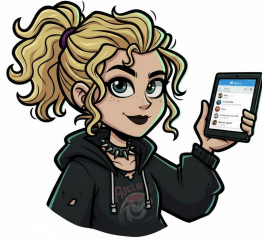
$\tilde{PNI}_B$

The First Attack



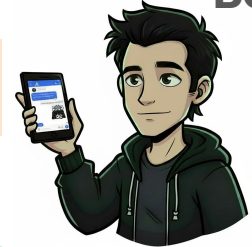
The First Attack

Alice



ACI_A

Bob



ACI_B

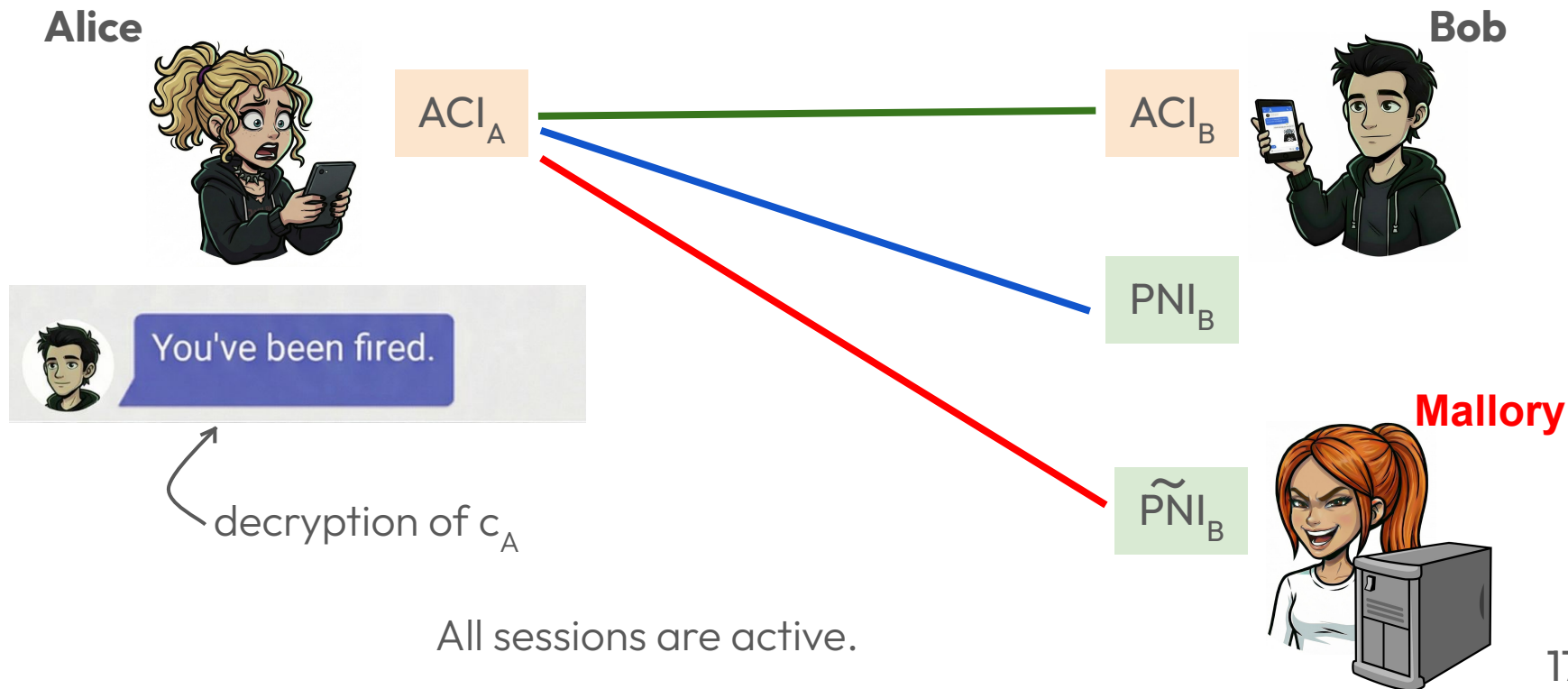
PNI_B

Mallory



$\tilde{PNI}_B$

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...

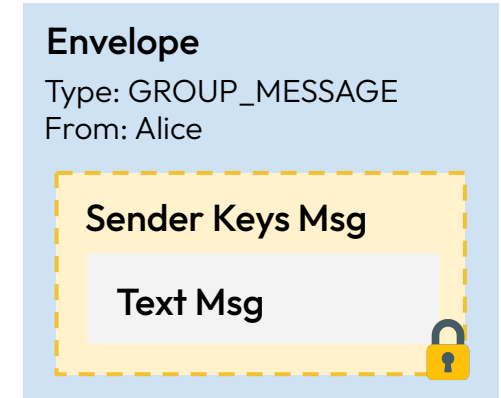
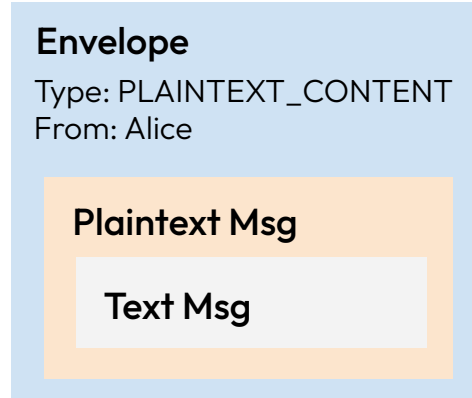
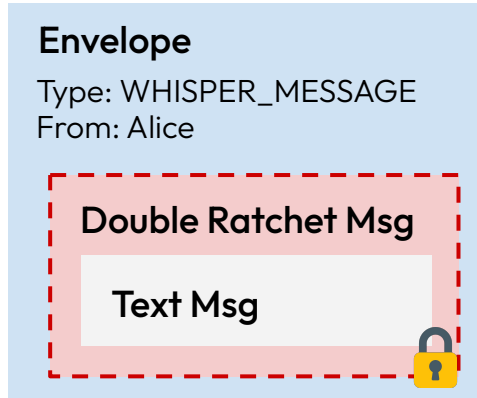
...but it cannot forge valid **ciphertexts**!

Injecting Messages

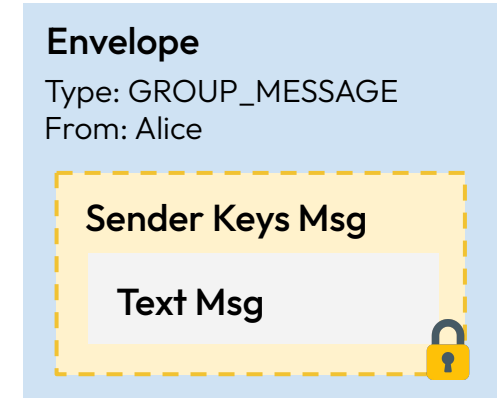
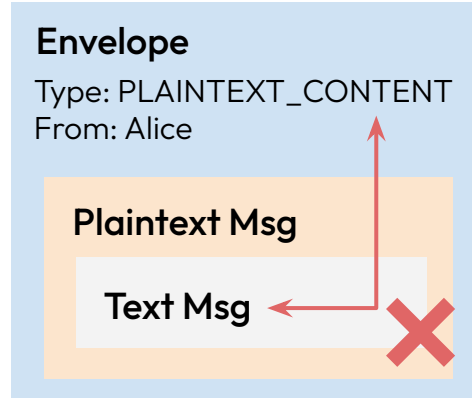
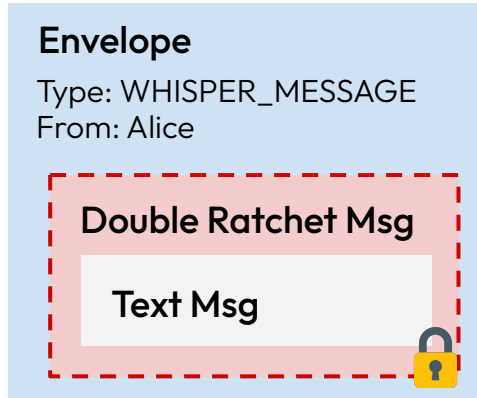


Plaintext content is allowed, but only to encode errors.

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

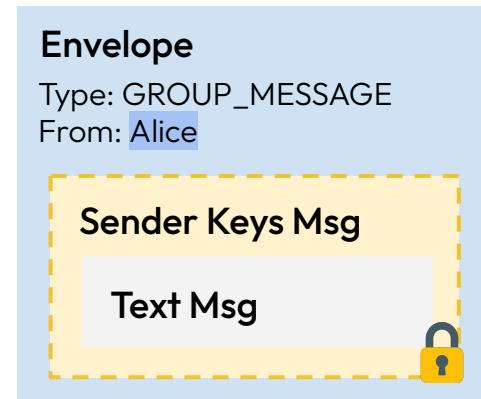
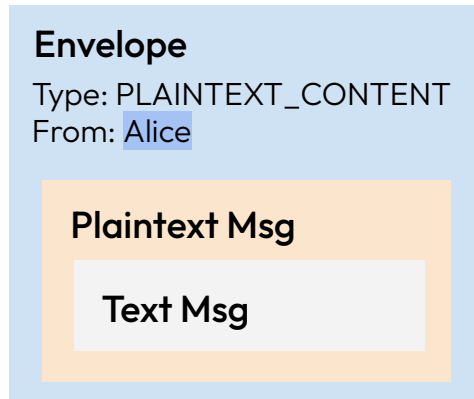
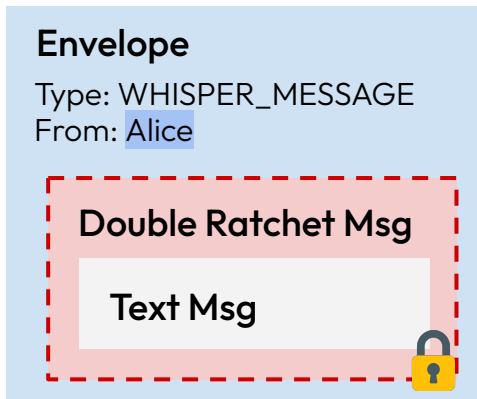


To understand the validation process, we need to look into the message structure.
Standard messages consist of an Envelope, wrapping some Content

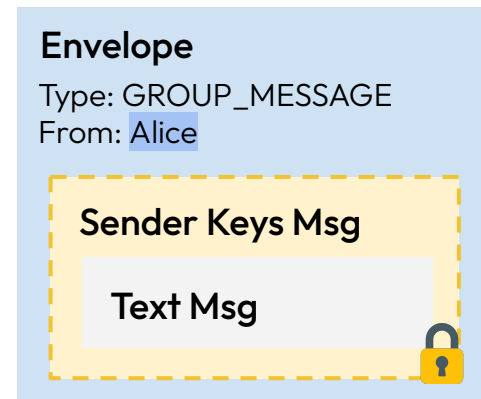
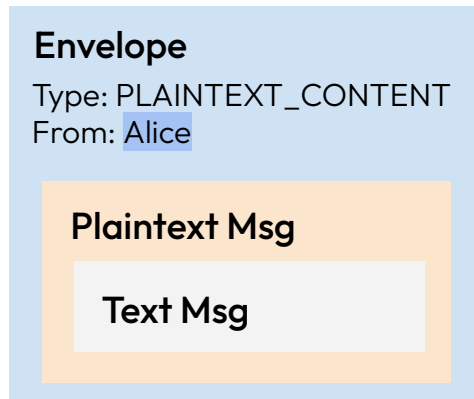
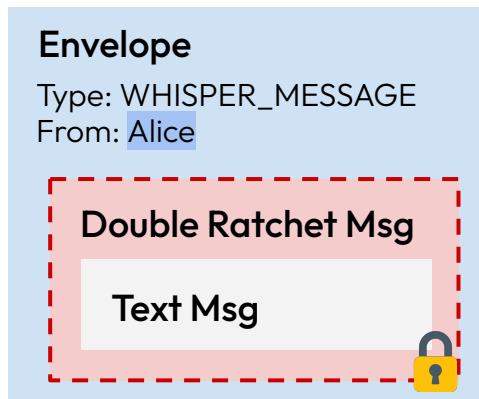


Plaintext messages containing Text messages are not accepted.

But these messages also leak the sender identity...

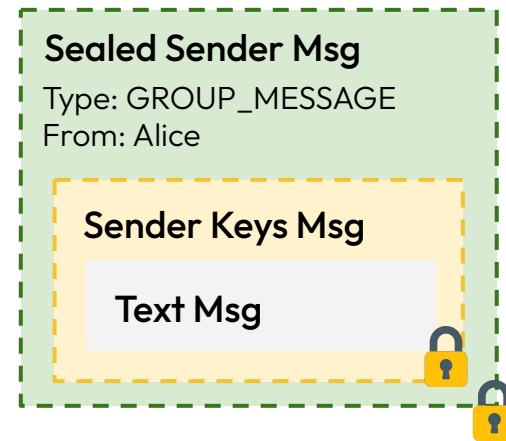
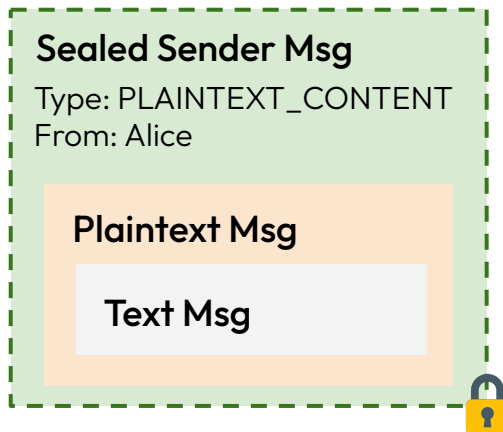
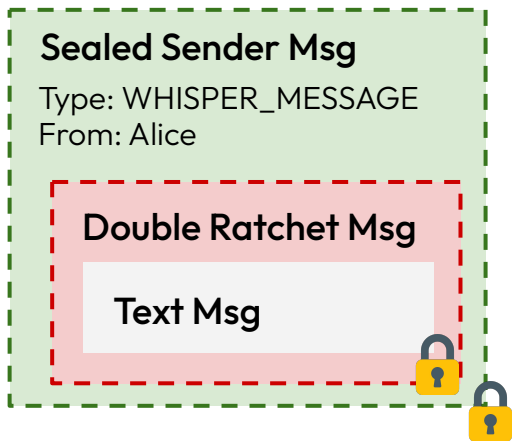


But these messages also leak the sender identity...

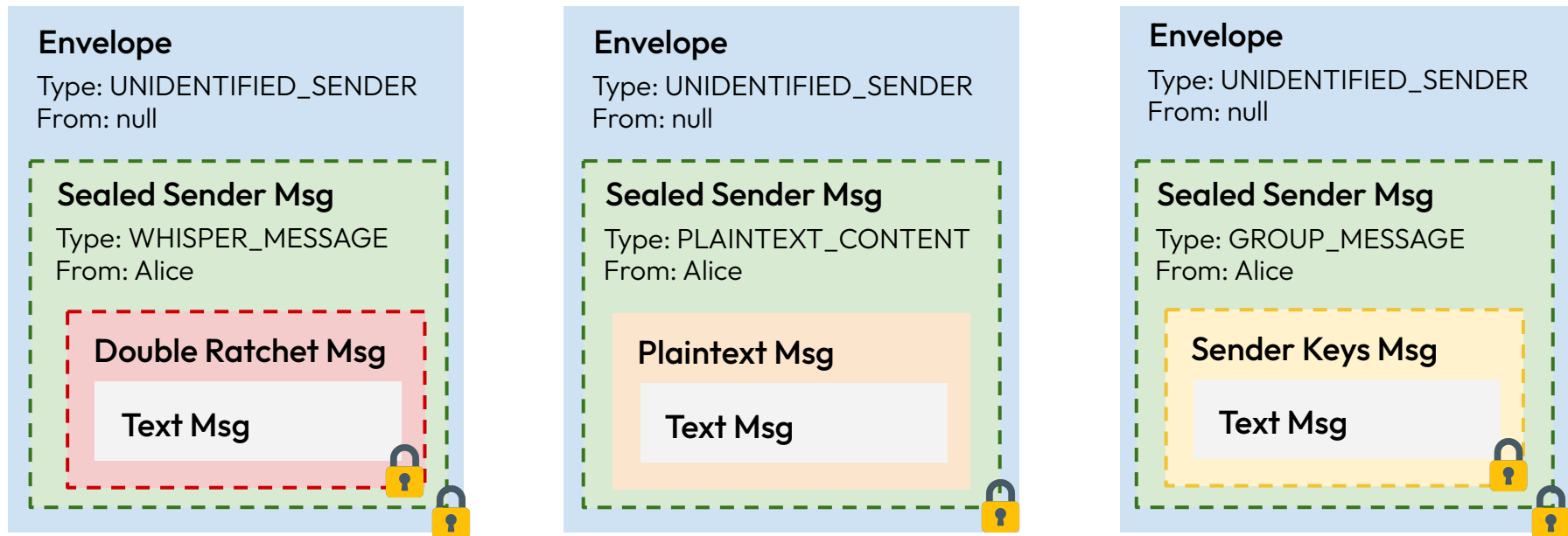


Sealed Sender is a protocol which aims to hide this metadata

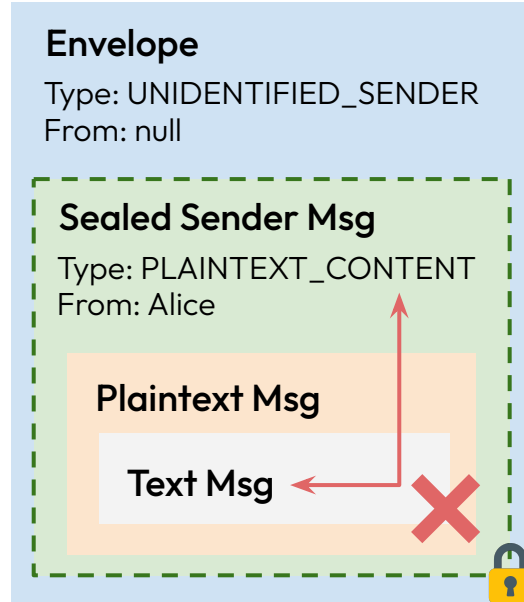
Sealed Sender encrypts and authenticates the envelope...



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



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

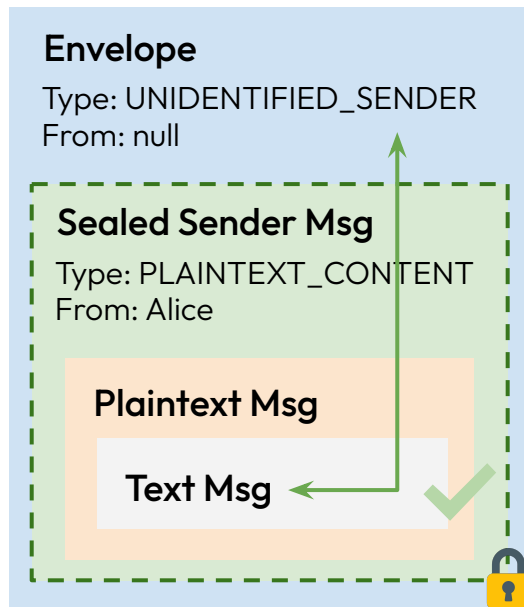




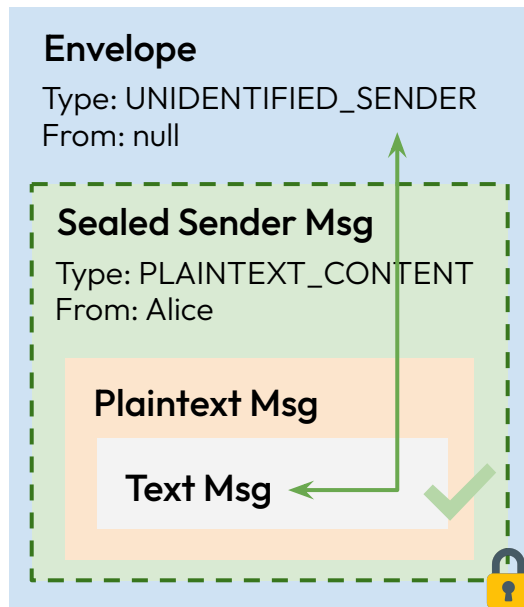




Signal (on Android) performs validation using the **outer envelope** type

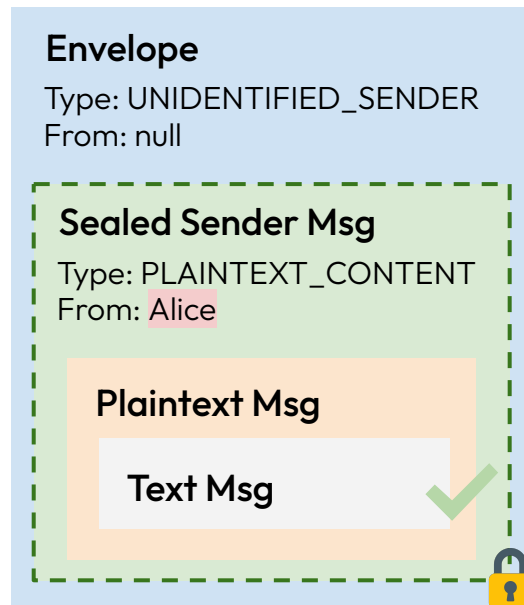


Signal (on Android) performs validation using the **outer envelope** type

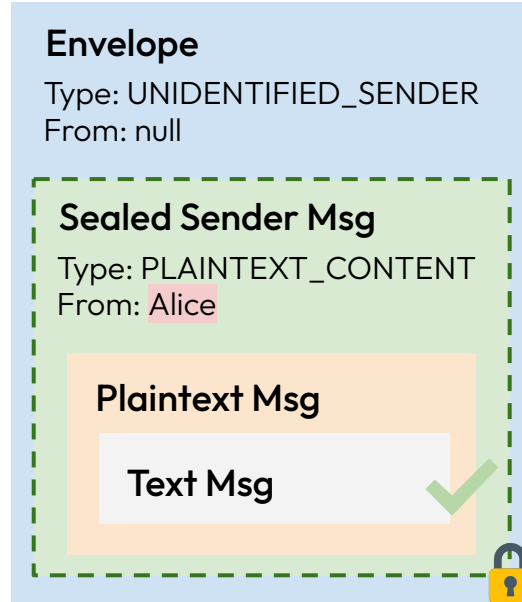


This passes validation and is therefore **accepted as a text message!**

But how can a malicious server forge the sender address?

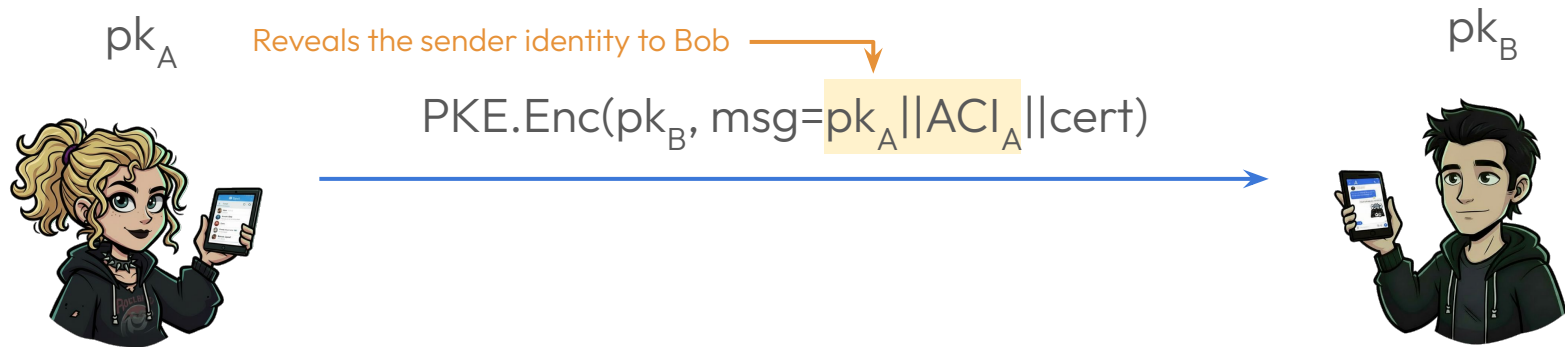


But how can a malicious server forge the sender address?



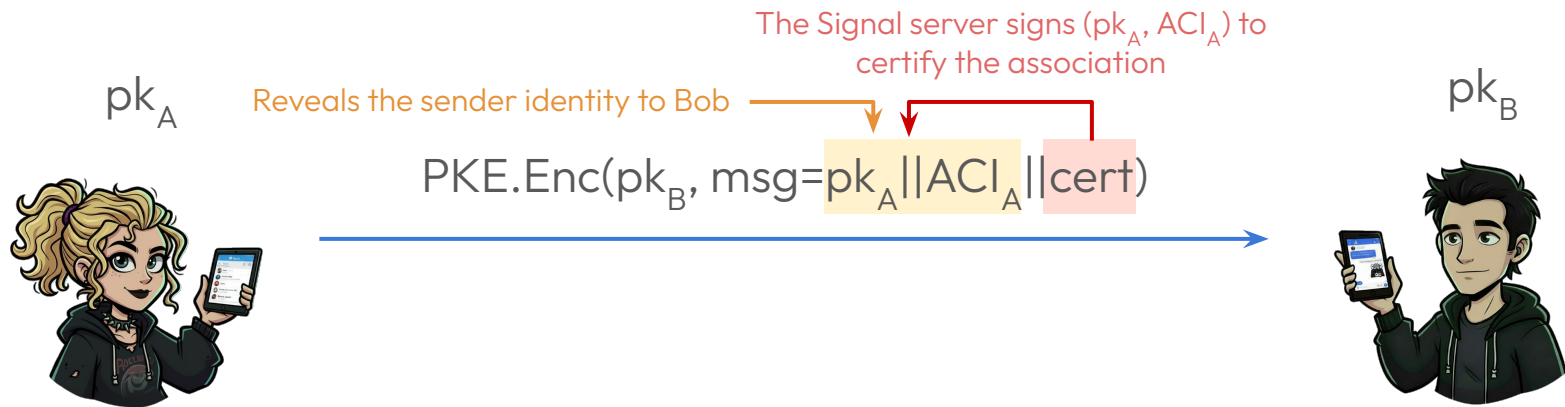
What properties does sealed sender actually provide?

Signal Sealed Sender

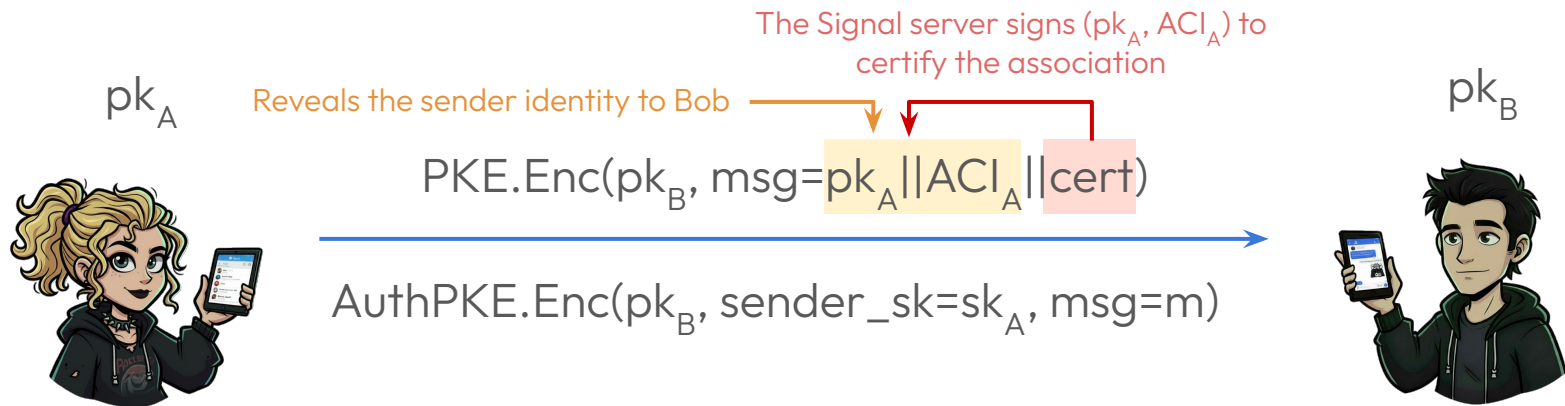


Wants to wrap msg m

Signal Sealed Sender

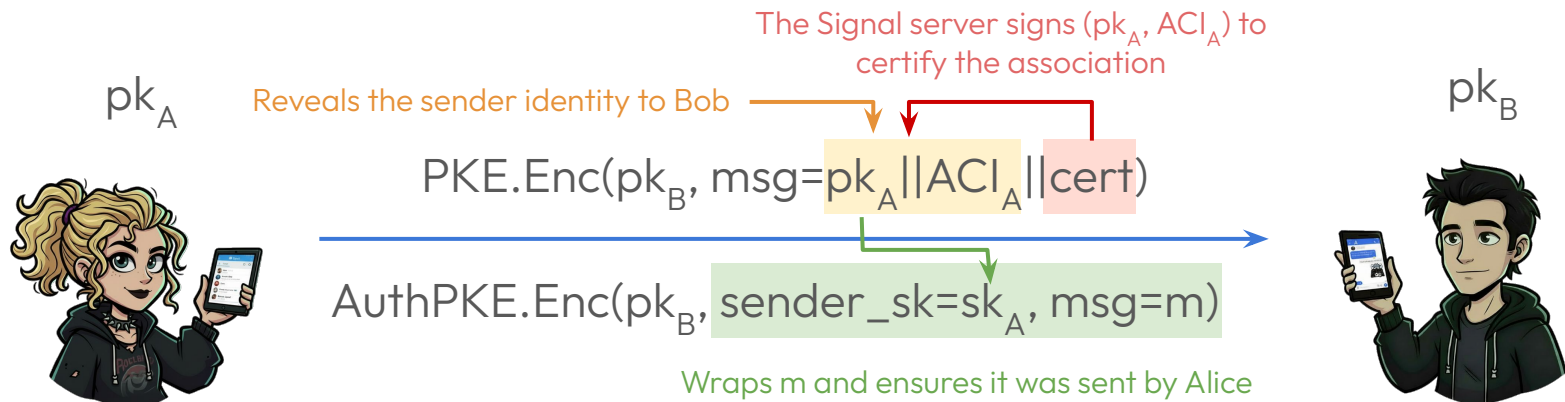


Signal Sealed Sender



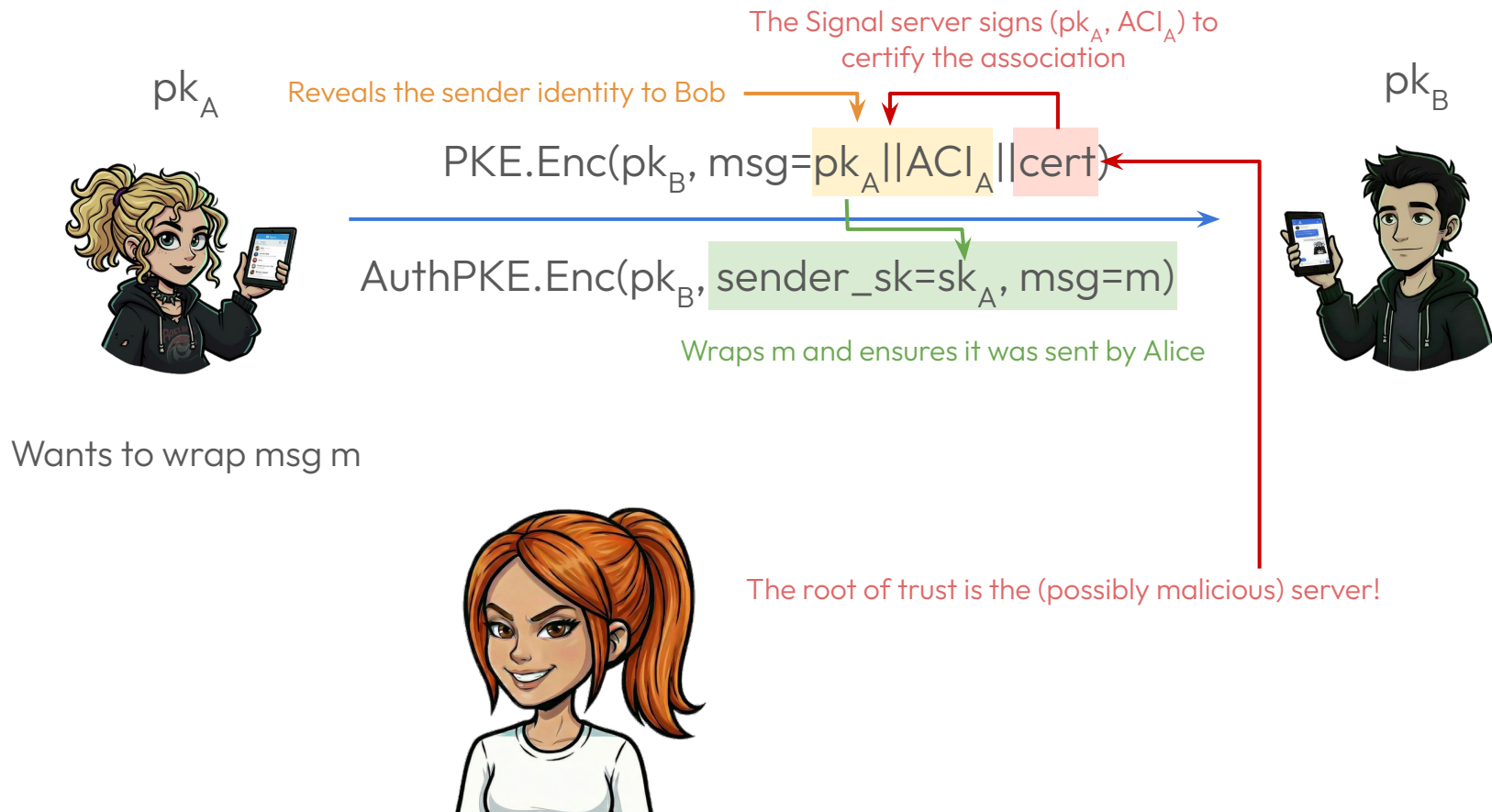
Wants to wrap msg m

Signal Sealed Sender



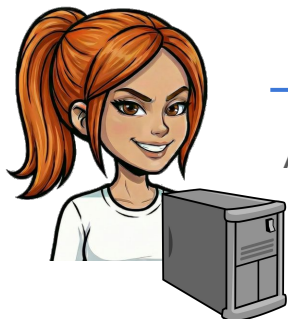
Wants to wrap msg m

Signal Sealed Sender



Sealed Sender does not authenticate

(sk_M, pk_M)



Wants to wrap msg m

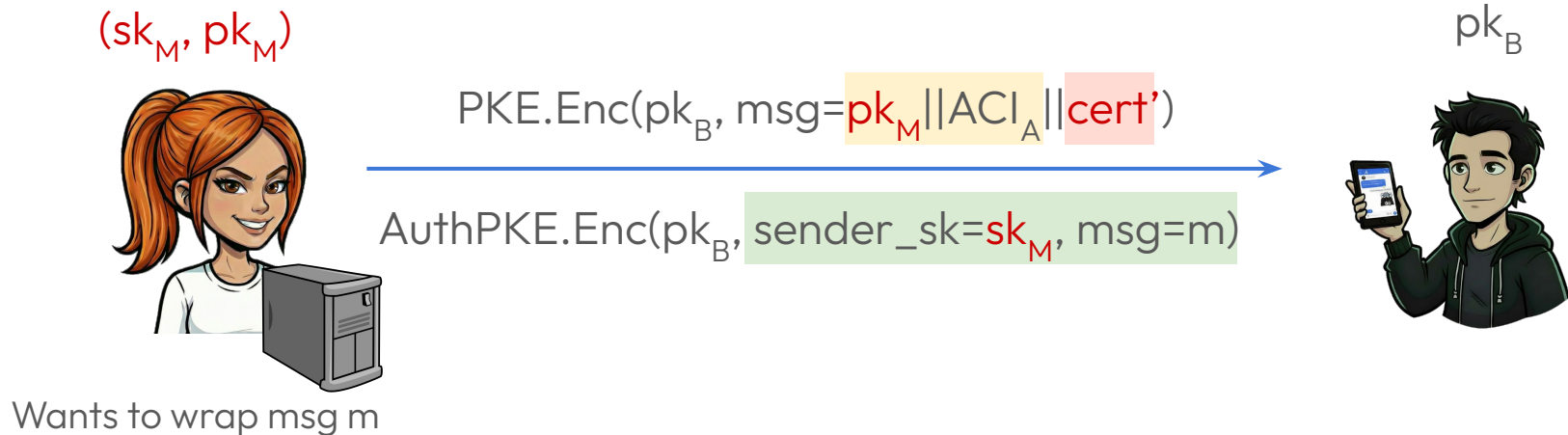
$\text{PKE.Enc}(pk_B, \text{msg} = pk_M || ACI_A || \text{cert}')$

$\text{AuthPKE.Enc}(pk_B, \text{sender_sk} = sk_M, \text{msg} = m)$

pk_B



Sealed Sender does not authenticate



But Bob knows pk_A if he contacted Alice beforehand, so surely he can check that $pk_M \neq pk_A \dots$



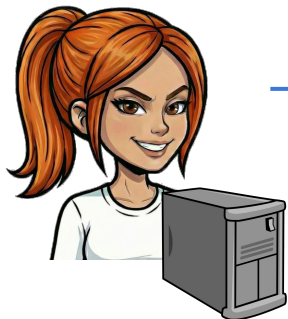
Sealed Sender does not authenticate



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



Sealed Sender does not authenticate

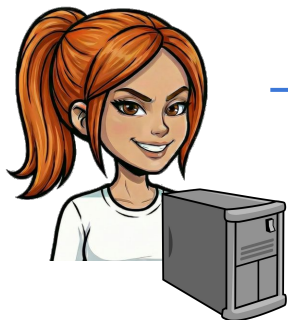


I'm Alice, and pk_M is my long-term key!

Here's a message encrypted using pk_M !

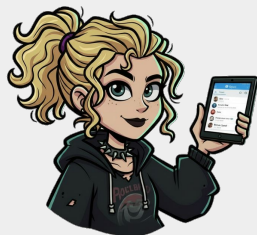


Sealed Sender does not authenticate



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

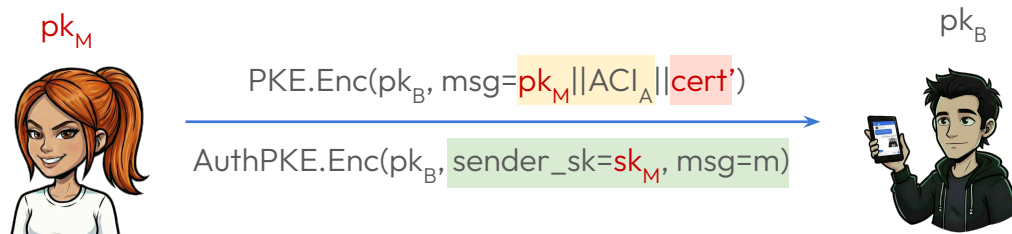


I'm Alice, and pk_M is my long-term key!

Here's a message encrypted using pk_M !

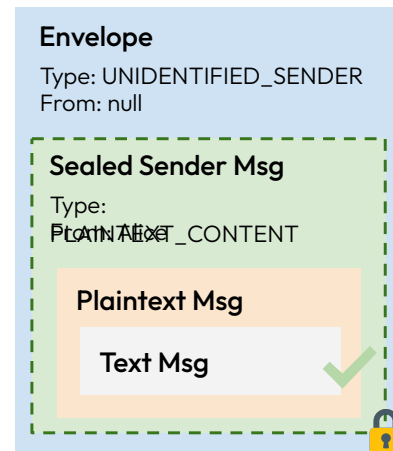


Combining the Two Issues

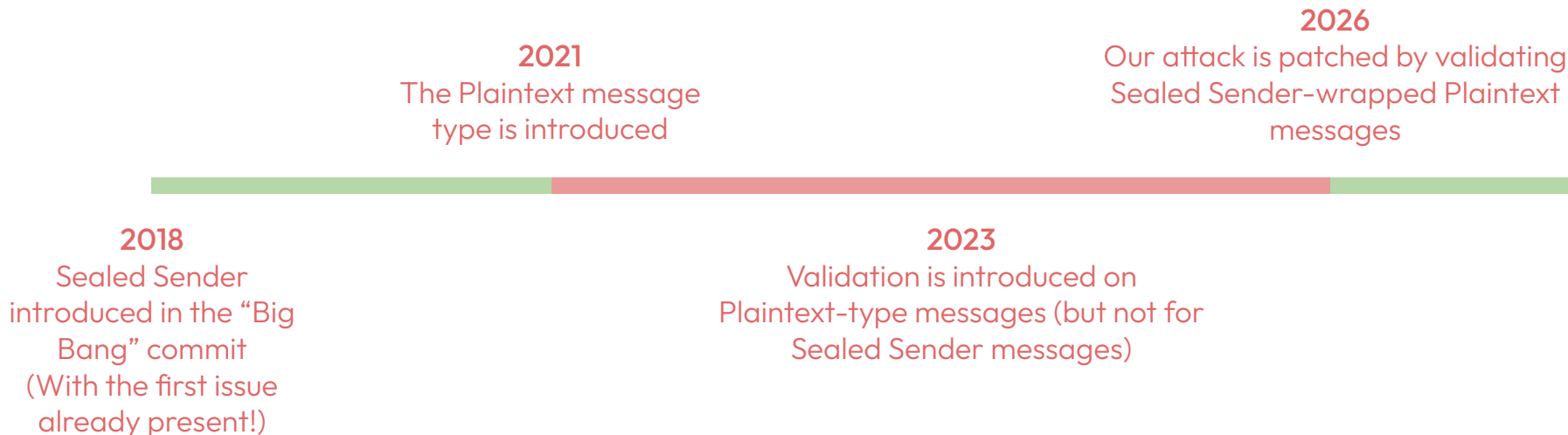


The server can send messages as any user...

... and send plaintext content, bypassing authentication



Vulnerability Timeline



This vulnerability existed for 5 years!

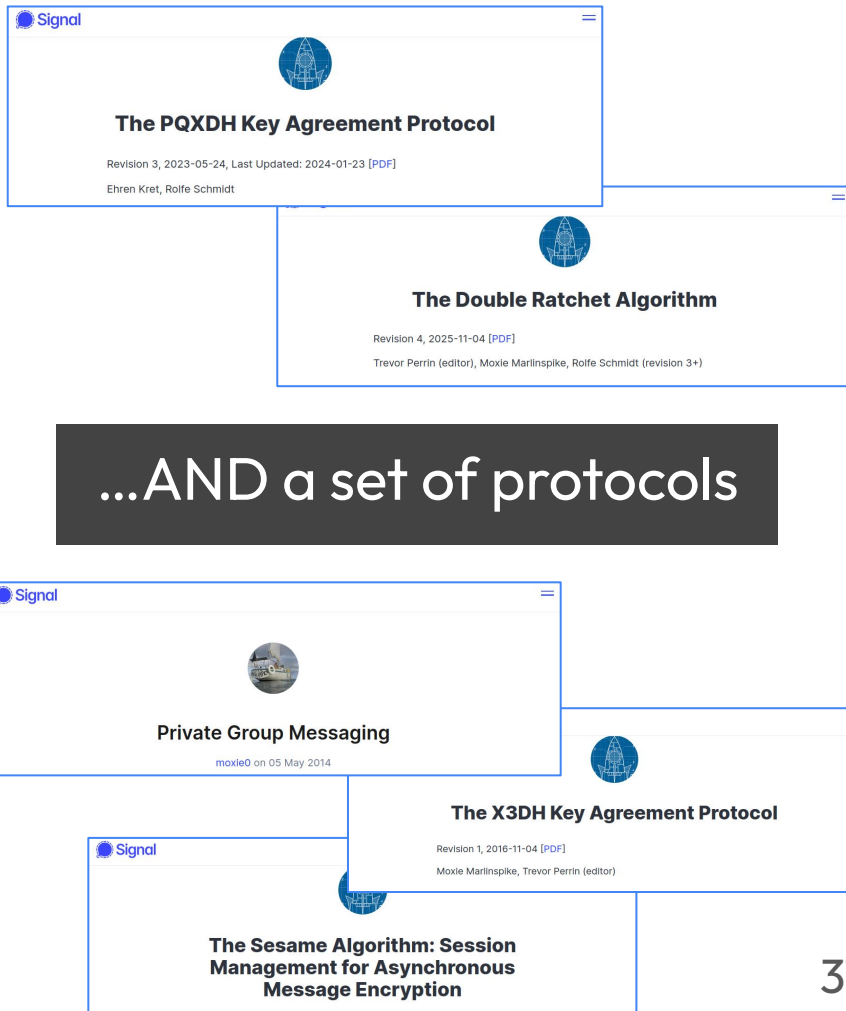


Conclusions

Signal is an app...

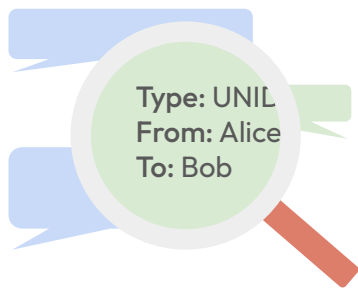


Its cryptography should be studied at the system level

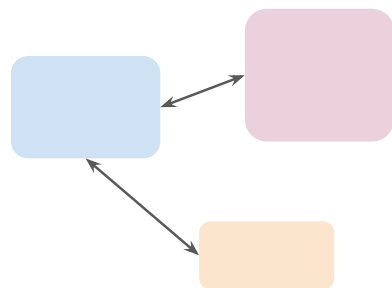


Could we have spotted this earlier?

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



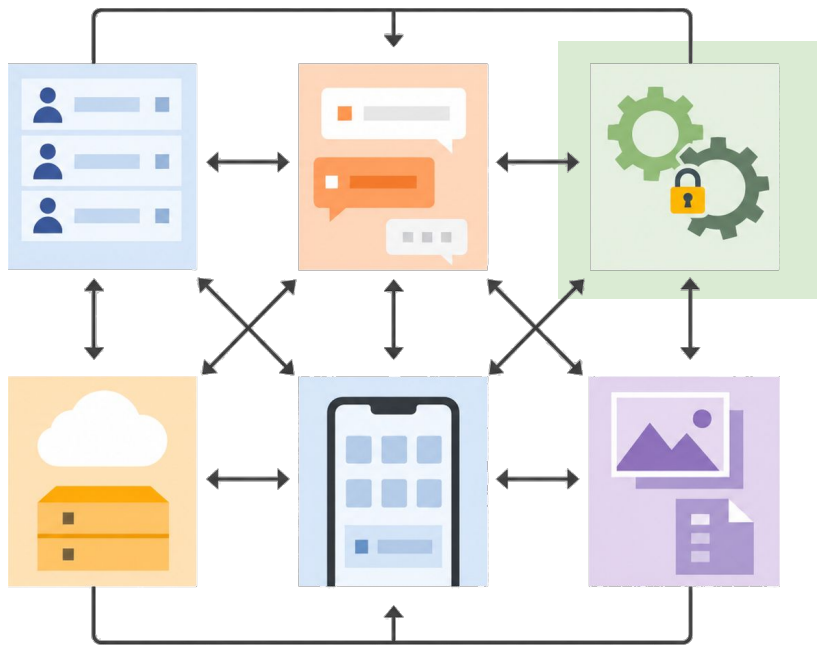
...but only if they captured the system in enough detail



...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.

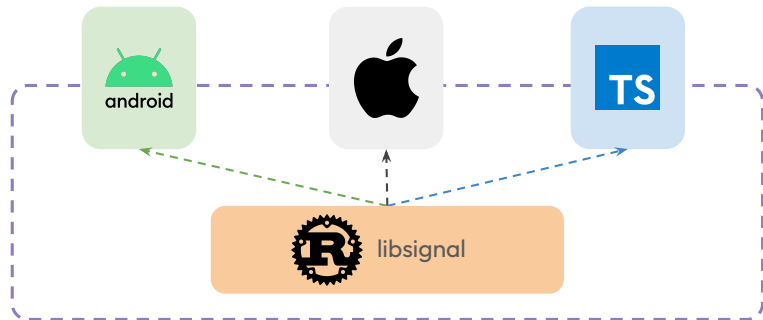
Cryptography should be audited along the rest of the code.

Platform-Dependent Logic

Attack 1 affects Android and Desktop.

Attack 2 affects only Android.

Avoid scattering cryptographic logic across modules and platforms.



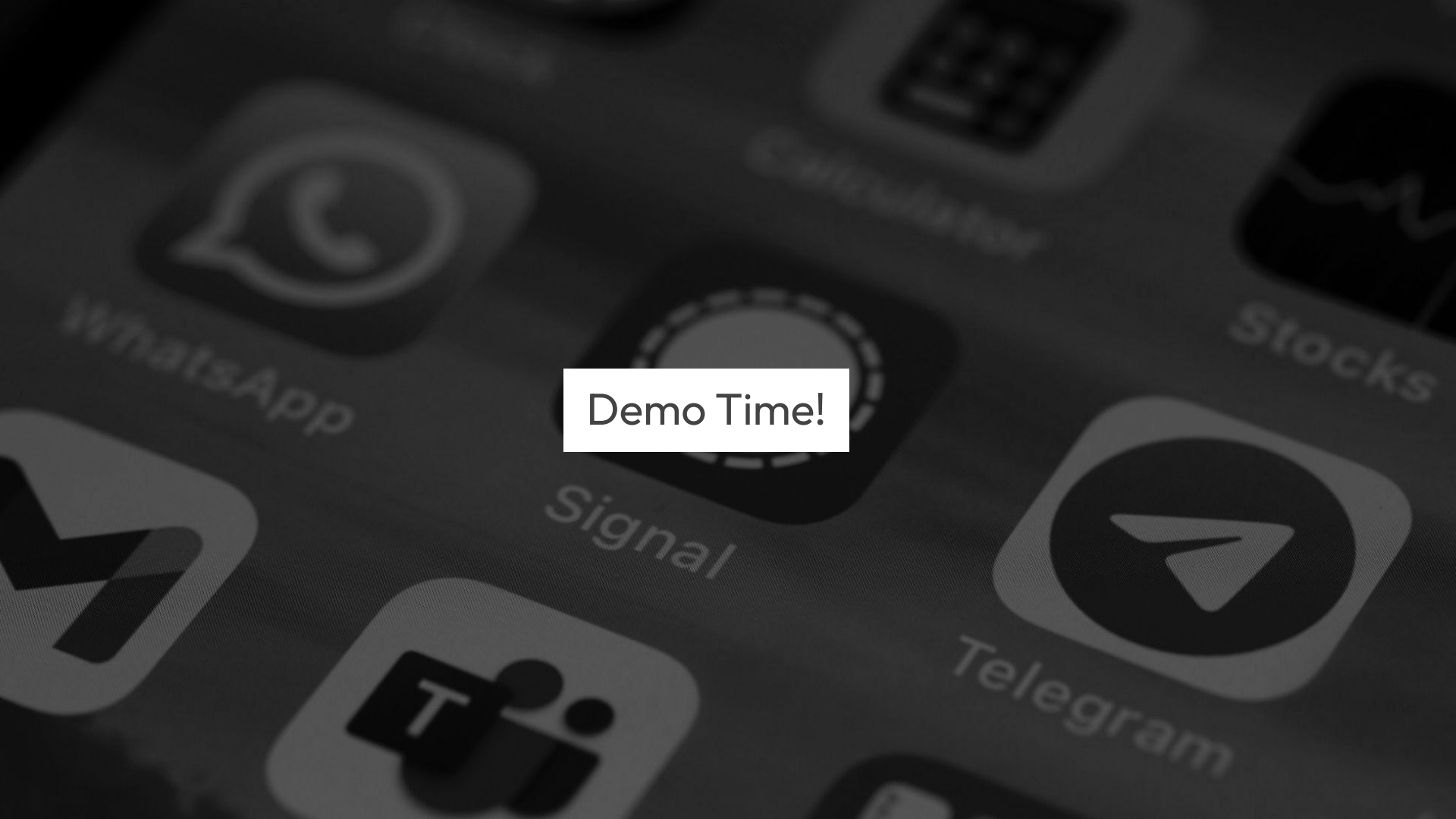
- We analysed Signal in the malicious server setting.
- We found two attacks which violate conversation integrity.
- Signal for iOS was unaffected by both attacks.
Signal Desktop was unaffected by the second attack.
- Both attacks were patched very quickly by the Signal team, whom we thank for the smooth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

kitruong@ethz.ch
<https://kitruong.dev>

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

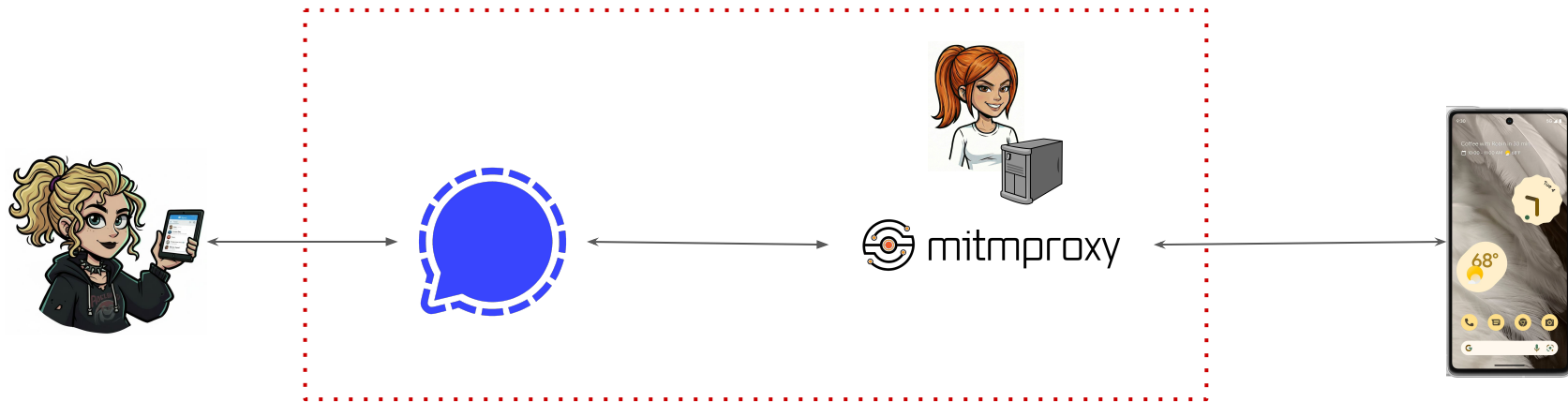


ePrint Link



Demo Time!

Injecting Messages



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