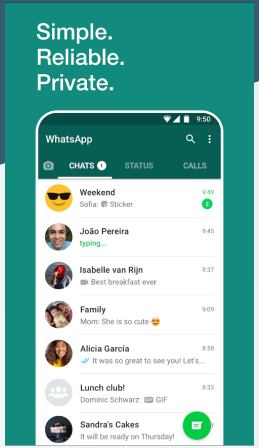
Formal Analysis of Multi-Device Group Messaging in WhatsApp

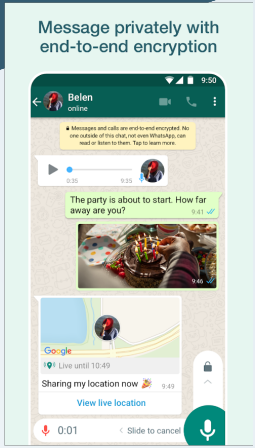
Martin R. Albrecht — King's College London, martin.albrecht@kcl.ac.uk
Benjamin Dowling — King's College London, benjamin.dowling@kcl.ac.uk
Daniel Jones — Royal Holloway, University of London, dan.jones@rhul.ac.uk

Eurocrypt 2025 (Madrid, Spain)

Thursday 8th May 2025

Motivation

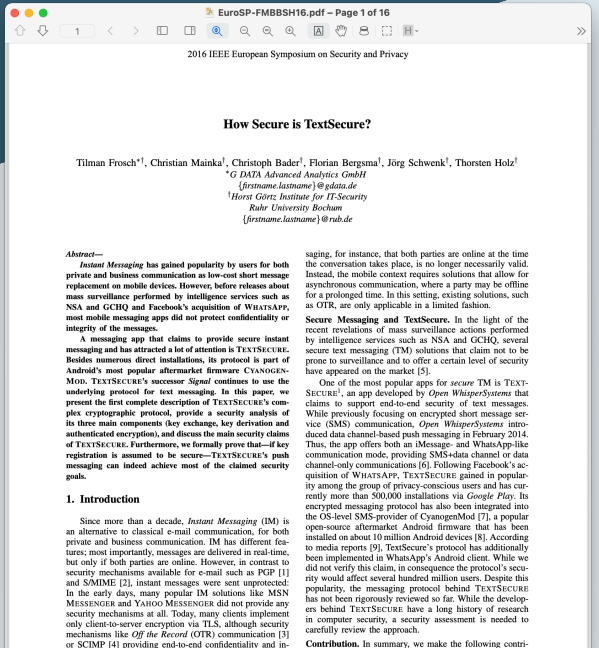




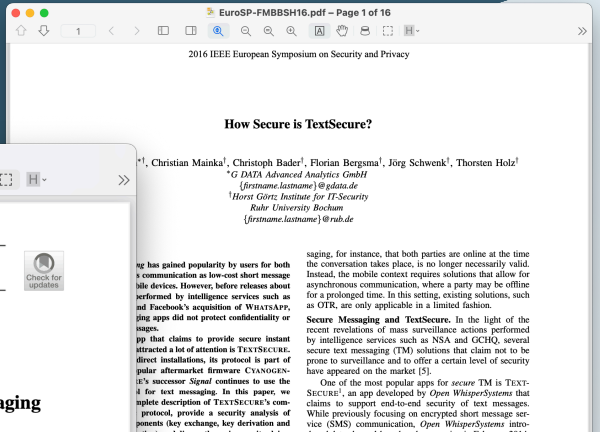
[https://play.google.com/store/apps/details?id=com.whatsapp]

Motivation

Motivation


EuroSP-FMBBSH16.pdf – Page 1 of 16
2016 IEEE European Symposium on Security and Privacy
How Secure is TextSecure?
Tilman Frosch*†, Christian Mainka†, Christoph Bader†, Florian Bergsma†, Jörg Schwenk†, Thorsten Holz†
*G DATA Advanced Analytics GmbH
{firstname.lastname}@gdata.de
†Horst Görtz Institute for IT-Security
Ruhr University Bochum
{firstname.lastname}@rub.de
Abstract—
Instant Messaging has gained popularity by users for both private and business communication as low-cost short message replacement on mobile devices. However, before releases about mass surveillance performed by intelligence services such as NSA and GCHQ and Facebook's acquisition of WhatsApp, most mobile messaging apps did not protect confidentiality or integrity of the messages.
A messaging app that claims to provide secure instant messaging and has attracted a lot of attention is TextSecure. Besides numerous direct installations, its protocol is part of Android's most popular aftermarket firmware CyanogenMod. TextSecure's successor Signal continues to use the underlying protocol for text messaging. In this paper, we present the first complete description of TextSecure's complex cryptographic protocol, provide a security analysis of its three main components (key exchange, key derivation and authenticated encryption), and discuss the main security claims of TextSecure. Furthermore, we formally prove that—if key registration is assumed to be secure—TextSecure's push messaging can indeed achieve most of the claimed security goals.
1. Introduction
Since more than a decade, Instant Messaging (IM) is an alternative to classical e-mail communication, for both private and business communication. IM has different features; most importantly, messages are delivered in real-time, but only if both parties are online. However, in contrast to security mechanisms available for e-mail such as PGP [1] and S/MIME [2], instant messages were sent unprotected: In the early days, many popular IM solutions like MSN Messenger and Yahoo Messenger did not provide any security mechanisms at all. Today, many clients implement only client-to-server encryption via TLS, although security mechanisms like Off the Record (OTR) communication [3] or SCIMP [4] providing end-to-end confidentiality and insaging, for instance, that both parties are online at the time the conversation takes place, is no longer necessarily valid. Instead, the mobile context requires solutions that allow for asynchronous communication, where a party may be offline for a prolonged time. In this setting, existing solutions, such as OTR, are only applicable in a limited fashion.
Secure Messaging and TextSecure. In the light of the recent revelations of mass surveillance actions performed by intelligence services such as NSA and GCHQ, several secure text messaging (TM) solutions that claim not to be prone to surveillance and to offer a certain level of security have appeared on the market [5].
One of the most popular apps for secure TM is TextSecure[1], an app developed by Open WhisperSystems that claims to support end-to-end security of text messages. While previously focusing on encrypted short message service (SMS) communication, Open WhisperSystems introduced data channel-based push messaging in February 2014. Thus, the app offers both an iMessage- and WhatsApp-like communication mode, providing SMS-data channel or data channel-only communications [6]. Following Facebook's acquisition of WhatsApp, TextSecure gained in popularity among the group of privacy-conscious users and has currently more than 500,000 installations via Google Play. Its encrypted messaging protocol has also been integrated into the OS-level SMS-provider of CyanogenMod [7], a popular open-source aftermarket Android firmware that has been installed on about 10 million Android devices [8]. According to media reports [9], TextSecure's protocol has additionally been implemented in WhatsApp's Android client. While we did not verify this claim, in consequence the protocol's security would affect several hundred million users. Despite this popularity, the messaging protocol behind TextSecure has not been rigorously reviewed so far. While the developers behind TextSecure have a long history of research in computer security, a security assessment is needed to carefully review the approach.
Contribution. In summary, we make the following contri-

Motivation



EuroSP-FMBBSH16.pdf – Page 1 of 16

2016 IEEE European Symposium on Security and Privacy

How Secure is TextSecure?

*†, Christian Mainka†, Christoph Bader†, Florian Bergsma†, Jörg Schwenk†, Thorsten Holz†

*G DATA Advanced Analytics GmbH
{firstname.lastname}@gdata.de
†Horst Görtz Institute for IT-Security
Ruhr University Bochum
{firstname.lastname}@rub.de

g has gained popularity by users for both s communication as low-cost short message ile devices. However, before releases about erformed by intelligence services such as nd Facebook's acquisition of WhatsApp, ing apps did not protect confidentiality or sages.

pp that claims to provide secure instant attracted a lot of attention is TextSecure. direct installations, its protocol is part of pular aftermarket firmware CyanogenE's successor *Signal* continues to use the d for text messaging. In this paper, we plete description of TextSecure's com- protocol, provide a security analysis of ponents (key exchange, key derivation and ption), and discuss the main security claims urthermore, we formally prove that—if key med to be secure—TextSecure's push ed achieve most of the claimed security

saging, for instance, that both parties are online at the time the conversation takes place, is no longer necessarily valid. Instead, the mobile context requires solutions that allow for asynchronous communication, where a party may be offline for a prolonged time. In this setting, existing solutions, such as OTR, are only applicable in a limited fashion.

Secure Messaging and TextSecure. In the light of the recent revelations of mass surveillance actions performed by intelligence services such as NSA and GCHQ, several secure text messaging (TM) solutions that claim not to be prone to surveillance and to offer a certain level of security have appeared on the market [5].

One of the most popular apps for *secure* TM is TextSecure[1], an app developed by *Open WhisperSystems* that claims to support end-to-end security of text messages. While previously focusing on encrypted short message service (SMS) communication, *Open WhisperSystems* introduced data channel-based push messaging in February 2014. Thus, the app offers both an iMessage- and WhatsApp-like communication mode, providing SMS-data channel or data channel-only communications [6]. Following Facebook's acquisition of WhatsApp, TextSecure gained in popularity among the group of privacy-conscious users and has currently more than 500,000 installations via *Google Play*. Its encrypted messaging protocol has also been integrated into the OS-level SMS-provider of CyanogenMod [7], a popular open-source aftermarket Android firmware that has been installed on about 10 million Android devices [8]. According to media reports [9], TextSecure's protocol has additionally been implemented in WhatsApp's Android client. While we did not verify this claim, in consequence the protocol's security would affect several hundred million users. Despite this popularity, the messaging protocol behind TextSecure has not been rigorously reviewed so far. While the developers behind TextSecure have a long history of research in computer security, a security assessment is needed to carefully review the approach.

Contribution. In summary, we make the following contri-

JC-CCDGS20.pdf (page 1 of 70)

1914

J Cryptol (2020) 33:1914–1983
https://doi.org/10.1007/s00145-020-09360-1

Journal of CRYPTOLOGY

Check for updates

A Formal Security Analysis of the Signal Messaging Protocol

Katriel Cohn-Gordon
Oxford, UK
me@katriel.co.uk

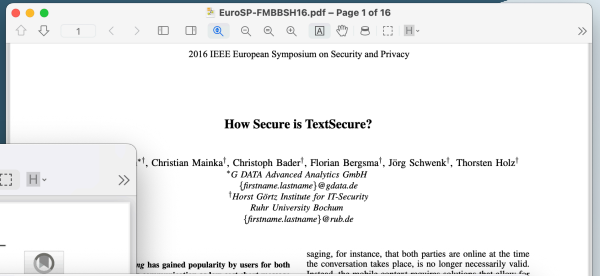
Cas Cremers
CISPA Helmholtz Center for Information Security, Saarbrücken, Germany
cremers@cispa.saarland

Benjamin Dowling
ETH Zürich, Zurich, Switzerland
benjamin.dowling@inf.ethz.ch

Luke Garratt
Cisco Systems, San Jose, USA
lgarratt@cisco.com

n a decade, *Instant Messaging* (IM) is lassical e-mail communication, for both ss communication. IM has different fea- ntly, messages are delivered in real-time, arties are online. However, in contrast to ns available for e-mail such as PGP [1] instant messages were sent unprotected: many popular IM solutions like MSN YAHOO Messenger did not provide any ns at all. Today, many clients implement r encryption via TLS, although security *ff the Record* (OTR) communication [3] viding end-to-end confidentiality and in-

Motivation


EuroSP-FMBBSH16.pdf – Page 1 of 16

2016 IEEE European Symposium on Security and Privacy

How Secure is TextSecure?

*†, Christian Mainka†, Christoph Bader†, Florian Bergsma†, Jörg Schwenk†, Thorsten Holz†
*G DATA Advanced Analytics GmbH
{firstname.lastname}@gdata.de
†Horst Görtz Institute for IT-Security
Ruhr University Bochum
{firstname.lastname}@rub.de

saging, for instance, that both parties are online at the time the conversation takes place, is no longer necessarily valid.


JC-CCDGS20.pdf (page 1 of 70)

J Cryptol (2020) 33:1914–1983
https://doi.org/10.1007/s00145-020-09360-1

Journal of CRYPTOLOGY

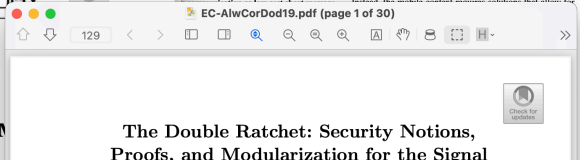
A Formal Security Analysis of the Signal M Protocol

Katriel Cohn-Gordon
Oxford, UK
me@katriel.co.uk

Cas Cremers
CISPA Helmholtz Center for Information Security, Saarbrücken, Ge
cremers@cispa.saarland

Benjamin Dowling
ETH Zürich, Zurich, Switzerland
benjamin.dowling@inf.ethz.ch

Luke Garratt
Cisco Systems, San Jose, USA
lgarratt@cisco.com


EC-AlwCorDod19.pdf (page 1 of 30)

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

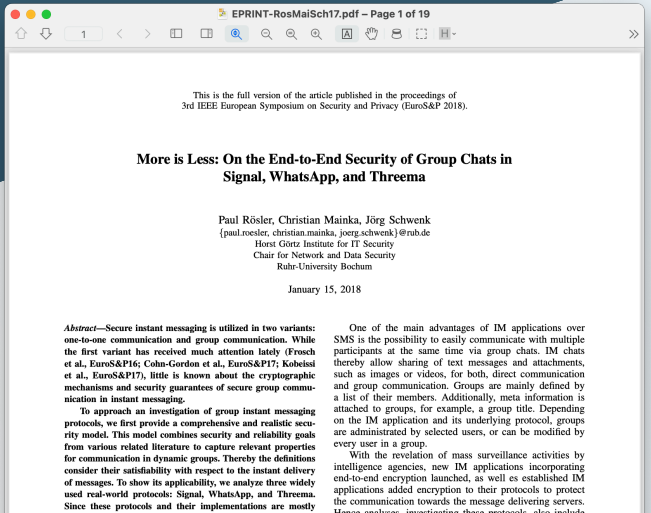
Joël Alwen[2(✉)], Sandro Coretti[1], and Yevgeniy Dodis[1]

[1] New York University, New York, USA
{corettis,dodis}@nyu.edu
[2] Wickr Inc., San Francisco, USA
jalwen@wickr.com

Abstract. Signal is a famous secure messaging protocol used by billions of people, by virtue of many secure text messaging applications including

Motivation

Motivation


EPRINT-RosMaiSch17.pdf – Page 1 of 19

This is the full version of the article published in the proceedings of
3rd IEEE European Symposium on Security and Privacy (EuroS&P 2018).

More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema

Paul Rösler, Christian Mainka, Jörg Schwenk
{paul.roesler, christian.mainka, joerg.schwenk}@rub.de
Horst Görtz Institute for IT Security
Chair for Network and Data Security
Ruhr-University Bochum

January 15, 2018

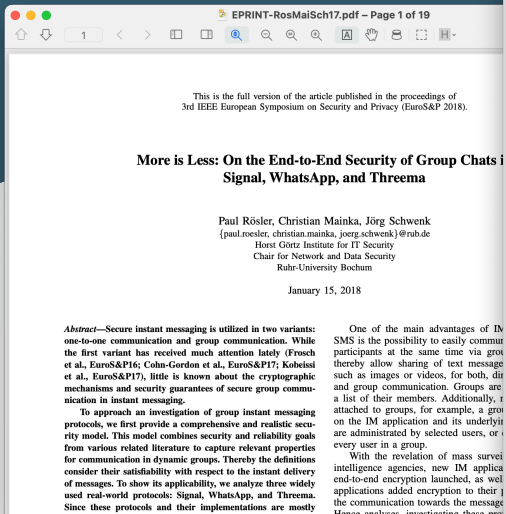
***Abstract*—Secure instant messaging is utilized in two variants: one-to-one communication and group communication. While the first variant has received much attention lately (Frosch et al., EuroS&P16; Cohn-Gordon et al., EuroS&P17; Kobeissi et al., EuroS&P17), little is known about the cryptographic mechanisms and security guarantees of secure group communication in instant messaging.**

To approach an investigation of group instant messaging protocols, we first provide a comprehensive and realistic security model. This model combines security and reliability goals from various related literature to capture relevant properties for communication in dynamic groups. Thereby the definitions consider their satisfiability with respect to the instant delivery of messages. To show its applicability, we analyze three widely used real-world protocols: Signal, WhatsApp, and Threema. Since these protocols and their implementations are mostly

One of the main advantages of IM applications over SMS is the possibility to easily communicate with multiple participants at the same time via group chats. IM chats thereby allow sharing of text messages and attachments, such as images or videos, for both, direct communication and group communication. Groups are mainly defined by a list of their members. Additionally, meta information is attached to groups, for example, a group title. Depending on the IM application and its underlying protocol, groups are administrated by selected users, or can be modified by every user in a group.

With the revelation of mass surveillance activities by intelligence agencies, new IM applications incorporating end-to-end encryption launched, as well es established IM applications added encryption to their protocols to protect the communication towards the message delivering servers. Hence analyses, investigating these protocols, also include

Motivation


EPRINT-RosMaiSch17.pdf – Page 1 of 19

This is the full version of the article published in the proceedings of
3rd IEEE European Symposium on Security and Privacy (EuroS&P 2018).

**More is Less: On the End-to-End Security of Group Chats i
Signal, WhatsApp, and Threema**

Paul Rösler, Christian Mainka, Jörg Schwenk
{paul.roesler, christian.mainka, joerg.schwenk}@rub.de
Horst Görtz Institute for IT Security
Chair for Network and Data Security
Ruhr-University Bochum

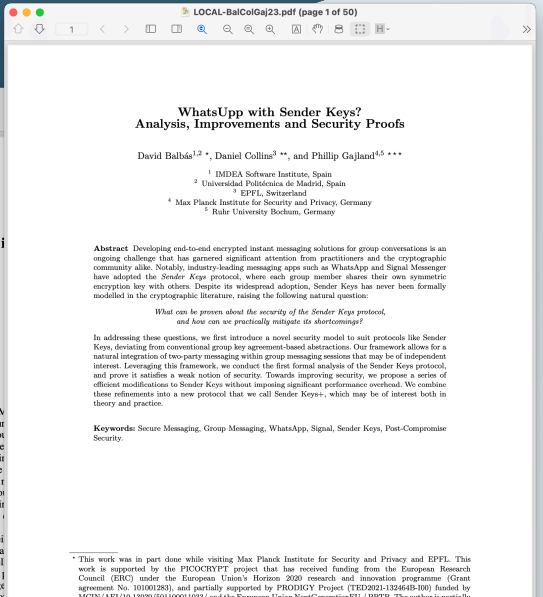
January 15, 2018

***Abstract*—Secure instant messaging is utilized in two variants: one-to-one communication and group communication. While the first variant has received much attention lately (Frosch et al., EuroS&P16; Cohn-Gordon et al., EuroS&P17; Kobeissi et al., EuroS&P17), little is known about the cryptographic mechanisms and security guarantees of secure group communication in instant messaging.**

To approach an investigation of group instant messaging protocols, we first provide a comprehensive and realistic security model. This model combines security and reliability goals from various related literature to capture relevant properties for communication in dynamic groups. Thereby the definitions consider their satisfiability with respect to the instant delivery of messages. To show its applicability, we analyze three widely used real-world protocols: Signal, WhatsApp, and Threema. Since these protocols and their implementations are mostly

One of the main advantages of IM
SMS is the possibility to easily commun
participants at the same time via grou
thereby allow sharing of text message
such as images or videos, for both, dir
and group communication. Groups are
a list of their members. Additionally, r
attached to groups, for example, a grou
on the IM application and its underlyin
are administrated by selected users, or
every user in a group.

With the revelation of mass survei
intelligence agencies, new IM applica
end-to-end encryption launched, as wel
applications added encryption to their
the communication towards the message


LOCAL-BalColGaj23.pdf (page 1 of 50)

**WhatsUpp with Sender Keys?
Analysis, Improvements and Security Proofs**

David Balbás[1,2] *, Daniel Collins[3] **, 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

Abstract Developing end-to-end encrypted instant messaging solutions for group conversations is an ongoing challenge that has garnered significant attention from practitioners and the cryptographic community alike. Notably, industry-leading messaging apps such as WhatsApp and Signal Messenger have adopted the *Sender Keys* protocol, where each group member shares their own symmetric encryption key with others. Despite its widespread adoption, Sender Keys has never been formally modelled in the cryptographic literature, raising the following natural question:

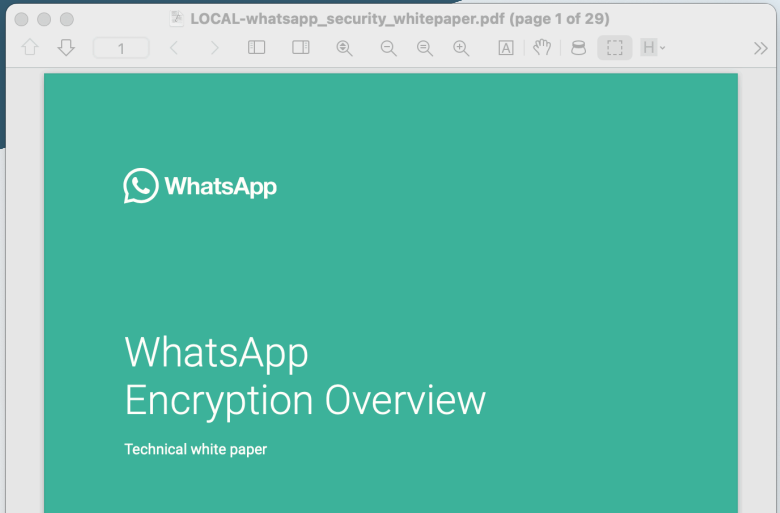
What can be proven about the security of the Sender Keys protocol, and how can we practically mitigate its shortcomings?

In addressing these questions, we first introduce a novel security model to suit protocols like Sender Keys, deviating from conventional group key agreement-based abstractions. Our framework allows for a natural integration of two-party messaging within group messaging sessions that may be of independent interest. Leveraging this framework, we conduct the first formal analysis of the Sender Keys protocol, and prove it satisfies a weak notion of security. Towards improving security, we propose a series of efficient modifications to Sender Keys without imposing significant performance overhead. We combine these refinements into a new protocol that we call Sender Keys+, which may be of interest both in theory and practice.

Keywords: Secure Messaging, Group Messaging, WhatsApp, Signal, Sender Keys, Post-Compromise Security.

* This work was in part done while visiting Max Planck Institute for Security and Privacy and EPFL. This work is supported by the PICOCRYPT project that has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (Grant agreement No. 101001283), and partially supported by PRODIGY Project (TED2021-132464B-I00) funded by

Motivation


LOCAL-whatsapp_security_whitepaper.pdf (page 1 of 29)
WhatsApp
WhatsApp
Encryption Overview
Technical white paper

① How does WhatsApp implement multi-device group messaging?

① How does WhatsApp *implement* multi-device group messaging?

② What security guarantees does it provide?

Our Work

1. Comprehensive description of multi-device group messaging in WhatsApp.
   - Group Messaging
   - Pairwise Channels (incl. Session Management)
   - Device Management
   - History Sharing

Our Work

1. Comprehensive description of multi-device group messaging in WhatsApp.
   - Group Messaging
   - Pairwise Channels (incl. Session Management)
   - Device Management
   - History Sharing

   → Sourced by reverse-engineering its client software.

Our Work

1. Comprehensive description of multi-device group messaging in WhatsApp.
   - Group Messaging
   - Pairwise Channels (incl. Session Management)
   - Device Management
   - History Sharing

   → Sourced by reverse-engineering its client software.
2. Propose a variant of *Device-Oriented Group Messaging* to capture device revocation.

Our Work

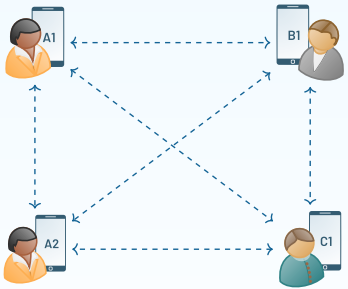
1. Comprehensive description of multi-device group messaging in WhatsApp.
   - Group Messaging
   - Pairwise Channels (incl. Session Management)
   - Device Management
   - History Sharing

   → Sourced by reverse-engineering its client software.
2. Propose a variant of *Device-Oriented Group Messaging* to capture device revocation.
3. *State and prove* WhatsApp's security guarantees within the DOGM w/ revocations model.

Group Messaging

Group Messaging with Sender Keys

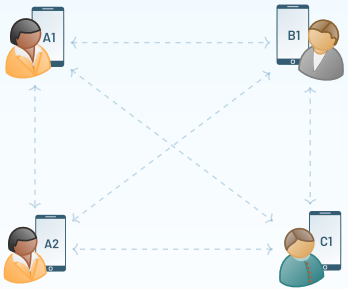
(1) Initialisation



Group Messaging with Sender Keys

(1) Initialisation

(a) Generate Sender Keys session.



Group Messaging with Sender Keys

(1) Initialisation

(a) Generate Sender Keys session.

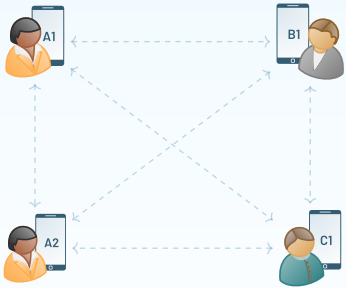
$z \leftarrow 0$ // message index

$ck \leftarrow\$ \{0,1\}^{256}$ // symmetric ratchet

$(gsk, gpk) \leftarrow\$ \text{XDH.Gen}()$ // signing key pair

$ust_{out,A1} \leftarrow (usid, z, ck, gsk)$ // outbound session

$ust_{in,A1} \leftarrow (usid, z, ck, gpk)$ // inbound session

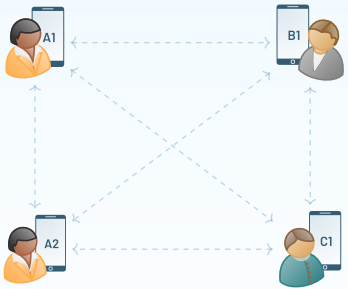


Group Messaging with Sender Keys

(1) Initialisation

(a) Generate Sender Keys session.

(b) Send inbound session over two-party channels.



Group Messaging with Sender Keys

(1) Initialisation

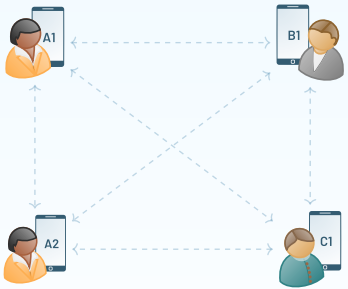
(a) Generate Sender Keys session.

(b) Send inbound session over two-party channels.

$(pst_{C1}, c_{C1}) \leftarrow \text{PAIR.Enc}(pst_{C1}, ust_{in,A1})$

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in,A1})$

$(pst_{A2}, c_{A2}) \leftarrow \text{PAIR.Enc}(pst_{A2}, ust_{in,A1})$



Group Messaging with Sender Keys

(1) Initialisation

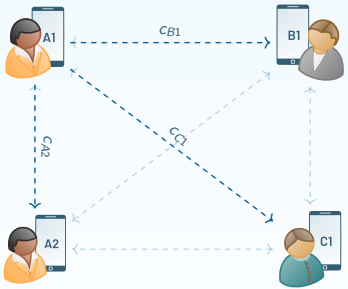
(a) Generate Sender Keys session.

(b) Send inbound session over two-party channels.

$(pst_{C1}, c_{C1}) \leftarrow \text{PAIR.Enc}(pst_{C1}, ust_{in,A1})$

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in,A1})$

$(pst_{A2}, c_{A2}) \leftarrow \text{PAIR.Enc}(pst_{A2}, ust_{in,A1})$



Group Messaging with Sender Keys

(1) Initialisation

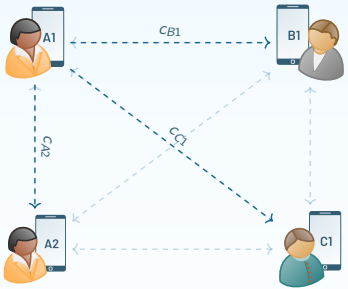
(a) Generate Sender Keys session.

(b) Send inbound session over two-party channels.

$(pst_{C1}, c_{C1}) \leftarrow \text{PAIR.Enc}(pst_{C1}, ust_{in,A1})$

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in,A1})$

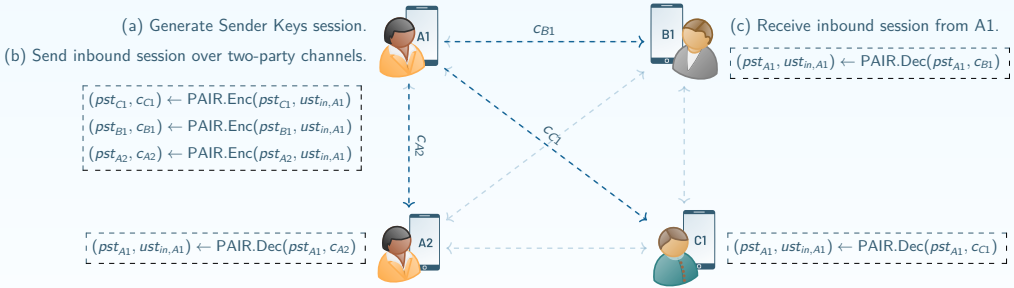
$(pst_{A2}, c_{A2}) \leftarrow \text{PAIR.Enc}(pst_{A2}, ust_{in,A1})$



(c) Receive inbound session from A1.

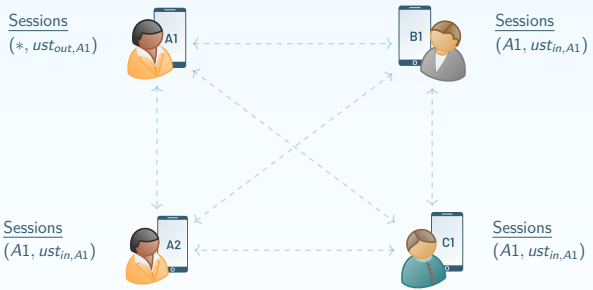
Group Messaging with Sender Keys

(1) Initialisation



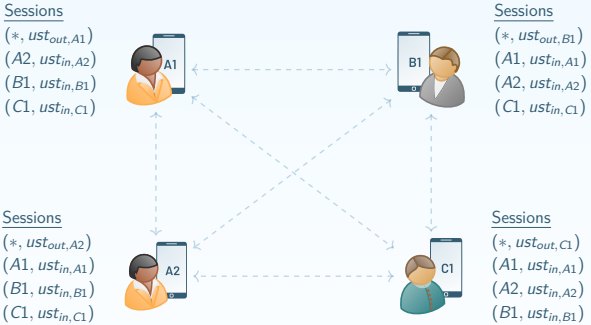
Group Messaging with Sender Keys

(1) Initialisation



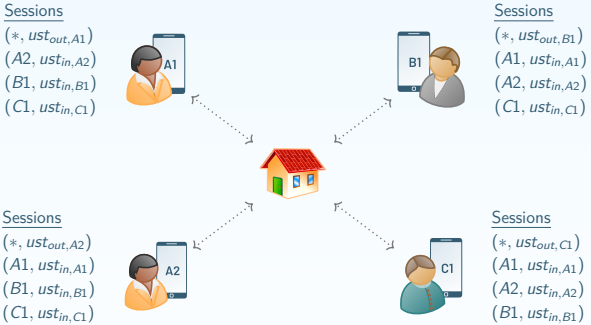
Group Messaging with Sender Keys

(1) Initialisation



Group Messaging with Sender Keys

(2) Messaging



Group Messaging with Sender Keys

(2) Messaging

(a) Increment message counter.

$z \leftarrow z + 1$

Sessions

$(*, ust_{out,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,B1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,A2})$
$(A1, ust_{in,A1})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,C1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$

Group Messaging with Sender Keys

(2) Messaging

(a) Increment message counter.

(b) Derive key material and ratchet symmetric key.

$mk \leftarrow \mathrm{HMAC}(ck, \texttt{0x01})$

$ck \leftarrow \mathrm{HMAC}(ck, \texttt{0x02})$

Sessions

$(*, ust_{out,A1})$

$(A2, ust_{in,A2})$

$(B1, ust_{in,B1})$

$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,B1})$

$(A1, ust_{in,A1})$

$(A2, ust_{in,A2})$

$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,A2})$

$(A1, ust_{in,A1})$

$(B1, ust_{in,B1})$

$(C1, ust_{in,C1})$

Sessions

$(*, ust_{out,C1})$

$(A1, ust_{in,A1})$

$(A2, ust_{in,A2})$

$(B1, ust_{in,B1})$

Group Messaging with Sender Keys

(2) Messaging

(a) Increment message counter.
(b) Derive key material and ratchet symmetric key.
(c) Encrypt then sign.

$k \leftarrow \mathrm{HKDF}(\varnothing, mk, \texttt{WhisperGroup}, 50\mathrm{B})$
$iv, ek \leftarrow k[0\mathrm{B} \rightarrow 15\mathrm{B}], k[16\mathrm{B} \rightarrow 47\mathrm{B}]$
$c \leftarrow \mathrm{AES\text{-}CBC.Enc}(ek, iv, m)$
$\sigma_U \leftarrow \mathrm{XEd.Sign}(gsk, (usid, z, c))$
$c_U \leftarrow (usid, z, c, \sigma_U)$

Sessions
$(*, ust_{out,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

Sessions
$(*, ust_{out,B1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(C1, ust_{in,C1})$

Sessions
$(*, ust_{out,A2})$
$(A1, ust_{in,A1})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

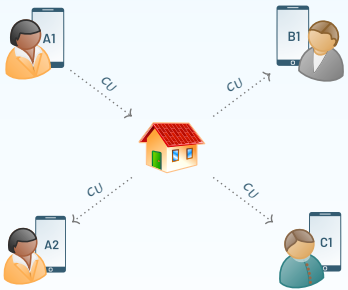
Sessions
$(*, ust_{out,C1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$

Group Messaging with Sender Keys

(2) Messaging

(a) Increment message counter.
(b) Derive key material and ratchet symmetric key.
(c) Encrypt then sign.
(d) Server distributes one ctxt.

Sessions
$(*, ust_{out,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$



Sessions
$(*, ust_{out,B1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(C1, ust_{in,C1})$

Sessions
$(*, ust_{out,A2})$
$(A1, ust_{in,A1})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

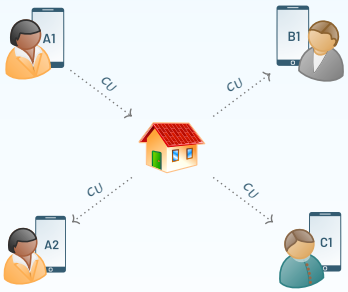
Sessions
$(*, ust_{out,C1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$

Group Messaging with Sender Keys

(2) Messaging

(a) Increment message counter.
(b) Derive key material and ratchet symmetric key.
(c) Encrypt then sign.
(d) Server distributes one ctxt.

Sessions
$(*, ust_{out,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$



Sessions
$(*, ust_{out,B1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(C1, ust_{in,C1})$

(e) Decrypt with inbound session.

Sessions
$(*, ust_{out,A2})$
$(A1, ust_{in,A1})$
$(B1, ust_{in,B1})$
$(C1, ust_{in,C1})$

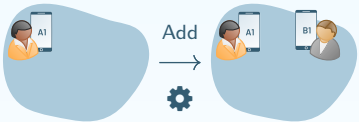
Sessions
$(*, ust_{out,C1})$
$(A1, ust_{in,A1})$
$(A2, ust_{in,A2})$
$(B1, ust_{in,B1})$

Group Messaging with Sender Keys

(3) Membership Changes

Group Messaging with Sender Keys

(3) Membership Changes



- Adding device – Share current value of the receiving session.
  → Symmetric ratchet protects old messages.

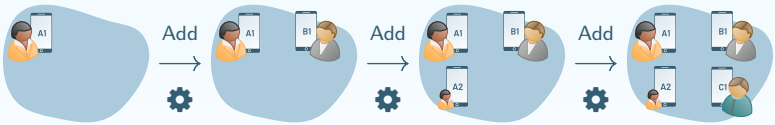
Group Messaging with Sender Keys

(3) Membership Changes

- Adding device – Share current value of the receiving session.
  → Symmetric ratchet protects old messages.

Group Messaging with Sender Keys

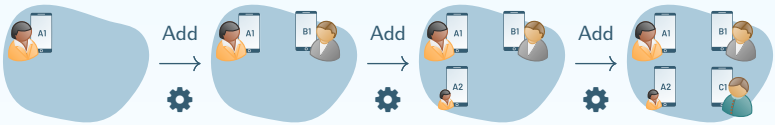
(3) Membership Changes



- Adding device – Share current value of the receiving session.
  → Symmetric ratchet protects old messages.

Group Messaging with Sender Keys

(3) Membership Changes



- Adding device – Share current value of the receiving session.
  → Symmetric ratchet protects old messages.

Group Messaging with Sender Keys

(3) Membership Changes

- Removing device – Generate and distribute a new Sender Keys session.
  → Two-party channels protect new session, which protects new messages.

Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in,A1})$

$(pst_{A1}, ust_{in,A1}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

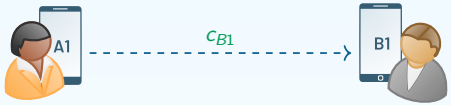
Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:

Intuitively, we expect that

Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:



Intuitively, we expect that

> if the two-party channel used to distribute a session was {confidential, authentic},
> so too should be the Sender Keys sessions and resulting messages.

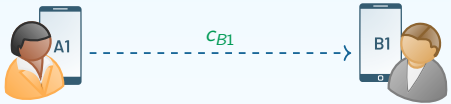
Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:

This should apply to state compromise, too.

Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:



$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in,A1})$

$(pst_{A1}, ust_{in,A1}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

This should apply to state compromise, too.

For post-compromise security, Sender Keys sessions are rotated at regular intervals.

Security of Sender Keys (in Theory)

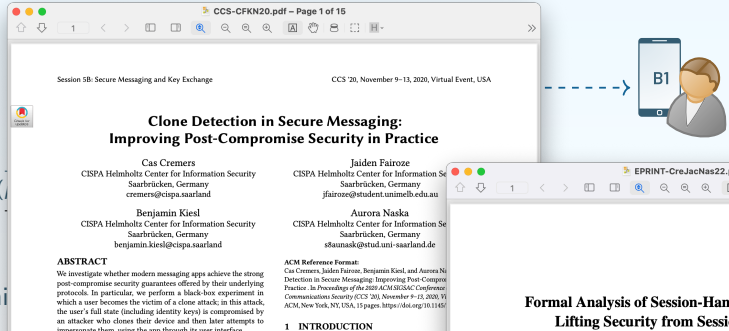
Consider Alice sharing her Sender Keys session with Bob:

This should apply to state compromise, too.

For post-compromise security, Sender Keys sessions are rotated at regular intervals.
For forward secrecy, confidentiality of two-party channels protects earlier versions of Sender Keys session.

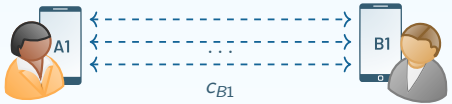
Security of Sender Keys (in Theory)

Consider Alice sharing her Sender Keys session with Bob:



Security of Sender Keys (in *Practice*)

Consider Alice sharing her Sender Keys session with Bob:



PST (A1)

pst_0

pst_1

$\cdots$

pst_{39}

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in})$

$(pst_{A1}, ust_{in}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

Security of Sender Keys (in *Practice*)

Consider Alice sharing her Sender Keys session with Bob:

PST (A1)

pst_0

pst_1

...

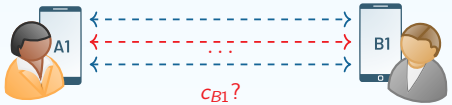
pst_{39}

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in})$

$(pst_{A1}, ust_{in}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

Security of Sender Keys (in *Practice*)

Consider Alice sharing her Sender Keys session with Bob:



PST (A1)

pst_0

pst_1

...

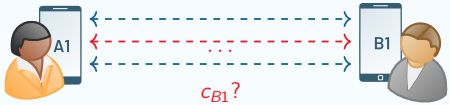
pst_{39}

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in})$

$(pst_{A1}, ust_{in}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

Security of Sender Keys (in *Practice*)

Consider Alice sharing her Sender Keys session with Bob:



PST (A1)	SKST (A1)
pst_0	$ust_{in,0}$
pst_1	$ust_{in,1}$
...	$ust_{in,2}$
pst_{39}	$ust_{in,3}$
	$ust_{in,4}$

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{B1}, ust_{in})$

$(pst_{A1}, ust_{in}) \leftarrow \text{PAIR.Dec}(pst_{A1}, c_{B1})$

Security of Sender Keys (in *Practice*)

Post-compromise security is limited:

Security of Sender Keys (in *Practice*)

Post-compromise security is limited:

- Recovery requires compromised pairwise session to be *ejected* from local cache of the 40 most recent pairwise sessions.

Security of Sender Keys (in *Practice*)

Post-compromise security is limited:

- Recovery requires compromised pairwise session to be *ejected* from local cache of the 40 most recent pairwise sessions.
  and all Sender Keys sessions distributed over it to be ejected from local cache of 5 most recent sessions.

Security of Sender Keys (in *Practice*)

Post-compromise security is limited:

- Recovery requires compromised pairwise session to be *ejected* from local cache of the 40 most recent pairwise sessions.
  and all Sender Keys sessions distributed over it to be ejected from local cache of 5 most recent sessions.
- Adversary's control over the network, coupled with (partial) state compromise, enables *occasionally active* attacker to maintain compromise indefinitely.

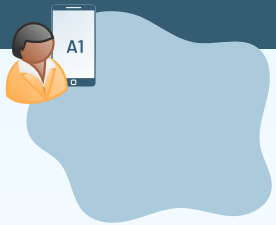
Security of Sender Keys (in *Practice*)

Does there exist any deployed messaging application that achieves PCS in practice?

$\rightarrow$ WhatsApp provides a compelling alternative: device revocation.

Device Management

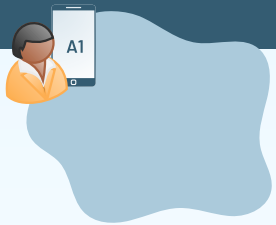
Device Management in WhatsApp



- Users are represented by their primary device.

$$isk_p, ipk_p \leftarrow\$ \text{ XDH.Gen}()$$

Device Management in WhatsApp



- Users are represented by their primary device.

$$isk_p, ipk_p \leftarrow\$ \text{ XDH.Gen}()$$

Device Management in WhatsApp

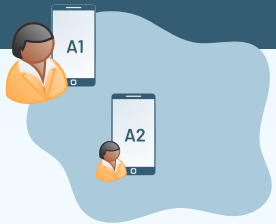
- Users are represented by their primary device.

$$isk_p, ipk_p \leftarrow\$ \text{ XDH.Gen}()$$

- A user may add a number of companion devices.

$$isk_c, ipk_c \leftarrow\$ \text{ XDH.Gen}()$$

Device Management in WhatsApp



- Users are represented by their primary device.

$$isk_p, ipk_p \leftarrow\$ \text{ XDH.Gen}()$$

- A user may add a number of companion devices.

$$isk_c, ipk_c \leftarrow\$ \text{ XDH.Gen}()$$

- Only the primary device may add or remove companion devices.

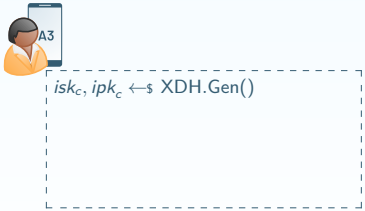
Device Management in WhatsApp

Linking a new companion device:

Device Management in WhatsApp

Linking a new companion device:

1. New companion device generates identity key.



Device Management in WhatsApp

Linking a new companion device:

1. New companion device generates identity key.
2. Verify keys out-of-band (e.g. through a QR code).

$isk_c, ipk_c \leftarrow\$ \text{XDH.Gen}()$

Device Management in WhatsApp

Linking a new companion device:

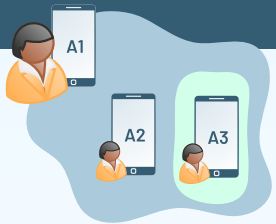
1. New companion device generates identity key.
2. Verify keys out-of-band (e.g. through a QR code).
3. Primary device generates and signs a timestamped linking metadata.

$\sigma_{p \to c} \leftarrow \mathrm{XEd.Sign}(isk_p, \texttt{0x0600} \parallel time_1 \parallel ipk_p \parallel ipk_c)$

$dr \leftarrow (time_1, ipk_p, ipk_c, \sigma_{p \to c})$

$isk_c, ipk_c \leftarrow\!\!\$\ \mathrm{XDH.Gen}()$

Device Management in WhatsApp



Linking a new companion device:

1. New companion device generates identity key.
2. Verify keys out-of-band (e.g. through a QR code).
3. Primary device generates and signs a timestamped linking metadata.
4. Primary device adds companion to their timestamped device list.

$\sigma_{p \to c} \leftarrow \text{XEd.Sign}(isk_p, \texttt{0x0600} \,\|\, time_1 \,\|\, ipk_p \,\|\, ipk_c)$

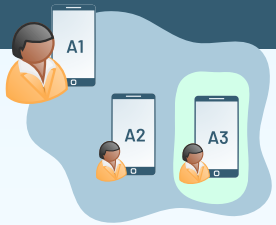
$dr \leftarrow (time_1, ipk_p, ipk_c, \sigma_{p \to c})$

$dl \leftarrow dl \,\|\, [ipk_c]$

$\sigma_{dl} \leftarrow \text{XEd.Sign}(isk_p, \texttt{0x0602} \,\|\, dl \,\|\, time_1)$

$isk_c, ipk_c \leftarrow\!\!\$\ \text{XDH.Gen}()$

Device Management in WhatsApp



Linking a new companion device:

1. New companion device generates identity key.
2. Verify keys out-of-band (e.g. through a QR code).
3. Primary device generates and signs a timestamped linking metadata.
4. Primary device adds companion to their timestamped device list.
5. Companion device signs the same linking metadata structure.

$\sigma_{p \to c} \leftarrow \text{XEd.Sign}(isk_p, \texttt{0x0600} \parallel time_1 \parallel ipk_p \parallel ipk_c)$

$dr \leftarrow (time_1, ipk_p, ipk_c, \sigma_{p \to c})$

$dl \leftarrow dl \parallel [ipk_c]$

$\sigma_{dl} \leftarrow \text{XEd.Sign}(isk_p, \texttt{0x0602} \parallel dl \parallel time_1)$

$isk_c, ipk_c \leftarrow\!\!\$\ \text{XDH.Gen}()$

$\sigma_{c \to p} \leftarrow \text{XEd.Sign}($

$\quad isk_c, \texttt{0x0601} \parallel time_1 \parallel ipk_p \parallel ipk_c)$

$dr \leftarrow dr \parallel \sigma_{c \to p}$

Device Management in WhatsApp

To revoke a companion device, the primary device:

Device Management in WhatsApp

To revoke a companion device, the primary device:

1. Removes the companion device from the device list.

$dl \leftarrow dl \setminus \{ipk_c\}$

Device Management in WhatsApp

To revoke a companion device, the primary device:

1. Removes the companion device from the device list.
2. Signs the new device list with an updated timestamp.

$dl \leftarrow dl \setminus \{ipk_c\}$

$\sigma_{dl} \leftarrow \text{XEd.Sign}(isk_p, \texttt{0x0602} \,\|\, dl \,\|\, time_2)$

Device Management in WhatsApp

To revoke a companion device, the primary device:

1. Removes the companion device from the device list.
2. Signs the new device list with an updated timestamp.
3. Requests that the server distributes the new device list.

$dl \leftarrow dl \setminus \{ipk_c\}$

$\sigma_{dl} \leftarrow \mathsf{XEd.Sign}(isk_p, \texttt{0x0602} \parallel dl \parallel time_2)$

Device Consistency

WhatsApp guarantees clients have an up-to-date view of each participant's device composition

Device Consistency

WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device*

Device Consistency

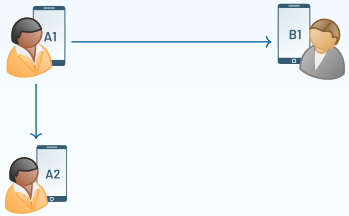
WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).

Device Consistency

WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).

Device Consistency

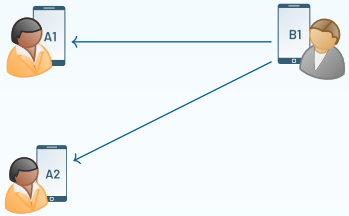
WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).



(a) At start, all parties agree on device-level membership as $\{ A1, A2, B1 \}$.

Device Consistency

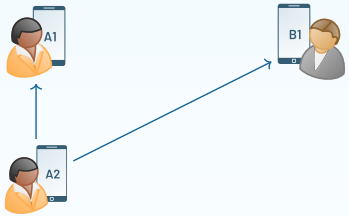
WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).



(a) At start, all parties agree on device-level membership as $\{ A1, A2, B1 \}$.

Device Consistency

WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).



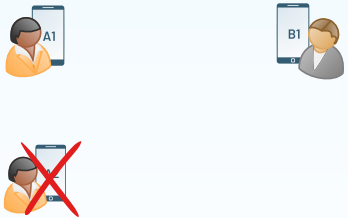
(a) At start, all parties agree on device-level membership as $\{ A1, A2, B1 \}$.

Device Consistency

WhatsApp guarantees clients have an up-to-date view of each participant's device composition

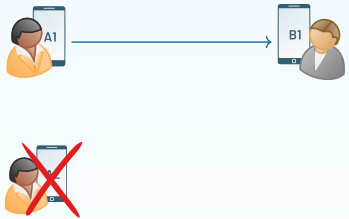
upon reception of the first pairwise message from the respective user's primary device

(or through a companion device that has been notified of the update, and so on…).



Device Consistency

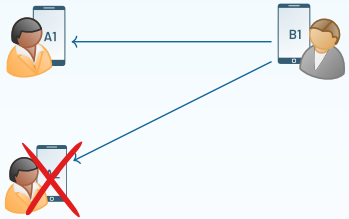
WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on...).



(b) When A1 revokes A2, A1 views device-level membership as $\{ A1, B1 \}$

Device Consistency

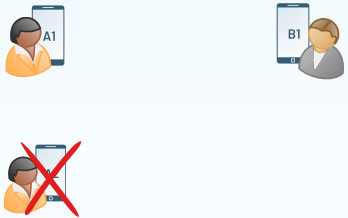
WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).



(b) When A1 revokes A2, A1 views device-level membership as $\{ A1, B1 \}$ but, *initially*, B1 views device-level membership as $\{ A1, A2, B1 \}$.

Device Consistency – *Revocation works*

WhatsApp guarantees clients have an up-to-date view of each participant's device composition *upon reception of the first pairwise message from the respective user's primary device* (or through a companion device that has been notified of the update, and so on…).



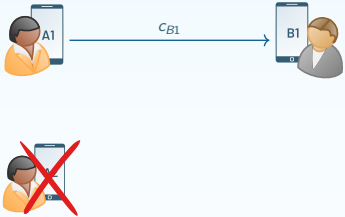
In practice, this is triggered automatically when the primary device rotates their own Sender Keys session.

Device Consistency – *Revocation works*

WhatsApp guarantees clients have an up-to-date view of each participant's device composition
upon reception of the first pairwise message from the respective user's primary device
(or through a companion device that has been notified of the update, and so on…).

(a) Generate Sender Keys session.
(b) Send inbound session over two-party channels (with inline device update).

$(pst_{B1}, c_{B1}) \leftarrow \text{PAIR.Enc}(pst_{A1}, (ust_{in,B1}, \textit{dev-upd}))$



(c) Bob learns of A2's revocation & rotates own Sender Keys session.

In practice, this is triggered automatically when
the primary device rotates their own Sender Keys session.

Group Membership

Server Controls the Group Membership

The server has full control over group membership.[1]

[1] Publicly known issue since at least 2017, see [EUROSP:RosMaiSch18].

Server Controls the Group Membership

The server has full control over group membership.[1]

→ Lack of participant consistency weakens visibility of this control.

[1] Publicly known issue since at least 2017, see [EUROSP:RosMaiSch18].

Lack of Participant Consistency

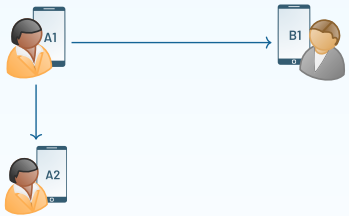
WhatsApp provides no guarantee that devices have a consistent view of a group's membership.

Lack of Participant Consistency

WhatsApp provides no guarantee that devices have a consistent view of a group's membership.

Lack of Participant Consistency

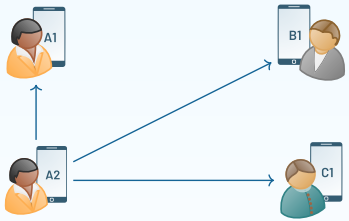
WhatsApp provides no guarantee that devices have a consistent view of a group's membership.



(a) A1 sees user-level group membership as $\{A, B\}$

Lack of Participant Consistency

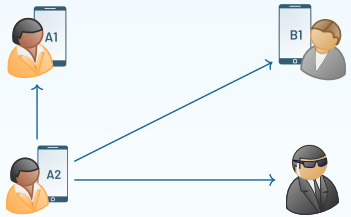
WhatsApp provides no guarantee that devices have a consistent view of a group's membership.



(b) A2 sees user-level group membership as $\{A, B, C\}$

Lack of Participant Consistency

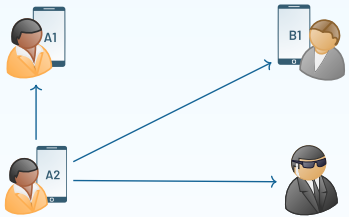
WhatsApp provides no guarantee that devices have a consistent view of a group's membership.



If Alice ensures Claire is not in the group on her phone (A1),
WhatsApp does not guarantee that Claire is not a recipient on Alice's laptop (A2).

Lack of Participant Consistency

WhatsApp provides no guarantee that devices have a consistent view of a group's membership.



Individual devices only have knowledge of the **(user)** recipient list for *messages they send*.

Results

③ Security Analysis

- Provides confidentiality and authentication in the straightforward case.

③ Security Analysis

- Provides confidentiality and authentication in the straightforward case.
- History sharing enables retroactive confidentiality and authenticity breaks, through the *reveal of plaintexts* and *modification of message history* *only when* two-party channels are compromised.

③ Security Analysis

- Provides confidentiality and authentication in the straightforward case.
- History sharing enables retroactive confidentiality and authenticity breaks,
  through the *reveal of plaintexts* and *modification of message history*
  only when two-party channels are compromised.
- *Limited* post-compromise security, against even *occasionally active* adversaries.
  But (!) it is unclear if *any* deployed messaging application achieves PCS in practice.

③ Security Analysis

- Provides confidentiality and authentication in the straightforward case.
- History sharing enables retroactive confidentiality and authenticity breaks,
  through the *reveal of plaintexts* and *modification of message history*
  only when two-party channels are compromised.
- *Limited* post-compromise security, against even *occasionally active* adversaries.
  But (!) it is unclear if *any* deployed messaging application achieves PCS in practice.
- WhatsApp provides a compelling alternative: device revocation.
  Device management provides strong revocation guarantees,
  giving *users* control over compromise recovery.

③ Security Analysis

- Provides confidentiality and authentication in the straightforward case.
- History sharing enables retroactive confidentiality and authenticity breaks,
  through the *reveal of plaintexts* and *modification of message history*
  only when two-party channels are compromised.
- *Limited* post-compromise security, against even *occasionally active* adversaries.
  But (!) it is unclear if *any* deployed messaging application achieves PCS in practice.
- WhatsApp provides a compelling alternative: device revocation.
  Device management provides strong revocation guarantees,
  giving *users* control over compromise recovery.
- Server-controlled group membership and lack of participant consistency,
  fail to fulfil our expectations for a *group* messaging protocol.

② Modelling

All of the features, limitations and attacks discussed in this talk are captured in our modelling and security analysis,

[1] A variant of the DOGM model introduced by the same authors in [SP:AlbDowJon24].

② Modelling

All of the features, limitations and attacks discussed in this talk
are captured in our modelling and security analysis,
i.e. within the Device Oriented Group Messaging with Revocation model we *introduce* in this work.[1]

[1] A variant of the DOGM model introduced by the same authors in [SP:AIbDowJon24].

② Modelling

All of the features, limitations and attacks discussed in this talk
are captured in our modelling and security analysis,
i.e. within the Device Oriented Group Messaging with Revocation model we *introduce* in this work.[1]

To see how our model and analysis captures these properties,
take a look at our DOGM with Revocation formalism,
and security analysis of WhatsApp in *Section 7* of our paper.

[1] A variant of the DOGM model introduced by the same authors in [SP:AlbDowJon24].

② Modelling

All of the features, limitations and attacks discussed in this talk
are captured in our modelling and security analysis,
i.e. within the Device Oriented Group Messaging with Revocation model we *introduce* in this work.[1]

To see how our model and analysis captures these properties,
take a look at our DOGM with Revocation formalism,
and security analysis of WhatsApp in *Section 7* of our paper.

+ Consider our Public Key Orbits formalism next time you need to analyze *device management*.

[1] A variant of the DOGM model introduced by the same authors in [SP:AlbDowJon24].

① Description

All of the features, limitations and attacks discussed in this talk are captured in our description,

① Description

All of the features, limitations and attacks discussed in this talk
are captured in our description, developed through our reverse-engineering effort.

① Description

All of the features, limitations and attacks discussed in this talk
are captured in our description, developed through our reverse-engineering effort.

For a more complete description of how group messaging in WhatsApp works,
take a look at our description in *Section 3* of our paper.

Interested?

ia.cr/2025/794