More is Less: How Group Chats Weaken the Security of Instant Messengers Signal, WhatsApp, and Threema

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

July 28, 2017

Abstract

Secure Instant Messaging (SIM) 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 SIM group communication.

In this paper, we investigate group communication security mechanisms of three main SIM applications: Signal, WhatsApp, and Threema. We first provide a comprehensive and realistic attacker model for analyzing group SIM protocols regarding security and reliability. We then describe and analyze the group protocols used in Signal, WhatsApp, and Threema. By applying our model, we reveal multiple weaknesses, and propose generic countermeasures to enhance the protocols regarding the required security and reliability goals. Our systematic analysis reveals that (1) the *communications' integrity* – represented by the integrity of all exchanged messages – and (2) the *groups' closeness* – represented by the members' ability of managing the group – are not end-to-end protected.

We additionally show that strong security properties, such as Future Secrecy which is a core part of the one-to-one communication in the Signal protocol, do not hold for its group communication.

1 Introduction

Short Message Service (SMS) has dominated the text-based communication on mobile phones for years. Instant Messaging (IM) applications started by providing free-of-charge SMS functionality, but today provide numerous additional features, and therefore are more and more displacing SMS [17, 31].

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, users have an increased awareness for security and privacy. Therefore servers delivering messages can no longer be seen as trusted third parties, and attacker models, which in the past focused on network-based attackers, now also include malicious server-based attacks [35, 40].

In contrast to open standardized communication protocols like Extensible Messaging and Presence Protocol (XMPP) and Internet Relay Chat (IRC), most IM protocols are centralized such that users of each application can only communicate among one another. As a result, a user cannot choose the most trustworthy provider but needs to fully trust the one provider that develops both, protocol and application.

End-to-end encryption is a major security feature of all Secure Instant Messaging (SIM) protocols, and additional security properties like *future secrecy* have been claimed [49], analyzed [33] and proven [23]. End-to-end encryption is part of all major IM apps, including Signal [53], WhatsApp [70], Threema [65], Google Allo [52], and Facebook Messenger [51]. One of the main achievements of SIM protocols is the *usability* of its end-to-end encryption. After the application installation, keys are automatically generated, and encryption is enabled — without any user-interaction. Experienced users may do some simple checks to verify the public key of their counterpart [27], but this is often an optional step.

Contrary to classical multi-user chats, for example, to IRC in which all members are online, groups in IM protocols must work in asynchronous settings; Groups must be createable and messages must be deliverable even if some group members are offline. When it comes to end-to-end encryption, this leads to the complex problem of asynchronously agreeing on a group key. Instead of establishing a group key between all group members and then encrypting messages once with that dedicated key, many modern SIM applications encrypt messages with keys established between the initiator (sender) and the receiver similar to direct messaging. The message is enriched with meta information, for example, a flag indicating that the message is a group message (instead of a direct message) and the unique identifier of the group, encrypted with each member's key, and delivered to these members. On the one hand, this mechanism leads to an increased message overhead depending on the total number of group members. On the other hand, security properties (e.g., key renewal, delivery status information, ...) can be gained without any further effort.

The fact that widely used SIM protocols are neither open source nor standardized makes it harder to analyze and compare their security properties. On the one hand, the applications must be reverse engineered [9, 33, 34] for retrieving a protocol description. On the other hand, third-party implementations are often blocked by providers [69] such that an active analysis is even more complicated.

When analyzing the protocols, the security properties in the setting of *asynchronous, centralized* messaging must be investigated with the whole group environment in mind. The security of a protocol does not only rely on single messages, exchanged between two group users. For example, the abstract security goal *confidentiality* is based on the composition of the strength of the encryption algorithm for protecting the content of single messages and the protocol's strength to ensure that users who do not belong to a group must not be able to add themselves to the group or send messages within the group without the members' permission. Additionally, the *integrity* of the communication is not restricted to the non-malleability of single exchanged messages but also consists of the correct message delivery between the communicating users.

Established definitions like *reliable multicast* [15, 36] and related formalizations like group communication systems (GCS) [22] provide a set of properties that need to be reached for achieving a secure and reliable group communication. However, they do not fully match the described setting and over-accomplish the reliability requirements. Therefore the modeling of our security and reliability definitions bases on the three representative SIM applications by extracting security properties from their features (provider statements or visual user interface). We matched these requirements to definitions from the mentioned and further related fields of research (e.g., authenticated key exchange, reliable broadcast, GCS) and thereby provide a novel comprehensive attacker model for the investigation of group SIM protocols.

Accordingly we investigate these three popular SIM applications: Signal [53], Threema [65], and WhatsApp [70]. Signal can be seen as a reference implementation for other SIM protocols that implement the Signal key exchange protocol like Facebook Messenger, Google Allo and other messengers. However, our analysis shows that the integration of the Signal key exchange protocol does not imply same group communication protocols. We chose to analyze WhatsApp, because it is one of the most widely used SIM applications with more than one billion users [59]. We additionally chose to analyze Threema as a widely used representative for the class of proprietary and closed source SIM applications. Signal and Threema are both used by at least one million Android users [58, 64]. Based on this examination, we apply our model and evaluate the security properties. In our systematical analysis, we reveal several design weaknesses in the group communication protocols of these applications. Our contributions are outlined as follows:

- ► We present a realistic and comprehensive attacker model for the analysis of group communications in SIM protocols (§ 2).

- ► We describe the group communication protocols of Signal (§4), WhatsApp (§5), and Threema (§6) and thereby present three fundamentally different implementations of SIM group communication protocols. We analyze them by applying our model and thereby reveal several security issues that break the traceable delivery, closeness and thereby confidentiality of their group chat implementations. As a result, we show that none of these group communication protocols achieves Future Secrecy.
- ► We provide and compare generic approaches to secure group communications, based on our observations and related literature (§7).

All findings have been responsibly disclosed to the application developers.

2 Security Model for Encrypted Group Chats

Secure Instant Messaging (SIM) protocols should satisfy the general security goals *confidentiality*, *integrity*, *authenticity* and *availability*. Some of them even claim advanced security goals like *future secrecy*.

One could expect that a SIM group protocol should provide the same properties, as well as several others that are naturally achived in a two-party scenario. Intuitively, a group protocol in SIM should achive all security goals that are gained when a group of people is communicating in an isolated room: everyone in the room hears the communication (*traceable delivery*), everyone knows who spoke (*authenticity*), nobody else can replay the words because the speaker's lips can be seen (*no duplication*), nobody outside the room can either speak into the room (*no creation*) or hear the communication inside (*confidentiality*), and the door to the room is only opened for invited persons (*closeness*).

2.1 Notation and Assumptions

Most modern SIM protocols are *centralized*: all exchanged messages are transmitted via a central server, which receives messages from the respective senders, caches them and forwards them as soon as the receivers are online. Hence the protocols are executed in an *asynchronous* environment in which only the server is always online.

We generally define a group gr as the tuple

$$(ID_{gr}, \mathcal{G}_{gr}, \mathcal{G}^*_{gr}, info_{gr}), \mathcal{G}^*_{gr} \subseteq \mathcal{G}_{gr} \subseteq \mathcal{U}$$

where $\mathcal{U}$ is the set of protocol users, $\mathcal{G}_{gr}$ is the set of members in the group and $\mathcal{G}^*_{gr}$ is the set of administrators of the group. The group is uniquely referenced by ID_{gr}. Additionally, a title and other usability information can be configured in $info_{gr}$.

We denote communicating users as $A, B, C, .., U, .., X \in \mathcal{U}$.

It is important to distinguish between actions that can be conducted by all members in $\mathcal{G}_{gr}$ and actions that are only permitted to the administrators in $\mathcal{G}^*_{gr}$. All members can *send* messages and *leave* a group. *Adding* members to the group should only be possible for administrators.

2.2 Threat Model

We consider five types of adversaries against SIM protocols:

Malicious User. Since all protocols are open for new users, the adversary may act as a malicious user who can arbitrarily deviate from the protocol specification. To exclude trivial attacks against the instant delivery of messages, we assume that members of the target group behave *correctly* by always following the protocol description.

Long-term Secret Compromise. This adversary extends the *malicious user* by being able to compromise a particular user during or after the protocol execution, to obtain her long-term secrets.

Session State Compromise. This adversary extends the *malicious user* by being able to compromise a user to obtain the full session state at some intermediate stage of the protocol execution.

Network Attacker. This adversary has full control over the communication network, and may access and modify all unprotected traffic.

Advanced Network Attacker. This adversary models attackers with access to the group chat protocol alone. This type of attacker is non-standard, but is motivated by our aim to analyze the reliance of the SIM protocols on the transport layer protection. One possibility is an attacker who can break transport security protocols employed between the users and the central SIM server. A practical justification of this attacker are recurrent successful attacks on TLS [3, 5, 46]. Similarly to other models [35, 40], this adversary also models an attacker impersonating the central SIM server or a forgery of its certificates.

2.3 Security Goals

Before describing security goals for SIM group protocols, we look into security goals for direct messaging in SIM. We will then extend their meaning for SIM group communication.

Security Goal: Confidentiality. In two-party communications, standard and advanced confidentiality goals are:

- ▶ *End-to-end Confidentiality* Only the two participants of a direct messaging communication can see the message plaintext.
- ▶ *Perfect Forward Secrecy and Future Secrecy* On leakage of secret values, neither past (Perfect Forward Secrecy) nor future (Future Secrecy) messages' confidentiality may be compromised. The attacker can only break it for a relatively short protocol execution frame.

Future Secrecy is also known as *Post-Compromise Security* [23]. Our further examination shows that compromise related security does not necessarily restricts to confidentiality. Therefore *Future Secrecy* can also be applied to the other security goals.

Security Goal: Integrity. This goal not only targets end-to-end integrity of single messages, but the whole communicated content.

- ▶ *Message Authentication* If a message is received and successfully validated, then it was indeed sent by the given sender.
- ▶ *Traceable Delivery* If a message is sent, then this message is received by all its intended receivers or the sender is informed about the failed delivery.
- ▶ *No Duplication* No message can be replayed to a recipient.

Both abstract security goals are applied to SIM group protocols by regarding the set of recipients accordingly.

Additional Security Goals in Groups. Group protocols must fulfill additional requirements to meet the abstract goals of *confidentiality* and *integrity.*

- ▶ *No Creation* Only group members can send messages to a group. More formally: if a member $B \in \mathcal{G}_{gr}$ receives message m from sender A, then $A \in \mathcal{G}_{gr}$ holds.
- ▶ *Closeness* Only an administrator may alter the set of users in a group. More formally: if a member $A \in \mathcal{G}_{gr}$ transforms the member set $\mathcal{G}_{gr}$ to $\mathcal{G}_{gr}^{new} : |\mathcal{G}_{gr}^{new}| > |\mathcal{G}_{gr}|$, then an administrator $U^* \in \mathcal{G}_{gr}^*$ added at least one new member $B \notin \mathcal{G}_{gr}$ to the member set $\mathcal{G}_{gr}^{new}$.

Some of these security goals are not stated explicitly in the SIM-companies' security whitepapers, but are implied by the GUI presented to the users. For example, the practical implementation of Traceable Delivery can be seen in Figure 1: the first checkmark is set when the message is delivered to the SIM server, and the second checkmark is only set *if the message was received by all group members.*

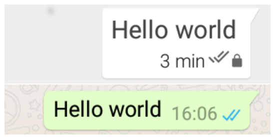

Hello world
3 min
Hello world 16:06

Figure 1: Double checkmarks in Signal (upper screenshot) and WhatsApp (lower screenshot) indicating that a group massage was successfully delivered to *all* members' devices.

3 Methodology

We describe our general evaluation methodology in the following.

Test Setup. For all three SIM applications, we used the official Android versions provided by the Google Play Store. In order to analyze groups, we created a group of at least three members using three different devices.

Protocol Descriptions. We derived the protocol descriptions by analyzing the source code and debugging the implementations. For Signal, we used source code available on Github [61, 62]. Since neither WhatsApp nor Threema provide official open source implementations, our analysis of these protocols mainly bases on the traffic that was received by unofficial protocol implementations [9, 34]. The respective messages and operations were sent by the official applications running on different devices and transmitted via the official SIM servers.

Vulnerability Proof-of-Concepts. In order to substantiate the described protocol weaknesses, we were able to implement proof-of-concept exploitations for a subset of them. We therefore modified the source code of the according application in order to execute the attack. Please note that we could only manipulate the source code of the SIM clients. We did not have any access to the SIM servers. Details are given in Sections 4.3, 5.3, and 6.3.

Protocol Observations. Some attacks could not be practically exploited. The attacks on WhatsApp use the advanced network attacker. Since we could not manipulate the incoming traffic to the official WhatsApp application due to transport layer protection, our results for these attacks base on the analysis of the observed protocols. If we could not practically exploit the vulnerability, we note them as *observation*.

Responsible Disclosure. All tested and untested weaknesses were acknowledged by the developers during the responsible disclosure process. Threema has already updated its application.

Description of an Exemplary Protocol Run. In order to provide a comparable description of the protocols, Figures 3 (Signal), 5 (WhatsApp) and 7 (Threema) depict an exemplary protocol run of each protocol containing direct and group communication. The figures are meant to highlight the differences in the three SIM group protocols. The depicted protocol sequence covers the key usage for the following actions:

(1) User A sends a direct message $m =$ `"Hi"` to user B.

(2) User A sends a group message $m =$ `"Hey"` to a group with members $G = \{A, B, C\}$.

(3) User A receives the information that user B leaves the group with members $G = \{A, B, C\}$, such that its members are $G = \{A, C\}$ afterwards.

(4) User A sends a group message $m =$ `"Ho"` to the group with members $G = \{A, C\}$.

(5) User A creates a group with members $H = \{A, B, C\}$.

(6) User A sends a group message $m =$ `"Yo"` to the group with members $H = \{A, B, C\}$.

(7) User A receives a group message $m =$ `"Yey"` from user C to the group with members $H = \{A, B, C\}$.

4 Signal

Signal is an open source SIM application available for Android, iOS and as a Google Chrome extension [63]. It is well-known for its key exchange that reaches the goals *Perfect Forward Secrecy* and *Future Secrecy*. Previous analyses focused on the key exchange protocol and direct messaging between two participants [23, 33].

Signal provides group messaging of text messages and other content such as pictures or videos. We restrict our investigation to group messaging including the transmission of text content. Our analysis is based on the source code of the Android application [61] and the Java library [62].

In Signal, a user is allowed to run multiple devices simultaneously, for instance, one mobile app (iOS or Android) plus an arbitrary number of Google Chrome extensions. Thereby sending and receiving of messages from all connected devices is possible and the chats (groups, and direct messages) are synchronized among them. Our analysis does not consider this feature and assumes multiple users with one device each to form groups because this strengthens the comparability of the analyzed protocols.

The Signal application implements *Curve25519* [11] and *HMAC-SHA256* [6] for the key derivation (*Double Ratchet algorithm*). The *HMAC* is also used for message authenticity in combination with *AES-CBC-PKCS5Padding* [26] for preserving confidentiality of the messages. We assume these implementations secure and did not look for implementation issues therein.

In the following sections, we shortly introduce the general protocol setting stripped down to the essence necessary to understand the group communication. We then describe the group protocol and evaluate it regarding the defined attacker model.

Figures 3, 5, and 7 depict an exemplary protocol run of the analyzed protocols and thereby give an overview on the fundamental differences in Signal, WhatsApp, and Threema.

4.1 General Initialization Protocol

4.1.1 Session Establishment with the Server

For identification and authentication, each user (more precisely, each device) holds credentials. This is a user name, which corresponds to the user's phone number, and a password that is randomly chosen by the Signal server during the device's initial usage. The credentials are sent to the Signal server in every request. Additionally, Signal uses Transport Layer Security (TLS) as a cryptographic primitive to protect the channel between users and the server.

4.1.2 Key Agreement and Key Derivation

The initial shared secret (root key) between two parties is calculated with the *X3DH Key Agreement Protocol* [45] that uses static and ephemeral Diffie-Hellman shares of both parties. This root key initializes the *Double Ratchet algorithm (DR algorithm)* [44], which can be seen as a stateful encryption algorithm [7]. The algorithm's state – consisting of multiple keys – is updated asymmetrically by both parties during the communication and symmetrically as long as only one communication party uses the algorithm. This key update process is called ratcheting. When only the symmetric updating is conducted – as in WhatsApp groups – this is called *symmetric ratcheting*. The DR algorithm is consequently the combination of symmetric and asymmetric ratcheting. Thereby the initialization keys of the symmetric ratcheting are called *chain keys*. Due to the characteristics of the symmetric ratcheting, it cannot provide *Future Secrecy* but provides *Perfect Forward Secrecy* of the resulting keys. The asymmetric ratcheting provides both properties.

The encryption DRE and decryption DRD of the DR algorithm have modifying access to the keys which are stored in the state (denoted as A, B in Figures 2 and 4). The key for encrypting and decrypting is generated as soon as it is needed and removed directly afterwards. Only intermediate keys (e.g., chain keys) that are not used for encryption and decryption are stored in the state.

$$c \overset{\$}{\leftarrow} \mathrm{DRE}_{A,B}(m), \qquad\qquad m := \mathrm{DRD}_{A,B}(c)$$

A schematic description of the DR algorithm when used in the Signal and WhatsApp messaging protocol can be seen in Figure 8. The usage of the key streams can be seen in the exemplary protocol run in Figures 3 and 5.

4.2 Group Protocol

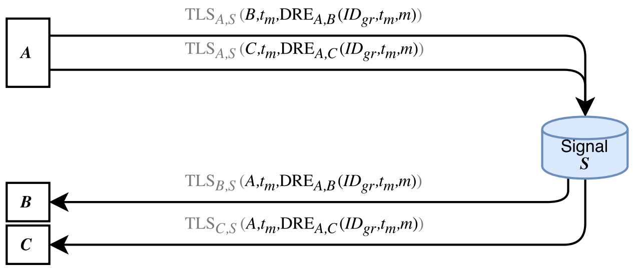

TLS_{A,S} (B,t_m,DRE_{A,B}(ID_{gr},t_m,m))
TLS_{A,S} (C,t_m,DRE_{A,C}(ID_{gr},t_m,m))
A
Signal
S
TLS_{B,S} (A,t_m,DRE_{A,B}(ID_{gr},t_m,m))
TLS_{C,S} (A,t_m,DRE_{A,C}(ID_{gr},t_m,m))
B
C

Figure 2: Schematic depiction of Signal's traffic, generated for a message m from sender A to receivers B and C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$. Transport layer protection is not in the analysis scope (gray).

In contrast to other SIM group protocols (e.g., WhatsApp and Threema), Signal implements non-administered groups such that all members of a group can manipulate the group management information (i.e. $\mathcal{G}^*_{gr} = \mathcal{G}_{gr}$). The group is uniquely referenced by a random 128 bit vector ID_{gr}.

4.2.1 Group Messages

A group message in Signal is treated as a direct message but the group ID is additionally attached to the encrypted plaintext. By using this approach, the Signal server cannot distinguish a group message from a direct message. Together with the timestamp t_m, the message is statefully end-to-end encrypted for each member of the group. Every resulting ciphertext is then sent to the server together with the respective receiver ID and the timestamp via TLS. The server forwards the end-to-end encrypted messages to the respective group members via TLS, as well. When the server forwards the message to the receivers, it replaces the receiver's ID by the sender ID.

Figure 2 describes the format of a group message from member A to members B and C in group $\mathcal{G}_{gr}$ that is sent via the server $\mathcal{S}$[1].

Messages for group management contain the updated group information in the end-to-end encrypted message part:

$$m := \begin{cases} plain\ content, & if \text{ group content message} \\ (\mathcal{G}_{gr}, info_{gr}), & if \text{ group update message} \\ \texttt{leave}, & if \text{ group leave message} \end{cases}$$

The server acknowledges messages from the sender, and the receivers acknowledge the receipt to the server. These acknowledgments contain the sender ID and the timestamp t_m of the original message but not the group ID. Once a receiver's acknowledgment is gained, the server forwards this receipt acknowledgment to the sender. All acknowledgments are not end-to-end encrypted, thus only rely on TLS. The sender collects the members' acknowledgments and displays a successful receipt (see checkmarks in Figure 1) as soon as all receivers' acknowledgments arrived.

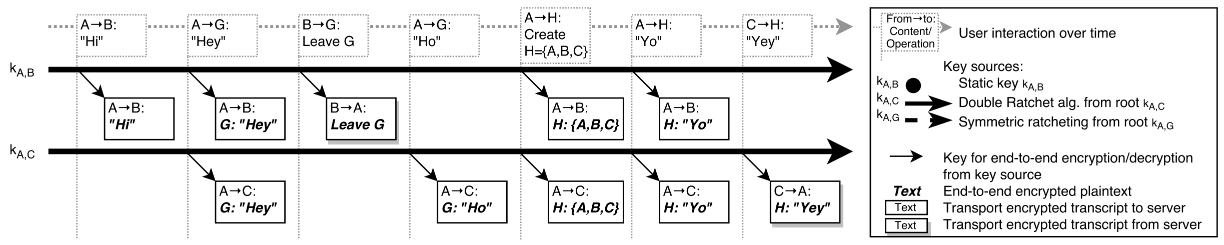
[1]We omitted irrelevant fields regarding our evaluation in the message format. The whole format can be found in the format description [60]. We also left out Google's Cloud Messaging (GCM) service for clarity.

4.2.2 Group Management

The group management consists of two protocol flows: an *update* flow and a flow that is processed once a user *leaves* the group.

The update flow is used for the creation of a group, for adding users and for changing group information like the title of a group. For creating and updating a group, the modifying member sends an end-to-end encrypted message to each group member, containing the new set of members $\mathcal{G}_{gr}$ and the new group information $info_{gr}$. Signal does not allow removing of other members from a group. As a result an update message, containing not the complete member set $\mathcal{G}_{gr}$, does not lead to the removal of missing group members.

If a member choses to leave the group, she sends a *leave* information together with ID_{gr} end-to-end encrypted to every other member.



A→B: "Hi"
A→G: "Hey"
B→G: Leave G
A→G: "Ho"
A→H: Create H={A,B,C}
A→H: "Yo"
C→H: "Yey"
k_{A,B}
k_{A,C}
A→B: "Hi"
A→B: G: "Hey"
B→A: Leave G
A→B: H: {A,B,C}
A→B: H: "Yo"
A→C: G: "Hey"
A→C: G: "Ho"
A→C: H: {A,B,C}
A→C: H: "Yo"
C→A: H: "Yey"
From→to: Content/ Operation
User interaction over time
Key sources:
Static key k_{A,B}
Double Ratchet alg. from root k_{A,C}
Symmetric ratcheting from root k_{A,G}
Key for end-to-end encryption/decryption from key source
Text End-to-end encrypted plaintext
Text Transport encrypted transcript to server
Text Transport encrypted transcript from server

Figure 3: Schematic depiction of key streams of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in Signal. The legend of the graphic also regards to Figures 5 and 7.

4.2.3 Exemplary Protocol Run

Figure 3 depicts an exemplary protocol run. We denote the key derivation function (ratcheting) by an arrow, which forks multiple keys used for encryption and decryption (strongly simplified). The only difference between group messages and direct messages can be found inside the end-to-end encrypted plaintext. Summarized, one group message results in multiple direct messages. The group management messages are also communicated via multiple direct messages.

As Figure 3 shows, A maintains two separate key streams, one for the communication with B and the one for C. Both are separately used for direct and group communication (with B resp. C).

4.3 Security Evaluation and Observations

We practically carried out two attacks on Signal and created proof-of-concepts for them. First, we burgle into a group by writing group management messages into it (breaks: Closeness, No Creation, and Future Secrecy). Second, we make a victim believe that a message is delivered while it is not (breaks Traceable Delivery).

4.3.1 Burgle into the Group

This vulnerability allows an attacker to become a member of the targeted group. The attacker can read any further group communication and contribute own content to the group chat. Because every group member in Signal has administrative privileges, the attacker automatically becomes a group administrator.

Preconditions. The attacker only needs to know the group ID ID_{gr} and the phone number B of one member.

► *Malicious User* In the simplest case, the attacker was a former member of the group, and has recorded the group ID using a modified client software.

- *Long-term secret compromise* This is our general adversarial model for Perfect Forward Secrecy and Future Secrecy. We regard the ID_{gr} as a long-term secret, and the phone number B as public information.
- *Session state compromise* Since ID_{gr} is part of every group message, it suffices to obtain a message key and the ciphertext of a group message in order to retrieve the group ID.

Attack description. The attacker $\mathcal{A}$, knowing the secret group ID ID_{gr}, sends the following *group update* $m = (\{\mathcal{A}\}, info_{gr}))$ to the known phone number B, using Signal's direct messaging channel between $\mathcal{A}$ and B:

$$(B, t, \mathrm{DRE}_{\mathcal{A},B}(ID_{gr}, t, (\{\mathcal{A}\}, info_{gr})).$$

In fact, $\mathcal{A}$ could also send a *content message* such that only this message is sent to B in the group without adding $\mathcal{A}$ to the group. This message breaks the *no creation* security goal. After receiving and validating this message, B's receiving Signal application updates its own group description:

$$\mathcal{G}_{gr}^{new} := \mathcal{G}_{gr} \cup \{\mathcal{A}\}.$$

B will use this set $\mathcal{G}_{gr}^{new}$ in all future communications with the group. However until now, $\mathcal{A}$ will only receive group messages from B, but not from the other members.

This changes once group member B sends a second update message to the group. For example, if B changes the group icon (which is part of $info_{gr}$), she will send some message

$$(U, t', \mathrm{DRE}_{B,U}(ID_{gr}, t', (\mathcal{G}_{gr}^{new}, info'_{gr}))$$

to all members $U \in \mathcal{G}_{gr}^{new}$. After receiving this message, each member U will update her group member set to $\mathcal{G}_{gr}^{new}$. From now on, $\mathcal{A}$ receives *all* group messages.

To all other group members except B, it seems that B has added $\mathcal{A}$ to the group, which would be fine since B was a member and thereby an administrator of the group.

Attack Optimizations. If $\mathcal{A}$ knows the phone number of multiple members, $\mathcal{A}$ can send this group update message (or a content message) to all of them. Thereby *no creation* and *closeness* is broken for a larger set of members, and it is more likely that one of these members sends the second update message.

Attack Impact. The attack violates the following security goals:

- *No Creation* A group member B accepts a message by $\mathcal{A}$, who is not part of the group. This can be either a *content* or an *update* message.
- *Closeness* By using an *update* message, breaking No Creation breaks also Closeness, since $\mathcal{A}$ can add herself to the group.
- *Future Secrecy* After adding herself to the group, the confidentiality of future plaintext messages is compromised.

4.3.2 Breaking Traceable Delivery

Signal provides information on the receipt status of messages for the sender in groups and for direct messaging (see Figure 1). However, this information can be forged by the Signal server.

Even though the Signal protocol internally provides two features to detect that sent messages were not received by the desired recipient, the detection is not effective. Hence messages can stealthily be dropped during the transmission.

Preconditions.

▶ *Advanced Network Attacker* The attacker $\mathcal{A}$ must be able to directly deliver a message to the victim's Signal application. Therefore, $\mathcal{A}$ must either compromise the Signal server, or be able to bypass the transport layer protection.

Attack Description. As soon as a sender B sends a group message

$$(U, t_m, \mathrm{DRE}_{B,U}(ID_{gr}, t_m, m))$$

to all members $U \in \mathcal{G}_{gr}$, the attacker $\mathcal{A}$ drops the message, for instance, she does not forward it to member X. She then sends multiple acknowledgment response messages to B:

$$(U, t_m, \texttt{ACK}), \forall U \in \mathcal{G}_{gr} \setminus \{B\}$$

B's application displays the successful delivery even though member X never saw message m.

Attack Impact. The attack violates the following security goal:

▶ *Traceable Delivery* The receivers, for whom the message was dropped, never see B's message. As a consequence, B's device indicates a successful message delivery (see Figure 1) while members did not receive the message.

Despite the fact that the *DR algorithm* provides a continuous key stream, omissions of keys are ignored at the receiver's side and thereby the statefulness of the key stream is not used. Since receiver acknowledgments in Signal are not end-to-end encrypted, $\mathcal{A}$ can drop messages and create the acknowledgments itself. Dropping messages is however slightly restricted: the client application only maintains the last 2000 keys such that a further deviation of the sender's and receiver's key streams causes the encryption to fail [2].

As a result *traceable delivery* is neither provided for group messages nor for direct messages by Signal.

4.3.3 Further Weaknesses

Vulnerabilities regarding additional security properties are described in subsection A.2.

5 WhatsApp

WhatsApp is a closed source instant messaging protocol. It uses the Signal protocol for key exchange and encryption but is independent of Signal's messaging protocol – especially, it is independent of the Signal group communication protocol. WhatsApp is available for most mobile operation systems[3].

Even though WhatsApp is a closed source application, there exist open source implementations [34, 47] whose usage is forbidden and aimed to be prevented by WhatsApp [69]. We used a fork[4] of Galal's implementation [34] to analyze the traffic, generated by the official WhatsApp Android application[5].

The algorithms for exchanging the keys and encrypting on the end-to-end layer use the same cryptographic primitives as the implementation of Signal relies on. The signatures of group messages are calculated on *Curve25519* [11].

Our analysis confirms the description of WhatsApp's technical white paper [38] regarding the implementation of the Signal key exchange protocol but further examines the messaging protocol as a whole. As a result, we present several protocol and implementation weaknesses.

[2]`https://github.com/WhisperSystems/libsignal-protocol-java/blob/master/java/src/main/java/org/whispersystems/libsignal/state/SessionState.java#L41`

[3]`https://www.whatsapp.com/download/`

[4]`https://github.com/colonyhq/yowsup`

[5]Version 2.17.107 from the Google Play Store

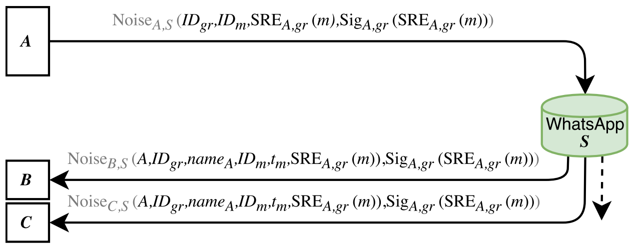

$\text{Noise}_{A,S}(ID_{gr},ID_m,\text{SRE}_{A,gr}(m),\text{Sig}_{A,gr}(\text{SRE}_{A,gr}(m)))$
A
WhatsApp S
$\text{Noise}_{B,S}(A,ID_{gr},name_A,ID_m,t_m,\text{SRE}_{A,gr}(m)),\text{Sig}_{A,gr}(\text{SRE}_{A,gr}(m)))$
B
$\text{Noise}_{C,S}(A,ID_{gr},name_A,ID_m,t_m,\text{SRE}_{A,gr}(m)),\text{Sig}_{A,gr}(\text{SRE}_{A,gr}(m)))$
C

Figure 4: Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in WhatsApp.

5.1 General Initialization Protocol

5.1.1 Session Establishment with the Server

WhatsApp uses *Noise Pipes* [54] to protect the communication between the clients and the server on the transport layer [38]. The *Noise Pipes* are implemented with *Curve25519*, *AES-GCM*, and *SHA256*.

5.1.2 Key Agreement and Key Derivation

The Signal key exchange protocol, consisting of the *X3DH Key Agreement Protocol* [45] and the *DR algorithm* [44], is integrated in WhatsApp in order to establish a confidential channel for messaging between two users [38]. A detailed description of these building blocks can be found in section 4 and Figure 8.

5.2 Group Protocol

WhatsApp limits the maximum number of users in a group to 256. A group is uniquely referenced by ID_{gr}, containing the creator's user ID and a timestamp. The initial set of administrators $\mathcal{G}^*_{gr}$ contains the group creator. By adding members to the administrator set, this set can be enlarged. The content of messages is protected on the end-to-end layer while group modification messages are only protected on the transport layer. As a result, the WhatsApp server is mainly responsible for the distribution of group messages based on the group management. This is a main difference in comparison to Signal and Threema.

Although WhatsApp integrates the Signal key exchange protocol for direct messaging, keys in groups are used very differently: instead of sending encrypted messages to each group member separately (cf. section 4), each user generates a symmetric key (*chain key*) for encrypting only her messages *to* the group. The key is then once transported to every other group member using the DR algorithm for direct messaging. The dedicated group key is not *refreshed* by Diffie-Hellman ratcheting but only with the symmetric key derivation function in contrast to direct messaging.

5.2.1 Group Content Messages

All messages between the users and the server are transport layer encrypted. On the end-to-end layer only the actual content is encrypted and integrity protected under the *symmetric ratcheted encryption* SRE (see subsubsection 4.1.2) with a message key from the symmetric ratcheting of the sender's chain key. As a result, the sender calculates one ciphertext for the whole group. This ciphertext is then signed with the current signature key for the respective group (denoted as Sig in Figure 4). The receiving members can compute the symmetric key for the decryption from the sender's chain key, that was sent with her first message after a group management operation (see below). Apart from the ciphertext, the transcript to the server also contains ID_{gr} and a message identifier ID_m. The server adds the sender ID, a readable sender name and a timestamp t_m to the message for the receivers.

Notifications on the receipt status for the sender and an acknowledgment for the WhatsApp server are sent protected by the transport layer only. The server forwards the receipt statuses to the sender. As soon as all members' receipts are collected by the sender, the successful delivery is displayed by the double checkmark (see Figure 1). Additionally, the individual receipt statuses are listed in an extended menu.

As a result, group messages only result in one ciphertext to the server independent of the group size.

5.2.2 Group Management

Group administrators send group modifications to the server. These modification messages are only encrypted on the transport layer and no cryptography is used to protect them on the end-to-end layer between a group's members.

The modification messages contain the tuple $OP = (action, \mathcal{H})$ where *action* indicates the operation type like adding or removing of members, adding of administrators, leaving of members and $\mathcal{H}$ is the set of affected users. After an administrator sent such a message to the server, the information is distributed to all group members:

$$(A, ID_{gr}, name_A, ID_{OP}, t_{OP}, OP)$$

The group secret of each member consist of the chain key and a signature key pair. Both are generated freshly for the first message to a new group or for the first message to the group after a user left or was removed from it as it can be seen in Figures 5 and 8. After the generation, the public signature key and the chain key are distributed to all members via direct messaging between the sender and the respective receiver using the DR algorithm. Consequently, the first message after which the group secrets are updated results in $|\mathcal{G}_{gr}|$ ciphertexts. When a user is added to the group, the current chain key and the signature key of each member is sent along with the first message after adding the new user the same way.

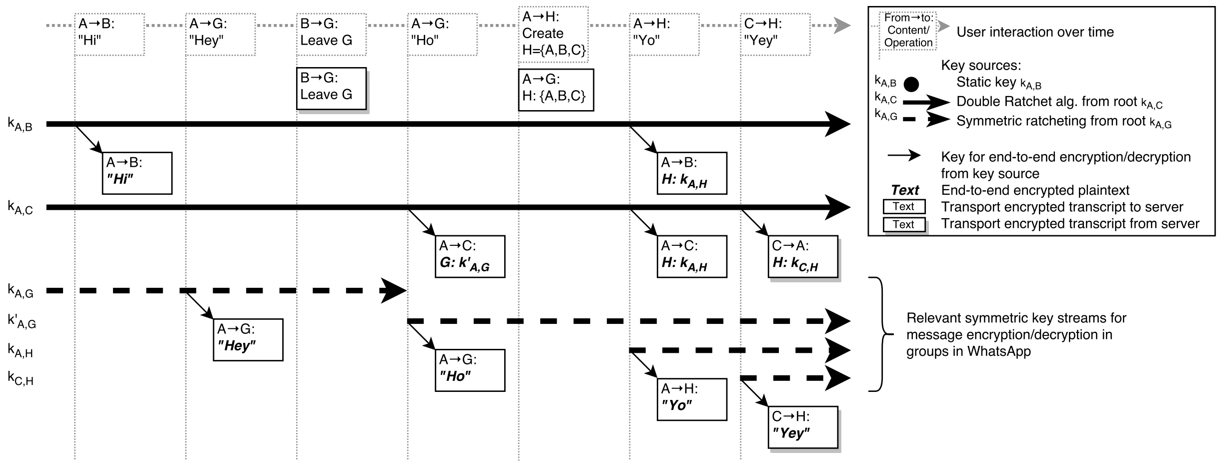

A→B: "Hi"
A→G: "Hey"
B→G: Leave G
A→G: "Ho"
A→H: Create H={A,B,C}
A→H: "Yo"
C→H: "Yey"
B→G: Leave G
A→G: H: {A,B,C}
$k_{A,B}$
$k_{A,C}$
$k_{A,G}$
$k'_{A,G}$
$k_{A,H}$
$k_{C,H}$
A→B: "Hi"
A→B: H: $k_{A,H}$
A→C: G: $k'_{A,G}$
A→C: H: $k_{A,H}$
C→A: H: $k_{C,H}$
A→G: "Hey"
A→G: "Ho"
A→H: "Yo"
C→H: "Yey"
From→to: Content/ Operation
User interaction over time
Key sources:
Static key $k_{A,B}$
Double Ratchet alg. from root $k_{A,C}$
Symmetric ratcheting from root $k_{A,G}$
Key for end-to-end encryption/decryption from key source
End-to-end encrypted plaintext
Transport encrypted transcript to server
Transport encrypted transcript from server
Relevant symmetric key streams for message encryption/decryption in groups in WhatsApp

Figure 5: Schematic depiction of key streams of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in WhatsApp.

5.2.3 Exemplary Protocol Run

Figure 5 depicts an exemplary protocol run. In contrast to Signal, WhatsApp maintains different key streams for direct messaging and for group messaging. Keys for the group communication are generated once they are used and distributed via the direct communication channels. If a group is created or a user is removed from a group, each member generates a new group key. Every member needs to store one key for every direct communication and one key for every member in each group. The information on group modifications is not end-to-end encrypted.

5.3 Security Evaluation and Observations

We observed two design weaknesses in WhatsApp's group protocol that allow to (1) burgle into a group and to (2) break Traceable Delivery. The weaknesses have a similar results to the attacks on Signal, although the underlying protocol and the weakness exploitation differs.

5.3.1 Burgle into a Group

The following protocol vulnerability allows an attacker $\mathcal{A}$, controlling some of the messages sent by the WhatsApp server, to become a member of the group or add other users to the group without any interaction of the other users.

Preconditions. The attacker $\mathcal{A}$ needs to modify the group information at the client side.

- ▶ *Advanced Network Attacker* can send group modification messages to the group members on the communication channel between the WhatsApp server and the group member.

Attack Description. Suppose we have a group gr with three members B, C, D whereas B is the group administrator:

$$gr = (ID_{gr}, \mathcal{G}_{gr} = \{B, C, D\}, \mathcal{G}^*_{gr} = \{B\}, info_{gr})$$

The attacker $\mathcal{A}$ can then break Closeness in the group by conducting the following steps. The attacker sends the following group modification message to users C, D[6]:

$$(B, ID_{gr}, name_B, ID_m, t_m, (\texttt{add}, \{\mathcal{A}\}))$$

Each receiving member sets

$$\mathcal{G}^{new}_{gr} := \mathcal{G}_{gr} \cup \{\mathcal{A}\}$$

and sends her current chain key and signature public key to $\mathcal{A}$ as soon as she sends a message to the group.

Since the modification of the group information is not bound to a cryptographic operation, it is not necessary that a group member initiates the operation. The WhatsApp server can thereby forge a message that indicates an added member for a group.

Attack Optimizations. The attack can be optimized by adding $\mathcal{A}$ to B's group perspective. There are different approaches to achieve this: (a) if B's client accepts group modification messages with source B even though B did not originate the operation, the described message is also sent to B to update $\mathcal{G}^{new}_{gr} := \mathcal{G}_{gr} \cup \{\mathcal{A}\}$, (b) if B's client accepts this message from a non-administrative member, the message is sent to B with source C or D, (c) in bigger groups with two or more administrators, the attacker pretends the message to be originated from one administrator when sending it to another.

Attack Impact. The attack violates the following security goals:

- ▶ *Closeness* $\mathcal{A}$ can write to the group and read messages.

5.3.2 Breaking Traceable Delivery

Even though WhatsApp's graphical user interface implies that a sender sees the receipt status of sent messages (double checkmark), this weakness allows the attacker to stealthily drop messages.

Preconditions. The attacker needs to drop messages and send notifications to the sender.

- ▶ *Advanced Network Attacker* can manipulate the transcript between sender and server or server and receivers.

[6]Schematic representation of modification message for adding a new member to a group.

Attack Description. The attacker drops a group message from the sender and replies with acknowledgments, indicating the successful receipt for all members. These acknowledgments are of the form

$$(U, ID_{gr}, ID_m, t_m, \texttt{ack}),\ U \in \mathcal{G}_{gr} \setminus \{A\}$$

where A is the original sender of the message.

Attack Impact.

► *Traceable Delivery* WhatsApp's delivery state information is vulnerable towards the described attacker.

Although the key derivation from the chain key provides a consecutive key stream, the omission of message keys is ignored by the receivers to a certain degree. Our practical evaluation showed that 1999 omitted keys were ignored. Additionally the receiver's acknowledgments are not authenticity protected. Consequently Traceable Delivery is not provided because the attacker can drop sent messages and tamper the receiver's receipt status arbitrarily by sending forged receipt notifications to the sender.

Although our description covers the group setting, this weakness directly applies for direct messaging.

5.3.3 No Future Secrecy

Since Diffie-Hellman key ratcheting, as one main component of the *DR algorithm*, is not integrated into the encryption of group messages, Future Secrecy cannot be reached.

5.3.4 Further Weaknesses

As described in subsection A.3, *ordering* is not protected in WhatsApp.

5.4 Impact of the Weaknesses' Combination

The described weaknesses enable attacker $\mathcal{A}$, who controls the WhatsApp server or can break the transport layer security, to take full control over a group. Entering the group however leaves traces since this operation is listed in the graphical user interface. The WhatsApp server can therefore use the fact that it can stealthily reorder and drop messages in the group. Thereby it can cache sent messages to the group, read their content first and decide in which order they are delivered to the members. Additionally the WhatsApp server can forward these messages to the members individually such that a subtly chosen combination of messages can help it to cover the traces.

6 Threema

Threema is a proprietary closed source instant messenger protocol available for most mobile operation systems [65]. It uses a centralized server architecture for relaying messages to the respective receivers and distributing user keys. The messenger application provides direct messaging and group chats. In both settings not only text messages but also pictures, arbitrary files, contacts and other content can be sent.

Even though the application is closed source, there are open source implementations available: we based our analysis on the implementation of Berger [9] which was based on an analysis of Ahrens [2]. We used the open source implementation only for analyzing the protocol flow and for proof-of-concept exploitation. We then observed the results of an attack on a parallel running, original Android application from the Google Play Store, which simulates the victim.[7]

[7]Version v.2.92.323

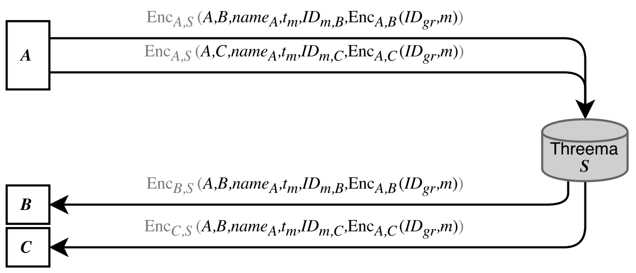

$\text{Enc}_{A,S}(A,B,name_A,t_m,ID_{m,B},\text{Enc}_{A,B}(ID_{gr},m))$
$\text{Enc}_{A,S}(A,C,name_A,t_m,ID_{m,C},\text{Enc}_{A,C}(ID_{gr},m))$
A
Threema
S
$\text{Enc}_{B,S}(A,B,name_A,t_m,ID_{m,B},\text{Enc}_{A,B}(ID_{gr},m))$
$\text{Enc}_{C,S}(A,B,name_A,t_m,ID_{m,C},\text{Enc}_{A,C}(ID_{gr},m))$
B
C

Figure 6: Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in Threema.

6.1 General Initialization Protocol

During the creation of an identity, the application of user A generates a Diffie-Hellman share pk_A^{lt}, sends it to the central key server of Threema with a fresh proof of possession of the corresponding private part and stores this private part sk_A^{lt} locally. The Diffie-Hellman share represents the long term public key of the user. It is used to authenticate the user during the session key agreement with the server and for all key agreements with other users.

6.1.1 Session Establishment with the Server

Once the application is started, a proprietary key exchange protocol is executed to derive a session key $k_{A,\mathcal{S}}^{ses}$ for the channel between the user A (client) and the Threema server $\mathcal{S}$. Both, the server's and the client's long term keys are used for the authentication. The protocol is built up on three dependent Diffie-Hellman key exchanges (DHKEs).

The session channel encryption and the end-to-end encryption are implemented with the *XSalsa20* cipher [12] with integrity and authenticity protection using the *Poly1305-AES* MAC [10].

We identified that Threema implements *Curve25519* for all DHKEs, which is also described in [66].

6.1.2 Key Agreement

A client can either request the public key of a contact from the central Threema key distribution server or scan it directly from the contact's device. In either case, two users A, B derive a symmetric contact key $k_{A,B} = \text{ECDH}(sk_A^{lt}, pk_B^{lt}) = \text{ECDH}(sk_B^{lt}, pk_A^{lt})$ from the DHKE of the long term key shares. This key is used for all direct and group messages between these two users as it can be seen in Figure 7.

6.2 Group Protocol

In Threema, only the creator U_{gr}^* of a group is the administrator $\mathcal{G}_{gr}^* = \{U_{gr}^*\}$. Threema limits the number of group members to 50 per group. Each group is uniquely referenced by ID_{gr} containing the administrator's user ID and a random bit vector, each of 64 bits.

6.2.1 Group Messages

All group messages contain the reference ID_{gr} as an identification value in the end-to-end encrypted part. The transmission is implemented the same way as for direct messages: one group message is sent to every member as a message that is encrypted with the long term contact key $k_{A,U}\ \forall\ U \in \mathcal{G}_{gr} \setminus \{A\}$ between the sender A and the respective group member (see Figure 7). These end-to-end encrypted messages are sent via the encrypted session channel between the respective users and the server. The format of a message can be seen in Figure 6 where $ID_{m,U}$ is a random message identifier for the respective receiver, t_m is a timestamp

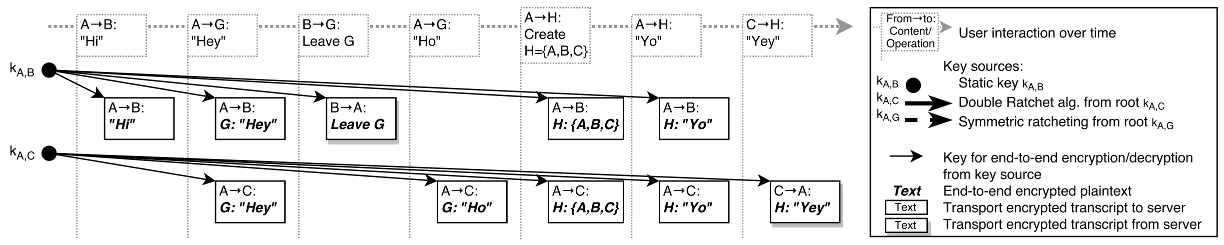

A→B: "Hi"
A→G: "Hey"
B→G: Leave G
A→G: "Ho"
A→H: Create H={A,B,C}
A→H: "Yo"
C→H: "Yey"
k_{A,B}
k_{A,C}
A→B: "Hi"
A→B: G: "Hey"
B→A: Leave G
A→B: H: {A,B,C}
A→B: H: "Yo"
A→C: G: "Hey"
A→C: G: "Ho"
A→C: H: {A,B,C}
A→C: H: "Yo"
C→A: H: "Yey"
From→to: Content/ Operation
User interaction over time
Key sources:
Static key k_{A,B}
Double Ratchet alg. from root k_{A,C}
Symmetric ratcheting from root k_{A,G}
Key for end-to-end encryption/decryption from key source
Text End-to-end encrypted plaintext
Text Transport encrypted transcript to server
Text Transport encrypted transcript from server

Figure 7: Schematic depiction of keys of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in Threema.

and $name_A$ is the readable name of A. The figure disregards message type labels on the direction and the content type of the message.

Additionally to the group ID, the end-to-end encrypted part can contain:

$$m := \begin{cases} plain\ content, & if \text{ content message} \\ \mathcal{G}_{gr}, & if \text{ update message 1} \\ info_{gr}, & if \text{ update message 2} \\ \texttt{leave}, & if \text{ leave message} \end{cases}$$

In contrast to direct messages between two users (outside of a group), content group messages are not end-to-end acknowledged: The server acknowledges the sender's messages and the receivers acknowledge the receipt towards the server. The latter acknowledgments are only encrypted by the session channel and not forwarded to the sender (i.e. the sender has no information on the receipt status).

6.2.2 Group Management

The group management is split into two protocol flows: an *update* flow and a flow that is processed for a user to *leave*. The update flow is used for the creation of a group, for adding and removing users and for changing group information like the title of a group. Note that in contrast to Signal, Threema allows the removal of other members in a group.

Group creation and update follow the same protocol consisting of two messages, sent from U^*_{gr} to $\mathcal{G}_{gr} \setminus \{U^*_{gr}\}$: (1) a message containing the new set $\mathcal{G}_{gr}$ and (2) a message containing the updated $info_{gr}$ of the group, such as the group title. The first message is sent to all users that were members until the operation is started and to all users that become a member due to the operation. The second message is only sent to users that will be members after the operation.

If a user leaves the group, she sends that information together with the group reference end-to-end encrypted to all other members.

A group member can request the administrator to synchronize the group information. The administrator then starts the group update protocol with the current group information.

6.2.3 Exemplary Protocol Run

Analogically to Signal, Threema handles group messages similarly to direct messages: a group message is sent as multiple direct messages. In contrast to Signal, a flag, which is readable by the Threema server, indicates the type of the message (e.g., group content message). As depicted in Figure 5, all group messages and all direct messages are encrypted and decrypted with the same key. There is no key derivation in Threema.

6.3 Security Evaluation and Observations

We practically carried out a replay attack on Threema with a proof-of-concept implementation. The attack breaks No Duplication and Closeness. We further observed that Threema does not achieve Perfect Forward Secrecy, Future Secrecy, or Traceable Delivery.

6.3.1 Replaying Messages

Even though a random ID is assigned to every message, messages can be resent to a group easily and thereby No Duplication is broken for the Threema group messaging protocol.

Preconditions. The attacker needs to control the channel somewhere between sender and receiver.

- ▶ *Advanced Network Attacker* can manipulate the transport layer protected traffic.

Attack Description. The attacker $\mathcal{A}$ needs to record an end-to-end encrypted message

$$(A, B, name_A, t_m, ID_{m,B}, \mathrm{Enc}_{A,B}(ID_{gr}, m))$$

once and can resend this message to sender A or receiver B later repeatedly:

$$(A, B, name_A, t'_m, ID'_{m,B}, \mathrm{Enc}_{A,B}(ID_{gr}, m))$$

$$(B, A, name_A, t'_m, ID'_{m,B}, \mathrm{Enc}_{A,B}(ID_{gr}, m))$$

Since Threema only protects the group ID and the actual content of a message on the end-to-end layer, $\mathcal{A}$ can update the timestamp (and all other unprotected metadata) and replay the encrypted message. The established encryption key is used for both directions between sender and receiver, thus, messages can be resent to the receiver and to the sender.

Attack Impact. The attack violates the following security goals:

- ▶ *No dublication* $\mathcal{A}$ can replay messages.
- ▶ *Closeness* This weakness also affects the Closeness of a group because $\mathcal{A}$ can rewind every group manipulation by resending previous group update messages. For example, $\mathcal{A}$ can rewind the removal of a group member.

6.3.2 No Forward and Future Secrecy

In Threema, every message between two users is encrypted with the same key, derived from the DHKE of their long term public keys. Consequently no security property can be reached when considering *compromising attackers*.

6.3.3 No Traceable Delivery

The Threema application provides no information on the receipt status of sent group messages. Consequently this property cannot be attacked.

Receivers actually acknowledge group messages only towards the server. As a result, the sender cannot verify the message status such that *delivery* in Threema cannot be *traced*.

6.3.4 Further Weaknesses

Weaknesses of Threema regarding *ordering*, *agreement* and the confidentiality of meta data are described in subsection A.4.

7 Lessons Learned

In this section we first briefly describe specific fixes for the analyzed protocols and then evaluate general approaches for reaching the security properties efficiently.

7.1 Fixing the Protocols

7.1.1 Signal

Closeness and No Creation. In Signal, Closeness can be reached by implementing a simple check when receiving a group message. If the sender is not part of the current group, the message is dropped. This efficiently preserves No Creation and Closeness. As a side effect, the group ID can then be public knowledge. We discussed this proposal with Open WhisperSystems, but unfortunately it turned out, that due to their current implementation, this verification is impossible. Open WhisperSystems is currently developing a new group management system with advanced administrative features so that they decided not apply our fix.

Traceable Delivery. Signal could reach Traceable Delivery by treating receipt messages like content messages and thus end-to-end encrypt them[8]. This would guarantee the authenticity of these messages. We will discuss and compare this approach with the usage of the properties of stateful encryption (see subsubsection 7.2.1).

7.1.2 WhatsApp

Closeness. In order to ensure that only administrators of a group can manipulate the member set, the authenticity of group manipulation messages needs to be protected. This can be achieved, for example, by signing these messages with the administrator's group signature key.

In order to maintain the member set at the server with regard to an advanced network attacker, a counter for the current modification step could be attached to every message and the signed manipulation notification could include the whole member set instead of its changes only. Thereby, the current signed notification could be distributed when a member loses their information of the group (e.g., due to a re-installation).

Traceable Delivery. The same countermeasure that is described for Signal applies to WhatsApp for providing Traceable Delivery.

7.1.3 Threema

No Duplication. Since there is already a message ID appended to every message, this ID only needs to be cryptographically bound to the message. This would prevent that one message is accepted by the client multiple times. We proposed this fix to the developers of Threema. They appreciated our effort and implemented a fix in Version 3.14[9].

7.2 General Outcomes

7.2.1 Reaching Traceable Delivery in General

The result of our analysis shows that Traceable Delivery in SIM protocols is seldom reached. Without going into detail, we also analyzed the respective direct messaging protocols regarding Traceable Delivery. Signal and WhatsApp do not reach Traceable Delivery, but the direct messaging in Threema reaches it by end-to-end encrypting the receipt acknowledgment to the sender and thereby cryptographically ensuring the authenticity of these acknowledgments.

[8]Signing the message would be sufficient but the encryption is already part of the protocol and additionally protects the confidentiality of the receipt messages.

[9]`https://threema.ch/en/versionhistory`

Using the approach of negative acknowledgments (NACKs) turns the responsibility of the Traceable Delivery from the sender to the receiver [1, 28, 32, 42]. The receiver can therefore use the Signal key exchange protocol since it is stateful. It provides a consecutive key stream such that Traceable Delivery can be reached by detecting an omitted key of this stream. Once a key is omitted, the receiver knows that a message was not delivered such that she can request the sender to resend this message. This approach however fails if the communication is disrupted for ever because thereby the sender never knows that the receiver did not receive the message.

7.2.2 Securely Managing a Group

In order to reach Closeness and No Creation in groups, members of a group need to distinguish between group members and outstanding users.

We see two major approaches of a secure group management:

(1) A consistent view on the member set for each of its members.

(2) A group secret that serves as a proof of membership.

Abstractly this means that either the receiver always checks her *guest list* or a sender always provides a *ticket*. While Signal only implements the second mechanism, Threema mainly uses the first one. WhatsApp somehow follows the *guest list* approach while the guest list in manipulable from outside.

Consistent View. For the effective group management, group information needs to be maintained locally on every member's device. Each user knows, who is part of the group, that means, who is allowed to write a group message and from whom group messages should be accepted. In order to ensure a consistent view on the member set, Traceable Delivery must be achieved because otherwise the server provider can drop messages that aim to manipulate the member set and thereby cause an inconsistent view. Even if the group information is centrally stored, it needs to be ensured, that (1) only members can modify this information and (2) all members are informed about a modification.

Schiper and Toueg showed that the problem of membership in groups can be reduced to the more general problem of maintaining a set of arbitrary elements and thereby decouple the group from the protocol [56]. Similarly we argue that a protocol, reaching consistency of all messages (content and group management), can be treated as a protocol considering static groups. Nevertheless the consistent message delivery in groups restricts the instant communication for SIM protocols.

Membership Proof. When solely using a group secret that protects Closeness and No Creation of the group, this secret needs to be calculated future secure, when the whole protocol reaches this property. Otherwise, a revealed group secret can be used to become part of the group without the members' permission.

This target is related to future secure group key exchange. A first group key exchange with this property was recently proposed by Cohn-Gordon et al. [24].

8 Related Work

Related work to this paper is structured in (1) analyses of IM applications in general, specific analyses of the analyzed protocols, as well as (2) theoretical concepts in multi user settings.

Analyses of IM Applications. Schrittwieser et al. [57] analyze IM applications regarding the initial authentication and the account management and describe weaknesses accordingly. Unger et al. [67] systematize current SIM application solutions by proposing an evaluation security framework. Regarding group communications, they conduct only a high level investigation on basic concepts and features of the protocols.

Analyses of Signal. The analysis of Signal started with Frosch et al. [33]. They analyze TextSecure v2, the predecessor of the Signal key exchange protocol. As a result, they identify an Unknown Key-Share (UKS) attack and propose fixes. Kobeissi et al. [40] describe the application of formal verification software for analyzing a slightly modified version of the Signal protocol and other real world protocols. They derive a proof from an automatic cryptographic verification tool but also model the UKS of Frosch et al. and present attacks on the protocol that go beyond the model for the proof. Cohn-Gordon et al. [23] conduct a formal analysis on the Signal key exchange protocol. Therefore, they develop a new multi-stage key exchange security model, identify security properties in the Signal protocol, and prove it to be secure. Previous to their analysis, they published a work on definitions and constructions for Future Secrecy [25]. Bellare et al. [8] investigate ratcheting as a cryptographic primitive. Their work does not specifically focus on a real world protocol, but forms the basis of a definition and application for this primitive.

All these works concentrate on two party communications instead of multi-user setups. For this reason, the security goals identified in this work differ significantly.

Analyses of WhatsApp. Schrittwieser et al. [57] analyzed WhatsApp among other IM applications regarding the authentication and account management and found several vulnerabilities. Another application specific analysis [4] focused on WhatsApp's Android application. A recent newspaper article described that, even though key verification is implemented in WhatsApp, its effectiveness can partially be circumvented for usability reasons [43, 50]. In addition to these analyses, the WhatsApp protocol was implemented and published as open source projects [34, 47].

Analyses of Threema. An initial analysis of the Threema protocol was conducted by Ahrens [2]. Based on this Berger [9] implemented an open source desktop client on which we based our protocol analysis. Independent of our work Schilling and Steinmetz presented a detailed description of the Threema message format and another open source implementation [55].

Security in Multi User Settings. *Multiparty non-interactive key exchange (mNIKE)* solves the issue of instantly agreeing on a group secret. Certainly all known mNIKE protocols for an unbounded number of users base on problems, for which no practical solutions exist [13, 14, 37].

Group key agreement protocols solve this problem by generating a key among the group members [16, 20, 29, 39]. These protocols, however, need the interaction of the users before the communication begins and are thereby not practical in an asynchronous environment.

Cohn-Gordon et al. [24][10] recently published a group key exchange protocol that enables the future secure ratcheting of a group secret. This protocol is a hybrid of multiple two-party protocols for the instantiation and a refreshable group key agreement. They also proof parts of their construction.

Bracha and Toueg introduced the notion of *reliable broadcast* in the asynchronous setting [15]. Since then many works introduced and improved algorithms to solve the problem of validly and consistently delivering messages in a multi user setting [18, 19, 21, 41] but also refined the notion and definition to provide realistic attacker models [18].

Chockler et al. [22] give an overview on various models and results regarding group communication systems (GCS) like [48, 68] and others. They systematize different notions and definitions regarding the reliability and security of group communication in the literature.

9 Conclusion

Nowadays, Instant Messaging (IM) applications rely more and more on end-to-end protection. Although the one-to-one communication of Secure Instant Messaging (SIM) applications has been in the focus of recent analyses [23, 33, 40], the investigation of end-to-end protected group communications has gained only little attention.

We fill this gap and provide a systematical analyses of the three major SIM applications Signal, WhatsApp, and Threema. This analysis provides a methodology for further examinations of similar protocols. While our investigation focuses on three major SIM applications, our methodology and underlying attack model is of generic purpose and can be applied to other SIM group protocols as well. For example, it would be interesting to analyze the group chat implementations of other Signal-based SIM applications, such as

[10]Cohn-Gordon et al. [24] describe Signal's group communication protocol differently than we do. We verified our protocol description with the latest versions of July 24, 2017.

Google's Allo and Facebook Messenger, or even non Signal-based protocols similarly to our investigation of Threema.

For one-to-one communication the Signal key exchange protocol is practically used and cryptographically proven secure. In contrast to this, for group communication no such protocol exists. A cryptographically future secure group key exchange was recently published [24]. Still on the one hand, this protocol was designed for a partially asynchronous setting and on the other hand, our work shows that the key exchange is only a building block for secure and reliable group SIM protocol. In fact, we demonstrate that *Future Secrecy* should not only be restricted to the establishment of a common secret for encryption.

Consequently our work can be seen as a structural survey, a base point and an illustration of a target for the design of secure and reliable group SIM protocols.

References

[1] B. Adamson, C. Bormann, M. Handley, and J. Macker. 2008. *Multicast Negative-Acknowledgment (NACK) Building Blocks*. RFC 5401. IETF. 1–42 pages. `https://tools.ietf.org/html/rfc5401`

[2] Jan Ahrens. 2014. Threema protocol analysis. (2014). `http://blog.jan-ahrens.eu/files/threema-protocol-analysis.pdf`

[3] Martin R. Albrecht and Kenneth G. Paterson. 2016. Lucky Microseconds: A Timing Attack on Amazon's s2n Implementation of TLS. In *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part I (Lecture Notes in Computer Science)*, Marc Fischlin and Jean-Sébastien Coron (Eds.), Vol. 9665. Springer, 622–643. `https://doi.org/10.1007/978-3-662-49890-3_24`

[4] Cosimo Anglano. 2014. Forensic analysis of WhatsApp Messenger on Android smartphones. *Digital Investigation* 11, 3 (2014), 201–213. `https://doi.org/10.1016/j.diin.2014.04.003`

[5] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. 2016. DROWN: Breaking TLS Using SSLv2. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 689–706. `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/aviram`

[6] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. 1996. Keying Hash Functions for Message Authentication. In *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings.* 1–15. `https://doi.org/10.1007/3-540-68697-5_1`

[7] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. 1997. A Concrete Security Treatment of Symmetric Encryption. In *38th Annual Symposium on Foundations of Computer Science, FOCS '97, Miami Beach, Florida, USA, October 19-22, 1997.* 394–403. `https://doi.org/10.1109/SFCS.1997.646128`

[8] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. 2016. Ratcheted Encryption and Key Exchange: The Security of Messaging. *IACR Cryptology ePrint Archive* 2016 (2016), 1028. `http://eprint.iacr.org/2016/1028`

[9] Philipp Berger. 2016. Open Source Implementation of a Threema Desktop Client. (2016). `https://github.com/blizzard4591/openMittsu` Based on the descriptions of Jan Ahrens' paper.

[10] Daniel J. Bernstein. 2005. The Poly1305-AES Message-Authentication Code. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*. 32–49. `https://doi.org/10.1007/11502760_3`

[11] Daniel J. Bernstein. 2006. Curve25519: New Diffie-Hellman Speed Records. In *Public Key Cryptography - PKC 2006, 9th International Conference on Theory and Practice of Public-Key Cryptography, New York, NY, USA, April 24-26, 2006, Proceedings*. 207–228. `https://doi.org/10.1007/11745853_14`

[12] Daniel J. Bernstein. 2008. The Salsa20 Family of Stream Ciphers. In *New Stream Cipher Designs - The eSTREAM Finalists*, Matthew J. B. Robshaw and Olivier Billet (Eds.). Lecture Notes in Computer Science, Vol. 4986. Springer, 84–97. `https://doi.org/10.1007/978-3-540-68351-3_8`

[13] Dan Boneh and Alice Silverberg. 2003. Applications of multilinear forms to cryptography. *Contemp. Math.* 324, 1 (2003), 71–90.

[14] Dan Boneh and Mark Zhandry. 2014. Multiparty Key Exchange, Efficient Traitor Tracing, and More from Indistinguishability Obfuscation. In *Advances in Cryptology - CRYPTO 2014 - 34th Annual Cryptology Conference, Santa Barbara, CA, USA, August 17-21, 2014, Proceedings, Part I*. 480–499. `https://doi.org/10.1007/978-3-662-44371-2_27`

[15] Gabriel Bracha and Sam Toueg. 1985. Asynchronous Consensus and Broadcast Protocols. *J. ACM* 32, 4 (1985), 824–840. `https://doi.org/10.1145/4221.214134`

[16] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. 2002. Dynamic Group Diffie-Hellman Key Exchange under Standard Assumptions. In *Advances in Cryptology - EUROCRYPT 2002, International Conference on the Theory and Applications of Cryptographic Techniques, Amsterdam, The Netherlands, April 28 - May 2, 2002, Proceedings*. 321–336. `https://doi.org/10.1007/3-540-46035-7_21`

[17] Business2Community. 2015. Are Instant Messaging Apps the Future of the (Mobile) Internet? (Aug. 2015). `http://www.business2community.com/mobile-apps/instant-messaging-apps-future-mobile-internet-01313577`

[18] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 2001. Secure and Efficient Asynchronous Broadcast Protocols. In *Advances in Cryptology - CRYPTO 2001, 21st Annual International Cryptology Conference, Santa Barbara, California, USA, August 19-23, 2001, Proceedings*. 524–541. `https://doi.org/10.1007/3-540-44647-8_31`

[19] Christian Cachin, Klaus Kursawe, and Victor Shoup. 2005. Random Oracles in Constantinople: Practical Asynchronous Byzantine Agreement Using Cryptography. *J. Cryptology* 18, 3 (2005), 219–246. `https://doi.org/10.1007/s00145-005-0318-0`

[20] Christian Cachin and Reto Strobl. 2004. Asynchronous group key exchange with failures. In *Proceedings of the Twenty-Third Annual ACM Symposium on Principles of Distributed Computing, PODC 2004, St. John's, Newfoundland, Canada, July 25-28, 2004*. 357–366. `https://doi.org/10.1145/1011767.1011820`

[21] Ran Canetti and Tal Rabin. 1993. Fast asynchronous Byzantine agreement with optimal resilience. In *Proceedings of the Twenty-Fifth Annual ACM Symposium on Theory of Computing, May 16-18, 1993, San Diego, CA, USA*. 42–51. `https://doi.org/10.1145/167088.167105`

[22] Gregory V. Chockler, Idit Keidar, and Roman Vitenberg. 2001. Group communication specifications: a comprehensive study. *ACM Comput. Surv.* 33, 4 (2001), 427–469. `https://doi.org/10.1145/503112.503113`

[23] Katriel Cohn-Gordon, Cas Cremers, Benjamin Dowling, Luke Garratt, and Douglas Stebila. 2017. A formal security analysis of the Signal messaging protocol. In *Proc. IEEE European Symposium on Security and Privacy (EuroS&P) 2017*. IEEE. To appear.

[24] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. 2017. On Ends-to-Ends Encryption: Asynchronous Group Messaging with Strong Security Guarantees. *IACR Cryptology ePrint Archive* 2017 (2017), 666. `http://eprint.iacr.org/2017/666`

[25] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. 2016. On Post-compromise Security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*. 164–178. `https://doi.org/10.1109/CSF.2016.19`

[26] Joan Daemen and Vincent Rijmen. 1998. The Block Cipher Rijndael. In *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*. 277–284. `https://doi.org/10.1007/10721064_26`

[27] Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith. 2016. An Empirical Study of Textual Key-Fingerprint Representations. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, Thorsten Holz and Stefan Savage (Eds.). USENIX Association, 193–208. `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/dechand`

[28] Christophe Diot, Walid Dabbous, and Jon Crowcroft. 1997. Multipoint Communication: A Survey of Protocols, Functions, and Mechanisms. *IEEE Journal on Selected Areas in Communications* 15, 3 (1997), 277–290. `https://doi.org/10.1109/49.564128`

[29] Ratna Dutta and Rana Barua. 2008. Provably Secure Constant Round Contributory Group Key Agreement in Dynamic Setting. *IEEE Trans. Information Theory* 54, 5 (2008), 2007–2025. `https://doi.org/10.1109/TIT.2008.920224`

[30] Electronic Frontier Foundation. 2017. Secure Messaging Scorecard. (2017). `https://www.eff.org/node/82654`

[31] eMarketer. 2015. Mobile Messaging to Reach 1.4 Billion Worldwide in 2015. (Nov. 2015). `https://www.emarketer.com/Article/Mobile-Messaging-Reach-14-Billion-Worldwide-2015/1013215`

[32] Sally Floyd, Van Jacobson, Ching-Gung Liu, Steven McCanne, and Lixia Zhang. 1997. A reliable multicast framework for light-weight sessions and application level framing. *IEEE/ACM Trans. Netw.* 5, 6 (1997), 784–803. `https://doi.org/10.1109/90.650139`

[33] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. 2016. How Secure is TextSecure?. In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*. 457–472. `https://doi.org/10.1109/EuroSP.2016.41`

[34] Tarek Galal. 2016. Open Source Implementation of a WhatsApp Client. (2016). `https://github.com/tgalal/yowsup` This code is not maintained anymore but some of its forks are still under development.

[35] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. 2016. Dancing on the Lip of the Volcano: Chosen Ciphertext Attacks on Apple iMessage. In *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.* 655–672. `https://www.usenix.org/conference/usenixsecurity16/technical-sessions/presentation/garman`

[36] Vassos Hadzilacos and Sam Toueg. 1993. Distributed Systems (2Nd Ed.). ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, Chapter Fault-tolerant Broadcasts and Related Problems, 97–145. `http://dl.acm.org/citation.cfm?id=302430.302435`

[37] Dennis Hofheinz, Tibor Jager, Dakshita Khurana, Amit Sahai, Brent Waters, and Mark Zhandry. 2016. How to Generate and Use Universal Samplers. In *Advances in Cryptology - ASIACRYPT 2016 - 22nd International Conference on the Theory and Application of Cryptology and Information Security, Hanoi, Vietnam, December 4-8, 2016, Proceedings, Part II*. 715–744. `https://doi.org/10.1007/978-3-662-53890-6_24`

[38] WhatsApp Inc. 2016. WhatsApp Encryption Overview. (2016). `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf` Technical white paper.

[39] Hyun-Jeong Kim, Su-Mi Lee, and Dong Hoon Lee. 2004. Constant-Round Authenticated Group Key Exchange for Dynamic Groups. In *Advances in Cryptology - ASIACRYPT 2004, 10th International Conference on the Theory and Application of Cryptology and Information Security, Jeju Island, Korea, December 5-9, 2004, Proceedings*. 245–259. `https://doi.org/10.1007/978-3-540-30539-2_18`

[40] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. 2017. Automated Verification for Secure Messaging Protocols and their Implementations: A Symbolic and Computational Approach. In *IEEE European Symposium on Security and Privacy (EuroS&P). Available at http://prosecco. gforge. inria. fr/personal/bblanche/publications/KobeissiBhargavanBlanchetEuroSP17. pdf. To appear.*

[41] Klaus Kursawe and Victor Shoup. 2005. Optimistic Asynchronous Atomic Broadcast. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*. 204–215. `https://doi.org/10.1007/11523468_17`

[42] Brian Neil Levine and J. J. Garcia-Luna-Aceves. 1998. A Comparison of Reliable Multicast Protocols. *Multimedia Syst.* 6, 5 (1998), 334–348. `http://link.springer.de/link/service/journals/00530/bibs/8006005/80060334.htm`

[43] Tobias Boelter Manisha Ganguly. 2017. WhatsApp vulnerability allows snooping on encrypted messages. *The Guardian* (2017). `https://www.theguardian.com/technology/2017/jan/13/whatsapp-backdoor-allows-snooping-on-encrypted-messages`

[44] Moxie Marlinspike and Trevor Perrin. 2016. The Double Ratchet Algorithm. (11 2016). `https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf`

[45] Moxie Marlinspike and Trevor Perrin. 2016. The X3DH Key Agreement Protocol. (11 2016). `https://whispersystems.org/docs/specifications/x3dh/x3dh.pdf`

[46] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. 2014. Revisiting SSL/TLS Implementations: New Bleichenbacher Side Channels and Attacks. In *23rd USENIX Security Symposium*. 733–748. `https://www.usenix.org/conference/usenixsecurity14/technical-sessions/presentation/meyer`

[47] mgp25. 2016. Open Source Implementation of a WhatsApp PHP API. (2016). `https://github.com/mgp25/Chat-API`

[48] Louise E. Moser, P. M. Melliar-Smith, Deborah A. Agarwal, Ravi K. Budhia, and Colleen A. Lingley-Papadopoulos. 1996. Totem: A Fault-Tolerant Multicast Group Communication System. *Commun. ACM* 39, 4 (1996), 54–63. `https://doi.org/10.1145/227210.227226`

[49] Moxie Marlinspike. 2013. Advanced cryptographic ratcheting. (2013). `https://whispersystems.org/blog/advanced-ratcheting/`

[50] Moxie Marlinspike. 2017. There is no WhatsApp 'backdoor'. (2017). `https://whispersystems.org/blog/there-is-no-whatsapp-backdoor/`

[51] Open Whisper Systems. 2017. Facebook Messenger deploys Signal Protocol for end to end encryption. (2017). https://whispersystems.org/blog/facebook-messenger/

[52] Open Whisper Systems. 2017. Open Whisper Systems partners with Google on end-to-end encryption for Allo. (2017). https://whispersystems.org/blog/allo/

[53] Open Whisper Systems. 2017. Signal Website. (2017). https://signal.org/

[54] Trevor Perrin. 2016. The Noise Protocol Framework. (2016). http://noiseprotocol.org/noise.pdf

[55] Roland Schilling and Frieder Steinmetz. 2016. A look into the Mobile Messaging Black Box. (12 2016). https://media.ccc.de/v/33c3-8062-a_look_into_the_mobile_messaging_black_box Talk at the 33c3 in Hamburg, Germany. Implementation: https://github.com/o3ma.

[56] André Schiper and Sam Toueg. 2006. From Set Membership to Group Membership: A Separation of Concerns. *IEEE Trans. Dependable Sec. Comput.* 3, 1 (2006), 2–12. https://doi.org/10.1109/TDSC.2006.13

[57] Sebastian Schrittwieser, Peter Frühwirt, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, and Edgar R. Weippl. 2012. Guess Who's Texting You? Evaluating the Security of Smartphone Messaging Applications. In *19th Annual Network and Distributed System Security Symposium, NDSS 2012, San Diego, California, USA, February 5-8, 2012.* http://www.internetsociety.org/guess-whos-texting-you-evaluating-security-smartphone-messaging-applications

[58] Signal. 2017. Signal Private Messenger in Google Play. (2017). https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms

[59] Statista. 2017. Most popular messaging apps. (2017). https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/

[60] Open Whisper Systems. 2016. Message Format in the Signal Protocol. (11 2016). https://github.com/WhisperSystems/libsignal-service-java/blob/4cedb5c31c11c1e8811b3bb7cd68d56ff7e0c03f/protobuf/SignalService.proto Specified with Google Protocol Buffers.

[61] Open Whisper Systems. 2016. Source Code of Signal-Android. (11 2016). https://github.com/WhisperSystems/Signal-Android/commit/ce812ed8ba49fc43db9de018c135be67b5b44f7d Android Application Version 3.23.0.

[62] Open Whisper Systems. 2016. Source Code of Signal-Service Library. (11 2016). https://github.com/WhisperSystems/libsignal-service-java/commit/460cd7559caa74bb6539c72865c71de660a69bac Java Library Version 2.4.1.

[63] Open Whisper Systems. 2017. Signal Github Repository. (05 2017). https://github.com/WhisperSystems/

[64] Threema. 2017. Threema in Google Play. (2017). https://play.google.com/store/apps/details?id=ch.threema.app

[65] Threema. 2017. Threema Website. (2017). https://threema.ch/en

[66] Threema GmbH. 2016. Threema Cryptography Whitepaper. (2016). https://threema.ch/press-files/2_documentation/cryptography_whitepaper.pdf

[67] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: Secure Messaging. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015.* 232–249. https://doi.org/10.1109/SP.2015.22

[68] Robbert van Renesse, Kenneth P. Birman, and Silvano Maffeis. 1996. Horus: A Flexible Group Communication System. *Commun. ACM* 39, 4 (1996), 76–83. `https://doi.org/10.1145/227210.227229`

[69] WhatsApp. 2016. Why am I banned for using WhatsApp Plus and how do I get unbanned? (2016). `https://www.whatsapp.com/faq/en/general/105`

[70] WhatsApp. 2017. WhatsApp Security. (2017). `https://www.whatsapp.com/security/`

A Additional Security and Reliability Properties

A.1 Definitions

A.1.1 Message Order

A primary feature regarding the consistency of communicated transcripts is ordering of messages.

▶ *FIFO Order* : All messages of one user are delivered in the same order in which this user sent them.

While *FIFO order* only considers the order of messages sent by one member, *total order* can only be achieved by constructing a protocol that establishes an order for the whole content of a group. The aim of ordering a group's content is twofold. On the one hand it presents a consistent view on delivered messages among the receivers if the sender is correct. On the other hand it prevents the breach of the existing order such that the causality of messages is preserved. For real time protocols a certain delay of the order establishment should therefore be granted.

▶ *Total Order* : All receivers of two messages decide on the same order between these two messages.

In addition to the *local* ordering schemes, there exists the definition *global* ordering in multicast settings. *Global* ordering implies that all messages, regardless in which group or multicast they were delivered, have a fixed order. In instant messaging this could be of interest if a message in one group is sent because of a message that was received in another group.

A.1.2 Reliable Multicast

Even though we disregard consistency of messages and group management in the presence of malicious group members, we give the definition of *agreement* for completeness.

▶ *AGR Agreement*: if a message m is delivered by one correct member $A \in \mathcal{G}_{gr}$ then m is delivered by all correct members $B \in \mathcal{G}_{gr}$.

In fact two of the protocols gain a weak form consistency in the presence of malicious senders. WhatsApp achieves agreement in case of an honest SIM server since a sender can only send one message to the group and the server has to distribute it to the receiving members. Threema restricts the permission to manipulate group management information to the creator of the group. Consequently only a malicious group creator can break the agreement regarding group management information in the long run.

A.2 Limitations of Signal

Ordering. A malicious *server provider* cannot only drop messages, but also reorder them. The receiving application orders simultaneously received messages by the timestamp which is manipulable for the provider. The decryption of received messages follows this order. Since old omitted keys are removed after a certain number of new keys are derived, reordering by the provider is restrictedly possible. Since the number of acceptably omitted keys is also at least 2000, this restriction is not effective. Henceforth neither *FIFO order* nor *total order* are provided by Signal.

Agreement. Signal does not implement an algorithm for providing *agreement.* Consequently the server provider and members can cause inconsistency of messages and of the member set in groups.

A.3 Limitations of WhatsApp

Ordering. In contrast to Signal and Threema, the sending time for a message is set at the server side. The honest server transmits messages to the receivers in the order the server received them from the sender. The receiving clients decrypt and display the messages in the order the server transmits them.

If messages are received in a different order than they were encrypted, this is disregarded by the client as the omission of message keys is. As a result a malicious server provider cannot only drop messages but also reorder them because they are not listed in the order of encryption but in the order of transmission by the server. Due to the limit of decryption of old messages up to 1999 messages can be reordered by the server. On the one hand this results in an immutable list of messages and a live display of received messages but on the other hand the server provider can disrupt the sender's order of messages.

A.4 Limitations of Threema

Ordering. Messages received by the application are ordered by the receiving time. The sending time is additionally not protected on the end-to-end layer. Therefore the server provider can reorder messages arbitrarily during the transmission.

Agreement. Threema does not implement an algorithm for providing *agreement* as Signal does not. Consequently the server provider and members can cause inconsistency of messages. The member set is managed by the administrator who can inconsistently provide a view on it to the members.

Additional Information Leakage. When a user in Threema sends a message to a group of which she is not a member, this message is not accepted by its members. In order to indicate this non-member status, the group administrator starts the group update protocol and sends both the set of members and the title to this user in response. A user who left the group or who was removed from the group can thereby keep informed about the group's management information. This weakness was also fixed in Threema version 3.14.

B Tabular Summary of Analysis Results

	E2E Confidentiality	Forward Secrecy	Future Secrecy	Msg. Authentication	Traceable Delivery	No Dublication	No Creation	Closeness
Signal			👤⚡		🗄		👤⚡	👤⚡
WhatsApp			✗		🗄		🗄	🗄
Threema		✗	✗		✗	🗄	🗄	🗄

Table 1: ✗: Not implemented feature; 👤⚡: Attackable by *Malicious User* who can compromise victim; 🗄: Attackable by *Advanced Network Attacker*; Gray symbols indicate that attack is side effect of another attack

Similar to previous analyses of SIM applications and protocols [30, 67], we provide a table that summarizes our results. Even though we found significant heuristic arguments that support the fact that non-attacked requirements are fulfilled by the protocols, we do not mark them as such since our analysis contains no proofs. Thereby Table 1 can be seen as a negative scorecard.

The gray cells indicate, that a feature is attackable because another feature is attackable. Since a compromising malicious user can break the Closeness of Signal, Future Secrecy is implicitly violated as well. In comparison to Signal, where Closeness and No Creation are independently attackable, breaking No Creation in WhatsApp results from breaking Closeness. In Threema breaking No Creation results from breaking Closeness which is caused by the successful attack against No Duplication. This is because an attacker can rewind group management operations and thereby add removed users who are then able to contribute content to the group.

According to our attack description, we always depict the weakest successful attacker. Consequently a *Compromising Advanced Network Attacker* can break the same requirements as a *Malicious User* with compromising access to the victim's secrets.

As described earlier, Threema updated their application in response to our responsible disclosure. Consequently No Duplication, No Creation, and Closeness are not attackable anymore.

C Signal Key Exchange Protocol and its Usage

Figure 8 describes the exact usage of keys for group communication in Signal and WhatsApp. The DR algorithm is initialized by the root key (RK) and is updated by Diffie Hellman key exchanges between the sender and the receiver (DH ratcheting). The output of these updates is the input of the symmetric ratcheting which only consists of a key derivation function. Half of the output is used for the consecutive ratcheting (chain keys CK) and the other half is used as encryption keys (message keys MK). While Signal uses these keys directly for all communication, WhatsApp generates a separate key stream for group communication. This additional key stream is update symmetrically only.

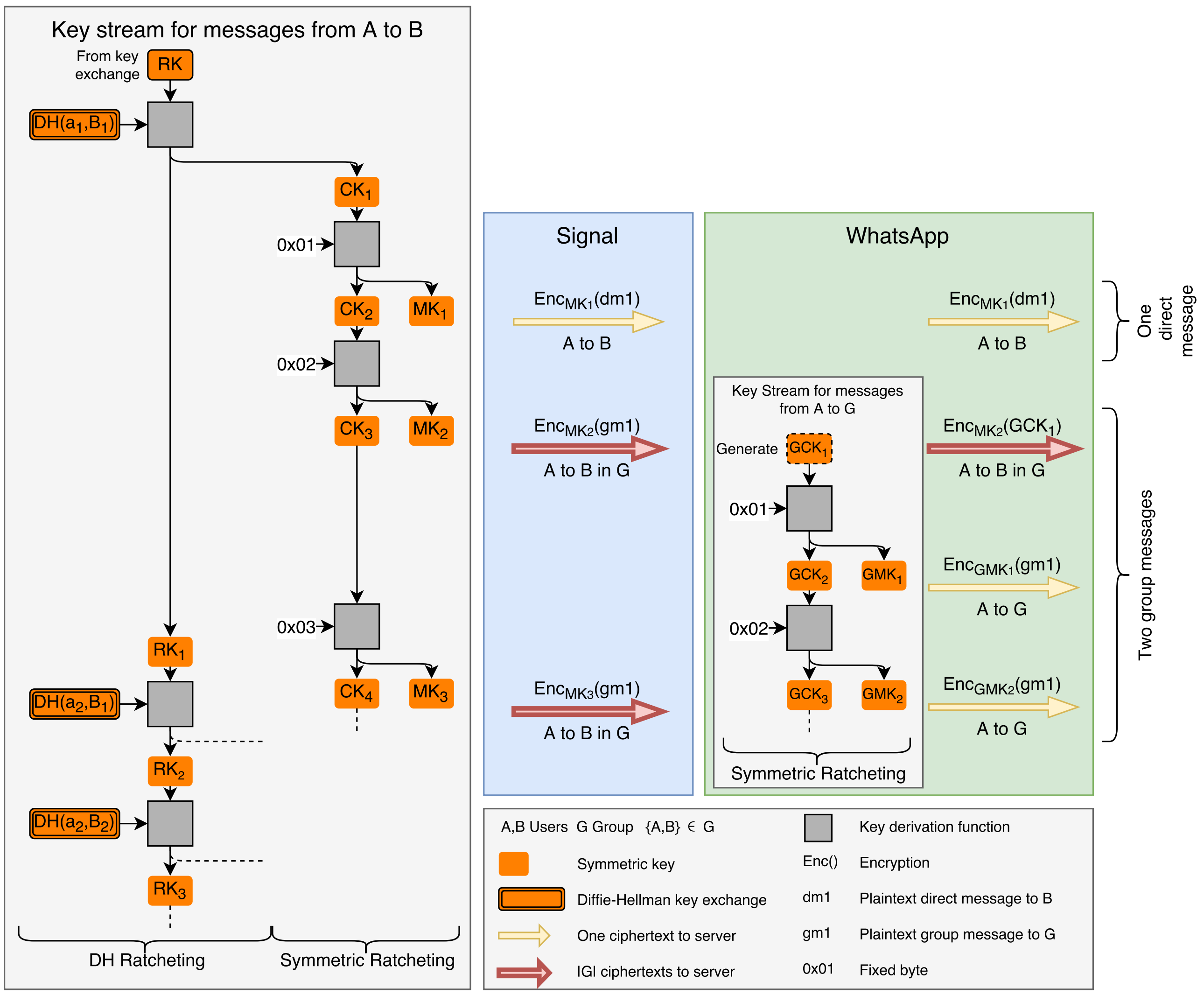


Figure 8: Sender key stream from A to B and ciphertexts from A to the server when sending one direct message to B and two group messages to a group G of which A and B are members in Signal and WhatsApp.