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

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

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

To approach an investigation of group instant messaging protocols, we first provide a comprehensive and realistic security model. This model combines security and reliability goals from various related literature to capture relevant properties for communication in dynamic groups. Thereby the definitions consider their satisfiability with respect to the instant delivery of messages. To show its applicability, we analyze three widely used real-world protocols: Signal, WhatsApp, and Threema. By applying our model, we reveal several shortcomings with respect to the security definition. Therefore we 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 dominate the messaging sector today [4, 5, 6].

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 all group members.

With the revelation of mass surveillance activities by intelligence agencies, new IM applications incorporating end-to-end encryption launched, as well es established IM applications added encryption to their protocols to protect the communication towards the message delivering servers. Hence analyses, investigating these protocols, also include malicious server-based attacks [3, 7].

In contrast to open standardized communication protocols like Extensible Messaging and Presence Protocol (XMPP) or 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 the major security feature of secure instant messaging protocols for protecting the protocol security when considering malicious server-based attacks. Additionally further security properties like *future secrecy* have been claimed [8], analyzed [1], and proven [2]. *Forward secrecy* and *future secrecy* are properties that describe the preservation or recovery of security if user secrets are leaked to the attacker at a later (resp. earlier) point of time. End-to-end encryption is part of all major IM apps, including Signal [9], WhatsApp [10], Threema [11], Google Allo [12], and Facebook Messenger [13]. One of the main achievements of secure instant messengers is the *usability* of its end-to-end encryption. After the application installation, keys are automatically generated, and encryption is (or can easily be) enabled. Experienced users may do some simple checks to verify the public key of their counterpart [14], 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.

The fact that widely used secure instant messenger protocols are neither open source nor standardized makes it

DOI 10.1109/EuroSP.2018.00036

harder to analyze and compare their security properties. This leads to two major challenges. First, the applications must be reverse engineered [1, 15, 16] for retrieving a protocol description. Second, third-party implementations are often blocked by providers [17] 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 members. 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 receive messages from 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* [18, 19] and related formalizations like group communication systems (GCS) [20] 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 at costs of the instant delivery of messages. Therefore, the modeling of our security and reliability definitions bases on the related literature and the satisfiability of real-world requirements such as asynchronous communication and instant message delivery. For this purpose we also considered representative secure instant messengers by extracting security properties from their features (provider statements or visual user interface). We matched these properties 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 security model for the investigation of secure group instant messaging protocols.

We investigate three popular secure instant messengers: Signal [9], Threema [11], and WhatsApp [10]. Signal can be seen as a reference implementation for other secure instant messenger 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 instant messenger applications with more than one billion users [21]. We additionally chose to analyze Threema as a widely used representative for the class of proprietary and closed source instant messengers – not implementing the Signal key exchange protocol. Signal and Threema are both used by at least one million Android users [22, 23]. Based on this examination, we apply our model and evaluate the security properties. In our systematical analysis, we reveal several discrepancies between the security definition of our model and the security provided by the group communication protocols of these applications.

Our contributions are outlined as follows:

- ► We present and discuss a realistic and comprehensive security model for the analysis of group communication in secure instant messenger protocols (§ 2). Therefore we employ definitions from related literature and fit them to the setting of *instant* message delivery in groups. As such, we lift strong notions of *reliability* to a realistic model and combine them with well established *security* goals to introduce a formalism that is applicable for real-world protocols.
- ► We describe the group communication protocols of Signal (§4), WhatsApp (§5), and Threema (§6) and thereby present three fundamentally different protocols for secure and instant group communication to enable further scientific analyses.
  We analyze them by applying our model and thereby reveal several insufficiencies showing that traceable delivery, closeness and thereby confidentiality of their group chat implementations are not achieved. 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

Secure instant messaging protocols should satisfy the general security goals *Confidentiality*, *Integrity*, *Authenticity* and *Reliability*. Some of them even claim advanced security goals like *Future Secrecy*.

One could expect that protocols for group communication reach the same properties – as well as several others –, that are naturally achieved in a two-party scenario. Intuitively, a secure group communication protocol should provide a level of security comparable to when a group of people communicates in an isolated room: everyone in the room hears the communication (*traceable delivery*), everyone knows who spoke (*authenticity*) and how often words have been said (*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*).

Even though some of these requirements seem to be well understood, it is essential for a comprehensive analysis to take them into account.

2.1. Notation and Assumptions

The instant messenger protocols in scope are *centralized*: all exchanged messages are transmitted via a central server, that receives messages from the respective senders, caches them, and forwards them as soon as the receivers are

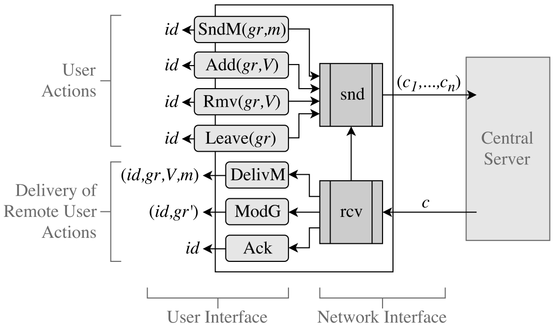


Figure 1. Overview over syntax of group instant messaging protocols showing the interacting user's interfaces on left and the interfaces of the application to the network on the right.

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

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

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 $A, B, C, .., U, .., X \in \mathcal{U}$ and an administrator as $U^* \in \mathcal{G}^*_{gr}$.

Every user maintains long-term secrets for initial contact with other users and a session state for each group in which she is member. The session state contains housekeeping variables and secrets for the exclusive usage in the group. Messages delivered in a group are not stored in the session state.

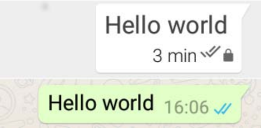
By distinguishing between *delivery* and *receiving* of messages, we want to emphasize that a received message is first processed by algorithms before the result is presented to the user.

2.2. Syntax

In order to provide a precise security model for secure group instant messaging, we define a group instant messaging protocol as the tuple of algorithms

$$\Sigma = ((\mathrm{snd}, \mathrm{rcv}),$$
$$(\mathrm{SndM}, \mathrm{Add}, \mathrm{Leave}, \mathrm{Rmv}, \mathrm{DelivM}, \mathrm{ModG}, \mathrm{Ack}))$$

The first two algorithms $(\mathrm{snd}, \mathrm{rcv})$ provide the application access to the network (network interface). Thereby snd outputs ciphertexts and rcv takes and processes ciphertexts. The latter seven algorithms process actions of the user or deliver remote actions of other users to the user's graphical interface (user interface)[1]. Each protocol specifies these algorithms and the interfaces among them. To denote that one algorithm algA has an interface to another algorithm algB we write $\mathrm{algA}^{\mathrm{algB}}$.



Hello world
3 min
Hello world 16:06

Figure 2. Double checkmarks in Signal (upper screenshot) and WhatsApp (lower screenshot) indicating that a group message was successfully delivered to *all* members' devices.

Every algorithm has modifying access to the session state of the calling user U for the communication in group gr. A schematic depiction of the syntax can be seen in Figure 1.

- $\mathrm{snd} \rightarrow \vec{c}$: Outputs a vector of ciphertexts, designated to the central server, to the network.
- $\mathrm{rcv}^{\mathrm{snd},\mathrm{DelivM},\mathrm{ModG},\mathrm{Ack}}(c)$: Receives ciphertext c from the central server and processes it by invoking one of the delivery algorithms and possibly the snd algorithm.

Actions of user U are processed by the following algorithms, which then invoke the snd algorithm for distributing the actions' results to the members $V_i \in \mathcal{G}_{gr}$ of group gr:

- $\mathrm{SndM}^{\mathrm{snd}}(gr, m) \rightarrow id$: Processes the sending of content message m to group gr.
- $\mathrm{Add}^{\mathrm{snd}}(gr, V) \rightarrow id$: Processes adding of user V to gr.
- $\mathrm{Leave}^{\mathrm{snd}}(gr) \rightarrow id$: Processes leaving of user U from gr.
- $\mathrm{Rmv}^{\mathrm{snd}}(gr, V) \rightarrow id$: Processes removal of user V from gr.

Every algorithm that processes the calling user's actions outputs a unique reference string id. In order to simplify the later defined security goals, we subsume the previous four algorithms as $\mathrm{Actn}(gr) \rightarrow id$.

Actions initiated by other users are first received as ciphertexts by the rcv algorithm and then passed to the following algorithms, which deliver the result to user U:

- $\mathrm{DelivM} \rightarrow (id, gr, V, m)$: Stores m with reference string id from sender V in group gr for displaying it to user U.
- $\mathrm{ModG} \rightarrow (id, gr')$: Updates the description of group gr with $ID_{gr} = ID_{gr'}$ to gr' after the remote modification with reference string id.
- $\mathrm{Ack} \rightarrow id$: Acknowledges that action with id was delivered and processed by all its designated receivers.

One practical implementation of the Ack algorithm for the acknowledgment of message delivery towards the sender is depicted in Figure 2: the first checkmark is set when the message is delivered to the central server, and the second checkmark is only set *if the message was delivered to all group members.*

For the same reason, for which we subsumed user actions under $\mathrm{Actn}(gr) \rightarrow id$, we denote both algorithms DelivM and ModG as $\mathrm{Deliv} \rightarrow (id, gr)$.

2.3. Threat Model

We consider three types of adversaries against secure instant messaging protocols. Thereby we define the ad-

[1] For clarity, our syntax disregards irrelevant features of instant messaging applications, such as the update of the group title.

versaries aiming to break one of the subsequently defined security goals in one designated group named the *target group*.

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.

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

Malicious Server. This adversary models attackers with access to the group instant messaging protocol alone. Motivated by our aim to analyze the reliance of the instant messaging protocols on the transport layer protection, we regard this attacker type. Besides the direct impersonation of the central server [3, 7], this adversary models attacks against the transport layer security between users and the central server [24, 25].

To analyze protocols' resilience against the compromise of user secrets, for the definition of *Perfect Forward Secrecy* and *Future Secrecy*, the following two capabilities are added to the previously described adversaries.

Long-term Secret Compromise. This enables the adversary to compromise a particular user during or after the protocol execution, to obtain her long-term secrets. As described for the malicious user, impersonations are considered as trivial attacks and therefore sessions, started after the compromise, are not considered as secure.

Session State Compromise. This enables the adversary to compromise a user to obtain the full session state at some intermediate stage of the protocol execution. In contrast to the *long-term secret compromise*, this additional capability is not restricted regarding the impersonation of members in the target group explicitly because the respective definitions of Perfect Forward Secrecy and Future Secrecy consider it accordingly.

2.4. Security Goals

Security and reliability in dynamic group communication can be divided into three aspects: 1) confidentiality of the conversations' content, 2) integrity of the conversation and 3) the confidentiality induced by the group management. Except from the last aspect, all defined goals are both applicable for group messaging and direct messaging. Indeed some of them are commonly ruled out because they seem to be reached trivially in a two party setting.

In addition to the subsequently defined security and reliability goals, there exist stronger definitions for the purpose of secure communication in groups [18, 26]. As we will argue in section 7, these definitions are not applicable for *instant* messaging in the described setting.

Confidentiality. We employ the one-way security notion for the confidentiality of the communicated content. As such, the definition of pure confidentiality is only applicable for non-compromising adversaries. We make then use of this definition for Perfect Forward Secrecy and Future Secrecy to regard compromising adversaries.

- ► *End-to-end Confidentiality.* No message m sent by a member $U \in \mathcal{G}_{gr}$ in the target group gr via $\mathrm{SndM}(gr, m)$ can be obtained by the adversary.

For defining confidentiality under session state compromise, we say *messages of* U *in* gr for messages that are sent by U by calling $\mathrm{SndM}(gr, m_s)$ and messages that are delivered to U via $\mathrm{DelivM} \to (id, gr, V, m_r)$.

- ► *Perfect Forward Secrecy.* On leakage of user U's session state, confidentiality of past *messages of* U *in* gr is maintained.

Future Secrecy – also known as *Post-Compromise Security* [2] – intuitively means that the protocol continuously renews the session state of a user's session and thereby invalidates old states in this session. Known protocols [2, 27, 28] reach this property by the interaction with the session partners.

Accordingly we define one *group round-trip* as the sequence of actions after which all members of a group output ciphertexts via snd and all group members received the ciphertexts designated to them via rcv.

- ► *Future Secrecy.* Let λ be a constant, then λ *group round-trips* after the leakage of user U's session state, confidentiality of future *messages of* U *in* gr is established.

The definition for groups can easily be applied to the two user setting. The proof of Cohn-Gordon et al. [2] shows that Signal provides Future Secrecy for $\lambda = 1$ for a static 'group' of size 2.

Integrity. Defining integrity as a goal for a consecutive communication not only targets end-to-end integrity of single messages, but the whole communicated content.

- ► *Message Authentication.* If a message m is delivered to $V \in \mathcal{G}_{gr}$ by $\mathrm{DelivM} \to (id, gr, U, m)$, then it was indeed sent by user U by calling $\mathrm{SndM}(gr, m)$.

While it is implied by *Message Authentication* that other users cannot plant messages into a communication between two parties, for groups it is necessary to be required[2]:

- ► *No Creation.* If a message m is delivered to member $U \in \mathcal{G}_{gr}$ via $\mathrm{DelivM} \to (id, gr, V, m)$ with sender V, then $V \in \mathcal{G}_{gr}$ holds.

Every member in a group can contribute content messages and it is therefore important for the receiver to know the exact sender. This differs for the initiation of group management actions (see *Confidentiality by Group Management*). Certainly, the following security goals are applicable to all actions a user initiates. For generality the definitions make use of the abstract algorithm names for actions of a user and for the respective delivery by its receivers[3].

[2] Please note that disregarding *No Creation* and *Closeness* as required goals provides a weak definition of *Reliable Multicast*.

[3] If $\mathrm{Actn}(gr) \to id$ refers to the SndM algorithm, then $\mathrm{Deliv} \to (gr, id)$ refers to the DelivM algorithm. All remaining actions by a user are delivered by the algorithm ModG.

- *No Duplication.* For every user action $\mathrm{Actn}(gr) \rightarrow id$ initiated by user U for group gr the resulting delivery algorithm $\mathrm{Deliv} \rightarrow (id, gr)$ is invoked at most once by each group member $V_i \in \mathcal{G}_{gr}$.
- *Traceable Delivery.* If the delivery of user action $\mathrm{Actn}(gr) \rightarrow id$ by user U is acknowledged to user $V' \in \mathcal{G}_{gr}$ by invoking $\mathrm{Ack} \rightarrow id$, then the respective $\mathrm{Deliv} \rightarrow (id, gr)$ algorithm was invoked by all members $V_i \in \mathcal{G}_{gr} \setminus \{U\}$.

Intuitively *Traceable Delivery* means that if a member is notified about the termination of an action performed in the group, then the respective delivery was invoked by all its members[4]. Please note that we do not restrict who obtains acknowledgments by the protocol. Commonly the initiator of an action is informed about the delivery state. Acknowledging the *leaving* of a user to this leaving user is, however, of little value.

We want to remark that Traceable Delivery provides no guarantees for the delivery of sent messages; it only provides guarantees regarding the validity of acknowledgments. A delivery guarantee can indeed not be provided since the centralized server can always refuse ciphertexts' forwarding.

- *Weak FIFO Order.* If user U calls $\mathrm{Actn}_1(gr) \rightarrow id_1$ before $\mathrm{Actn}_2(gr) \rightarrow id_2$, then member $V \in \mathcal{G}_{gr}$ will not invoke $\mathrm{Deliv}_1 \rightarrow (id_1, gr)$ after she invoked $\mathrm{Deliv}_2 \rightarrow (id_2, gr)$.
- *Weak Causal Order.* If user U_1 calls $\mathrm{Actn}_1(gr) \rightarrow id_1$ and user U_2 calls $\mathrm{Actn}_2(gr) \rightarrow id_2$ after she invoked $\mathrm{Deliv}_1 \rightarrow (id_1, gr)$, then member $V \in \mathcal{G}_{gr}$ will not invoke $\mathrm{Deliv}_1 \rightarrow (id_1, gr)$ after she invoked $\mathrm{Deliv}_2 \rightarrow (id_2, gr)$.

In contrast to all other security and reliability goals, ordering – in its strict definition – limits the instant delivery of messages: a later message would only be delivered if all its predecessors were delivered. For this reason we relax the definition such that the instant delivery is possible under the restriction that message omissions are accepted as long as the order among delivered messages is preserved.

Confidentiality by Group Management. Group protocols must fulfill additional requirements to meet *confidentiality* as a security goal. While authenticity for content messages is defined via *Message Authentication*, the following definitions also imply authenticity for group management actions:

- *Additive Closeness.* If member $U \in \mathcal{G}_{gr}$ modifies the member set $\mathcal{G}_{gr}^{old}$ to $\mathcal{G}_{gr'}$ via $(id, gr') \leftarrow \mathrm{ModG} : |\mathcal{G}_{gr}^{old}| < |\mathcal{G}_{gr'}|, ID_{gr} = ID_{gr'}$, then an administrator $U^* \in \mathcal{G}_{gr}^*$ called $\mathrm{Add}(gr, V)$ to add new member $V = \mathcal{G}_{gr'} \setminus \mathcal{G}_{gr}^{old}$ to the group.
- *Subtractive Closeness.* If member $U \in \mathcal{G}_{gr}$ modifies the member set $\mathcal{G}_{gr}^{old}$ to $\mathcal{G}_{gr'}$ via $(id, gr') \leftarrow \mathrm{ModG} : |\mathcal{G}_{gr}^{old}| > |\mathcal{G}_{gr'}|, ID_{gr} = ID_{gr'}$, then either an administrator $U^* \in \mathcal{G}_{gr}^*$ called $\mathrm{Rmv}(gr, V)$ to remove member $V = \mathcal{G}_{gr}^{old} \setminus \mathcal{G}_{gr'}$ from the group, or member V called $\mathrm{Leave}(gr)$ to leave the group.

While *Additive Closeness* is a security goal, *Subtractive Closeness* is defined as a correctness property and thereby targets reliability. Turning the latter into a security definition, requiring the enforcement of member set reduction, is of no value in a centralized server structure where the server can drop every ciphertext. Since *Traceable Delivery* is applied to all user actions, a certain security assertion can be made: if a member invokes $\mathrm{Ack} \rightarrow id$ for the removal operation with the same id, then the removal was conducted by all remaining members.

Secure and Reliable Group Instant Messaging.

Definition 1. A protocol Σ is a *Secure and Reliable Group Instant Messaging Protocol* if it fulfills *End-to-end Confidentiality*, *Message Authentication*, *No Creation*, *No Duplication*, *Traceable Delivery*, *Additive Closeness*, and *Subtractive Closeness* in the presence of *Malicious User*, *Network Attacker*, and *Malicious Server*.

Furthermore the protocol may reach *Perfect Forward Secrecy* and *Future Secrecy* to defend compromising adversaries.

Substantiating *reliability* of the protocol is reached if it provides *Weak FIFO Order* and *Weak Causal Order*. Our ordering definitions, however, illustrate that there exists a tradeoff between reliability and instant message delivery. As such we address ordering as a soft goal for secure and reliable instant messaging. An additional discussion regarding this tradeoff can be found in section 7.

3. Methodology

We describe our general evaluation methodology in the following.

Test Setup. For all three investigated protocols, 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 [29, 30]. 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 [15, 16]. The respective messages and operations were sent by the official applications running on different devices and transmitted via the official messenger servers.

Proof-of-Concepts. In order to substantiate the described protocol shortcomings, we were able to implement proof-of-concept exploitations for a subset of them. Attacks involving a malicious server could only partially be exploited since we did not have access to the official servers. The protocol

[4] It may seem that this property is implied by *Message Authentication* and *No Creation*. It would therefore be necessary to implement Ack by explicit acknowledgment messages that are processed as content messages. In order to keep our model generic, we define *Traceable Delivery* independent of assumptions on the implementation.

descriptions however strongly suggest our evaluation results. Details are given in sections 4.3, 5.3, and 6.3.

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

Constraints of Attack Descriptions. Even though the developers do not explicitly claim to satisfy our definition of security, we will call discrepancies between the security provided by the protocols and security required by our definition *attacks* since our model requires its fulfillment.

Description of an Example Protocol Run. To support the comprehension of the analyzed protocols, our extended version [31] includes a visual and written description of an example protocol sequence for each messaging protocol. This sequence depicts the evolution and distribution of employed key material as well as the distribution of the ciphertexts for normal communication and group operations such as leaving and creating a group.

4. Signal

Signal is an open source instant messaging application available for Android, iOS, and as a Google Chrome extension [32]. 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 [1, 2].

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.

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.

We assume the cryptographic primitives' implementations secure and thereby treat them as black boxes in our analysis.

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 model.

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* [33] that uses static and ephemeral Diffie-Hellman shares of both parties. This root key initializes the *Double Ratchet algorithm (DR algorithm)* [34], which can be seen as a stateful encryption algorithm. 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 contributes messages. 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*. By its characteristics, the symmetric ratcheting cannot provide *Future Secrecy* but provides *Perfect Forward Secrecy* of the resulting keys. The asymmetric ratcheting provides both properties such that the combination (DR algorithm) also 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 3 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.[5]

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

4.2. Group Protocol

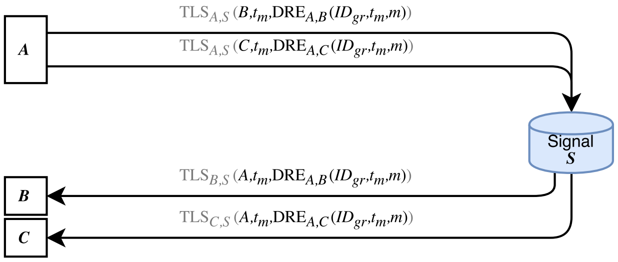

$\mathrm{TLS}_{A,S}\,(B,t_m,\mathrm{DRE}_{A,B}(ID_{gr},t_m,m))$
$\mathrm{TLS}_{A,S}\,(C,t_m,\mathrm{DRE}_{A,C}(ID_{gr},t_m,m))$
A
Signal
S
$\mathrm{TLS}_{B,S}\,(A,t_m,\mathrm{DRE}_{A,B}(ID_{gr},t_m,m))$
B
$\mathrm{TLS}_{C,S}\,(A,t_m,\mathrm{DRE}_{A,C}(ID_{gr},t_m,m))$
C

Figure 3. 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 secure group messaging protocols (e.g., WhatsApp and Threema), Signal implements non-administered groups such that all members of a group

[5] A more detailed description of the DR algorithm when used in the Signal and WhatsApp messaging protocol is provided in our extended version [31].

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 3 describes the format of a group message from member A to members B and C in group gr that is sent via the server $\mathcal{S}$[6].

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

$$m := \begin{cases} m, & \textit{if } \mathrm{SndM}(gr, m) \\ (\mathcal{G}_{gr}, \textit{info}_{gr}), & \textit{if } \mathrm{Add}(gr, V) : \mathcal{G}_{gr} := \mathcal{G}^{old}_{gr} \cup \{V\} \\ \texttt{leave}, & \textit{if } \mathrm{Leave}(gr) \end{cases}$$

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 2) as soon as all receivers' acknowledgments arrived.

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 $\textit{info}_{gr}$. 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.

4.3. Security Evaluation and Observations

We practically verified two weaknesses of Signal and created proof-of-concepts for them.

4.3.1. Burgle into the Group. Performing the following steps 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.
- *Session State Compromise.* ID_{gr} is stored in the session state and can thereby be revealed via this compromise.

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

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

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:

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

B will use this set $\mathcal{G}^{new}_{gr}$ 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 $\textit{info}_{gr}$), she will send some message

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

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

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 Additive Closeness is broken for a larger set of members, and it is more likely that one of these members sends the second update message.

Impact. The protocol does not provide the following security goals:

- *No Creation.* A group member B accepts a content message by $\mathcal{A}$, who is not part of the group.

[6] We omitted irrelevant fields regarding our evaluation in the message format. The whole format can be found in the format description [35]. We also left out Google's Cloud Messaging (GCM) service for clarity.

- *Additive Closeness.* By sending an *update* message, $\mathcal{A}$ can add herself to the group which breaks Additive Closeness.
- *Future Secrecy.* After adding herself to the group, the confidentiality of future plaintext messages is compromised.

4.3.2. Forging Acknowledgments. Signal provides information on the receipt status of messages for the sender in groups and for direct messaging (see Figure 2). 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.

- *Malicious Server.* 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.

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.

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 2) 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 [7]. As a result *Traceable Delivery* is neither provided for group messages nor for direct messages by Signal.

4.3.3. Ordering. The *Malicious Server* cannot only drop messages, but also reorder them. The receiving application orders simultaneously received messages by the timestamp which is manipulable for the server. Henceforth neither *FIFO Order* nor *Causal Order* are provided by Signal.

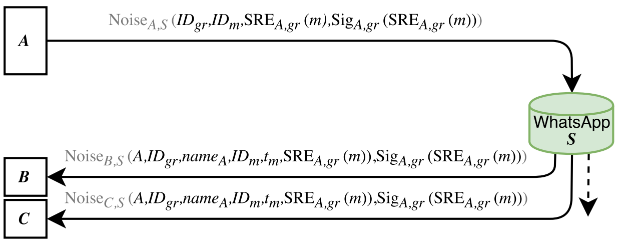

A
$\mathrm{Noise}_{A,S}$ $(ID_{gr}, ID_m, \mathrm{SRE}_{A,gr}(m), \mathrm{Sig}_{A,gr}(\mathrm{SRE}_{A,gr}(m)))$
WhatsApp
S
B
$\mathrm{Noise}_{B,S}$ $(A, ID_{gr}, name_A, ID_m, t_m, \mathrm{SRE}_{A,gr}(m)), \mathrm{Sig}_{A,gr}(\mathrm{SRE}_{A,gr}(m)))$
C
$\mathrm{Noise}_{C,S}$ $(A, ID_{gr}, name_A, ID_m,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

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. 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 operating systems[8]. Our analysis confirms the description of WhatsApp's technical white paper [36] 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 shortcomings.

5.1. General Initialization Protocol

5.1.1. Session Establishment with the Server. WhatsApp uses *Noise Pipes* [37] to protect the communication between the clients and the server on the transport layer [36].

5.1.2. Key Agreement and Key Derivation. The Signal key exchange protocol, consisting of the *X3DH Key Agreement Protocol* [33] and the *DR algorithm* [34], is integrated in WhatsApp in order to establish a confidential channel for messaging between two users [36]. A detailed description of these building blocks can be found in section 4.

5.2. Group Protocol

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.

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

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

[8] https://www.whatsapp.com/download/

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 on 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 2). 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.

The WhatsApp application enables users, for sending a message, to highlight a reference to a previous message. The protocol therefore attaches the whole referenced message and its ID ID_m to the newly sent message, such that the referenced message, the new message, and the ID of the referenced message are encrypted.

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 a message of this format to the server, the information is distributed to all group members:

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

The session state 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. 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.

5.3. Security Evaluation and Observations

We observed two shortcomings in the design of WhatsApp's group protocol that allow to (1) burgle into a group and to (2) forge acknowledgments. The shortcomings have similar results as the attacks on Signal, although the underlying protocol and exploitation differ.

5.3.1. Burgle into a Group. The subsequently described protocol design weakness 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.

► *Malicious Server.* can send group modification messages to the group members.

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 Additive Closeness in the group by conducting the following steps. The attacker sends the following group modification message to users C, D[9]:

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

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.

Optimizations. The attack can be optimized by also adding $\mathcal{A}$ to B's view of the group. 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.

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

Impact. Due to the described attack, the protocol does not reach:

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

5.3.2. Forging Acknowledgments. 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.

- *Malicious Server.* can manipulate the transcript between sender and receivers.

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.

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 in WhatsApp.

5.3.4. Ordering. The sending time for a message is set at the server side. 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* can disrupt the order of messages. By employing a reference to a previous message, *Causal Order* is at least preserved for this reference.

6. Threema

Threema is a proprietary closed source instant messenger protocol available for most mobile operating systems [11]. 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.

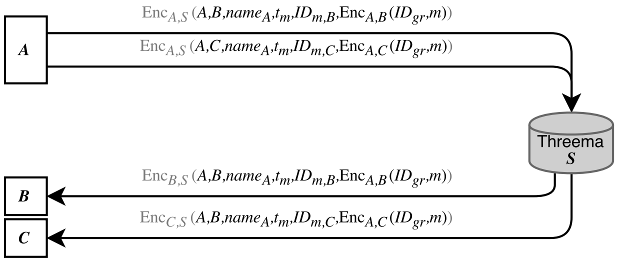

$\mathrm{Enc}_{A,S}\,(A,B,name_A,t_m,ID_{m,B},\mathrm{Enc}_{A,B}(ID_{gr},m))$
$\mathrm{Enc}_{A,S}\,(A,C,name_A,t_m,ID_{m,C},\mathrm{Enc}_{A,C}(ID_{gr},m))$
A
Threema
S
$\mathrm{Enc}_{B,S}\,(A,B,name_A,t_m,ID_{m,B},\mathrm{Enc}_{A,B}(ID_{gr},m))$
B
$\mathrm{Enc}_{C,S}\,(A,B,name_A,t_m,ID_{m,C},\mathrm{Enc}_{A,C}(ID_{gr},m))$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

Figure 5. 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).

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 $k_{A,B} = \mathrm{ECDH}(sk_A^{lt}, pk_B^{lt}) = \mathrm{ECDH}(sk_B^{lt}, pk_A^{lt})$ from the DHKE of the long term key shares. This key is used for all direct and group messages between these two users.

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. 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 5 where $ID_{m,U}$ is a random message identifier for the respective receiver, t_m is a timestamp 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:

$$m := \begin{cases} m, & \textit{if } \mathrm{SndM}(gr, m) \\ \mathcal{G}_{gr}, & \textit{if } \text{update message 1} \\ \textit{info}_{gr}, & \textit{if } \text{update message 2} \\ \texttt{leave}, & \textit{if } \mathrm{Leave}(gr) \end{cases}$$

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).

Like WhatsApp, Threema provides the users the ability to explicitly refer to a previous message when writing a new one. Thereby also the whole referenced message is attached to the new message (and then encrypted together).

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 all $U \in \mathcal{G}_{gr} \setminus \{U^*_{gr}\}$: (1) a message containing the new set $\mathcal{G}_{gr}$ and (2) a message containing the updated $\textit{info}_{gr}$ 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.

6.3. 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 Additive 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 access to the channel somewhere between sender and receiver.

► *Malicious Server.* has control over the transmitted ciphertexts.

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.

Impact. The attack violates the following security goals:

► *No Duplication.* $\mathcal{A}$ can replay messages.

► *Additive Closeness.* This weakness also affects the Additive 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. 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 *Malicious Server* can reorder messages arbitrarily during the transmission. By referencing previous messages, at least for this reference *Causal Order* is preserved.

6.3.5. 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. [10]

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. Additive Closeness and No Creation. In Signal, Additive 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 Additive Closeness. As a side effect, the group ID can then be public knowledge. We discussed this proposal with Open Whisper-Systems, but it turned out that this verification is unfeasible due to their current implementation. Open WhisperSystems is currently developing a new group management system with advanced administrative features so that they decided not to apply our fix.

Traceable Delivery. Signal could reach Traceable Delivery by treating receipt messages like content messages and thus end-to-end encrypt them[11]. 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. Additive 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 a malicious server, 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[12].

7.2. General Outcomes

7.2.1. Reaching Traceable Delivery in General. The results of our analysis may imply that Traceable Delivery in instant messaging 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.

Using the approach of negative acknowledgments (NACKs) turns the responsibility of the Traceable Delivery from the sender to the receiver [38, 39, 40]. 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. As part of this, the receiver can refer to the last in-order delivered message within her normal content messages such that the initial sender can mark them as delivered (which also reduces communication complexity). 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.

7.2.2. Securely Managing a Group. In order to reach Additive Closeness and No Creation in groups, members of a group need to distinguish between group members and external users.

We see two natural 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 is 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

[10] This weakness was also fixed in Threema version 3.14.

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

[12] https://threema.ch/en/versionhistory

Delivery must be achieved because otherwise the server provider can undetectably 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 [41]. 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 messaging protocols.

Membership Proof. When solely using a group secret that protects Additive 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.

The underlying problem is related to future secure group key exchange. A first group key exchange with this property was recently proposed by Cohn-Gordon et al. [28].

7.2.3. Preserving Order. While FIFO Order can easily be established by enforcing the properties of stateful encryption, it inevitably restricts the *instant* delivery of messages because if a later message is received earlier, it needs to be cached until the all previous messages were delivered. According to our (weaker) definition, messages can be delivered instantly, but then older messages, that were not received in the correct order, would need to be dropped. As Marson and Poettering [42] provide a causal broadcast algorithm employing authenticated encryption with associated data and vector clocks, Causal Order can also be achieved with standard cryptographic primitives under the same tradeoff between reliability and instant delivery.

The analyzed applications already employ visual features to provide information on the order of messages in the user interface. However, as our descriptions of the shortcomings reveal, these features are not appropriately protected towards the Malicious Server. Since the order of messages – especially the causal context – is very important for the sense of their content, and instant delivery of messages is the inevitable characteristic of *instant messaging*, it is up to the developers how far reliability is reached with respect to order preservation.

8. Related Work

Analyses of IM Applications. Unger et al. [43] systematize current secure instant messengers 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. [1]. 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. [3] 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. [2] 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 [44]. Bellare et al. [27] investigate ratcheting as a cryptographic primitive. Their work does not specifically focuses 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. [45] analyzed WhatsApp among other IM applications regarding the authentication and account management and found several vulnerabilities. Another application specific analysis [46] 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 [47, 48]. In addition to these analyses, the WhatsApp protocol was implemented and published as open source projects [16, 49].

Analyses of Threema. An initial analysis of the Threema protocol was conducted by Ahrens [50]. Based on this Berger [15] 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 [51].

Security in Multi User Settings.

Cohn-Gordon et al. [28] 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 provide a proof for parts of their construction.

Bracha and Toueg introduced the notion of *reliable broadcast* in the asynchronous setting [19]. Since then many works introduced and improved algorithms to solve the problem of validly and consistently delivering messages in a multi user setting (e.g., [52, 53]) but also refined the notion and definition to provide realistic attacker models [53].

Chockler et al. [20] give an overview on various models and results regarding group communication systems (GCS). They systematize different notions regarding the reliability and security of group communication in the literature.

Marson and Poettering [42] recently defined a security model that captures confidentiality and integrity in a multi user setting, and provided provably secure constructions. Their work, however, does not cover dynamic groups. Furthermore, as discussed in section 7, their model requires stronger notions of reliability at costs of the *instant* message delivery.

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 applications has been in the focus of recent analyses [1, 2, 3], the investigation of end-to-end protected group communications has gained only little attention.

We fill this gap by providing a security model and a methodology for analyzing group instant messaging protocols. We demonstrate their applicability by conducting a systematical analysis of three major secure group instant messengers: Signal, WhatsApp, and Threema.

While our investigation focuses on three major instant messaging applications, our methodology and the underlying model is of generic purpose and can be applied to other secure group instant messaging protocols as well. For example, it would be interesting to analyze the group chat implementations of other Signal-based messaging protocols, such as Google's Allo, Wire, 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 [28]. 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 a secure and reliable group messaging 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 instant messaging protocols.

Acknowledgments. We thank Tibor Jager, Julian Loss, Moxie Marlinspike, the Threema Security Team, the CCS 2017, and EuroS&P 2018 reviewers for their helpful comments and fruitful discussions.

References

[1] T. Frosch, C. Mainka, C. Bader, F. Bergsma, J. Schwenk, and T. Holz, "How secure is textsecure?" in *EuroS&P*, 2016.

[2] K. Cohn-Gordon, C. Cremers, B. Dowling, L. Garratt, and D. Stebila, "A formal security analysis of the Signal messaging protocol," in *EuroS&P*, 2017.

[3] N. Kobeissi, K. Bhargavan, and B. Blanchet, "Automated verification for secure messaging protocols and their implementations: A symbolic and computational approach," in *EuroS&P*, 2017.

[4] (2017) Communications market report. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0017/105074/cmr-2017-uk.pdf

[5] eMarketer. (2015, Nov.) Mobile messaging to reach 1.4 billion worldwide in 2015. [Online]. Available: https://www.emarketer.com/Article/Mobile-Messaging-Reach-14-Billion-Worldwide-2015/1013215

[6] Business2Community. (2015, Aug.) Are instant messaging apps the future of the (mobile) internet? [Online]. Available: http://www.business2community.com/mobile-apps/instant-messaging-apps-future-mobile-internet-01313577

[7] C. Garman, M. Green, G. Kaptchuk, I. Miers, and M. Rushanan, "Dancing on the lip of the volcano: Chosen ciphertext attacks on apple imessage," in *USENIX Security Symposium*, 2016.

[8] Moxie Marlinspike, "Advanced cryptographic ratcheting," 2013. [Online]. Available: https://whispersystems.org/blog/advanced-ratcheting/

[9] Open Whisper Systems, "Signal website," 2017. [Online]. Available: https://signal.org/

[10] WhatsApp, "Whatsapp security," 2017. [Online]. Available: https://www.whatsapp.com/security/

[11] Threema, "Threema website," 2017. [Online]. Available: https://threema.ch/en

[12] Open Whisper Systems, "Open whisper systems partners with google on end-to-end encryption for allo," 2017. [Online]. Available: https://whispersystems.org/blog/allo/

[13] ——, "Facebook messenger deploys signal protocol for end to end encryption," 2017. [Online]. Available: https://whispersystems.org/blog/facebook-messenger/

[14] S. Dechand, D. Schürmann, K. Busse, Y. Acar, S. Fahl, and M. Smith, "An empirical study of textual key-fingerprint representations," in *USENIX Security Symposium*, 2016.

[15] P. Berger, "Open source implementation of a threema desktop client," 2016. [Online]. Available: https://github.com/blizzard4591/openMittsu

[16] T. Galal, "Open source implementation of a whatsapp client," 2016. [Online]. Available: https://github.com/tgalal/yowsup

[17] WhatsApp, "Why am i banned for using whatsapp plus and how do i get unbanned?" 2016. [Online]. Available: https://www.whatsapp.com/faq/en/general/105

[18] V. Hadzilacos and S. Toueg, "Distributed systems (2nd ed.)," S. Mullender, Ed., 1993, ch. Fault-tolerant Broadcasts and Related Problems.

[19] G. Bracha and S. Toueg, "Asynchronous consensus and broadcast protocols," *J. ACM*, 1985.

[20] G. V. Chockler, I. Keidar, and R. Vitenberg, "Group communication specifications: a comprehensive study," *ACM Comput. Surv.*, 2001.

[21] Statista, "Most popular messaging apps," 2017. [Online]. Available: https://www.statista.com/statistics/

258749/most-popular-global-mobile-messenger-apps/

[22] Signal, “Signal private messenger in google play,” 2017. [Online]. Available: https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms

[23] Threema, “Threema in google play,” 2017. [Online]. Available: https://play.google.com/store/apps/details?id=ch.threema.app

[24] C. Meyer, J. Somorovsky, E. Weiss, J. Schwenk, S. Schinzel, and E. Tews, “Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks,” in *USENIX Security Symposium*, 2014.

[25] M. R. Albrecht and K. G. Paterson, “Lucky microseconds: A timing attack on amazon’s s2n implementation of TLS,” in *EUROCRYPT*, 2016.

[26] S. Duan, L. Nicely, and H. Zhang, “Byzantine reliable broadcast in sparse networks,” in *15th IEEE International Symposium on Network Computing and Applications, NCA*, 2016.

[27] M. Bellare, A. C. Singh, J. Jaeger, M. Nyayapati, and I. Stepanovs, “Ratcheted encryption and key exchange: The security of messaging,” in *CRYPTO*, 2017.

[28] K. Cohn-Gordon, C. Cremers, L. Garratt, J. Millican, and K. Milner, “On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees,” *IACR Cryptology ePrint Archive*, 2017. [Online]. Available: http://eprint.iacr.org/2017/666

[29] Open Whisper Systems, “Source code of signal-android,” 11 2016, android Application Version 3.23.0. [Online]. Available: https://github.com/WhisperSystems/Signal-Android/commit/ce812ed8ba49fc43db9de018c135be67b5b44f7d

[30] ——, “Source code of signal-service library,” 11 2016, java Library Version 2.4.1. [Online]. Available: https://github.com/WhisperSystems/libsignal-service-java/commit/460cd7559caa74bb6539c72865c71de660a69bac

[31] (2017) More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema (extended version). Cryptology ePrint Archive, Report 2017/786. https://eprint.iacr.org/2017/713.

[32] Open Whisper Systems, “Signal github repository,” 05 2017. [Online]. Available: https://github.com/WhisperSystems/

[33] M. Marlinspike and T. Perrin, “The x3dh key agreement protocol,” 11 2016. [Online]. Available: https://whispersystems.org/docs/specifications/x3dh/x3dh.pdf

[34] ——, “The double ratchet algorithm,” 11 2016. [Online]. Available: https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf

[35] Open Whisper Systems, “Message format in the signal protocol,” 11 2016, specified with Google Protocol Buffers. [Online]. Available: https://github.com/WhisperSystems/libsignal-service-java/blob/4cedb5c31c11c1e8811b3bb7cd68d56ff7e0c03f/protobuf/SignalService.proto

[36] WhatsApp Inc., “Whatsapp encryption overview,” 2016, technical white paper. [Online]. Available: https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf

[37] T. Perrin, “The noise protocol framework,” 2016. [Online]. Available: http://noiseprotocol.org/noise.pdf

[38] C. Diot, W. Dabbous, and J. Crowcroft, “Multipoint communication: A survey of protocols, functions, and mechanisms,” *IEEE J. SAC*, 1997.

[39] B. N. Levine and J. J. Garcia-Luna-Aceves, “A comparison of reliable multicast protocols,” *Multimedia Syst.*, 1998.

[40] B. Adamson, C. Bormann, M. Handley, and J. Macker, “Multicast Negative-Acknowledgment (NACK) Building Blocks,” Internet Requests for Comments, IETF, RFC 5401, November 2008. [Online]. Available: https://tools.ietf.org/html/rfc5401

[41] A. Schiper and S. Toueg, “From set membership to group membership: A separation of concerns,” *IEEE Trans. Dependable Sec. Comput.*, 2006.

[42] G. A. Marson and B. Poettering, “With one it is easy, with many it gets complicated: Understanding channel security for groups,” Cryptology ePrint Archive, Report 2017/786, 2017, https://eprint.iacr.org/2017/786.

[43] N. Unger, S. Dechand, J. Bonneau, S. Fahl, H. Perl, I. Goldberg, and M. Smith, “Sok: Secure messaging,” in *S&P*, 2015.

[44] K. Cohn-Gordon, C. J. F. Cremers, and L. Garratt, “On post-compromise security,” in *IEEE CSF*, 2016.

[45] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl, “Guess who’s texting you? evaluating the security of smartphone messaging applications,” in *NDSS*, 2012.

[46] C. Anglano, “Forensic analysis of whatsapp messenger on android smartphones,” *Digital Investigation*, 2014.

[47] T. B. Manisha Ganguly, “Whatsapp vulnerability allows snooping on encrypted messages,” *The Guardian*, 2017. [Online]. Available: https://www.gu.com/technology/2017/jan/13/whatsapp-backdoor-allows-snooping-on-encrypted-messages

[48] Moxie Marlinspike, “There is no whatsapp ’backdoor’,” 2017. [Online]. Available: https://whispersystems.org/blog/there-is-no-whatsapp-backdoor/

[49] mgp25, “Open source implementation of a whatsapp php api,” 2016. [Online]. Available: https://github.com/mgp25/Chat-API

[50] J. Ahrens, “Threema protocol analysis,” 2014. [Online]. Available: http://blog.jan-ahrens.eu/files/threema-protocol-analysis.pdf

[51] R. Schilling and F. Steinmetz, “A look into the mobile messaging black box,” 2016, at 33c3. Implementation: https://github.com/o3ma. [Online]. Available: https://media.ccc.de/v/33c3-8062-a_look_into_the_mobile_messaging_black_box

[52] R. Canetti and T. Rabin, “Fast asynchronous byzantine agreement with optimal resilience,” in *ACM STOC*, 1993.

[53] C. Cachin, K. Kursawe, F. Petzold, and V. Shoup, “Secure and efficient asynchronous broadcast protocols,” in *CRYPTO*, 2001.