RUHR-UNIVERSITÄT BOCHUM

RUB

RUHR-UNIVERSITÄT BOCHUM

On the End-to-End Security of Group Chats in Instant Messaging Protocols

Paul Rösler

Master's Thesis – December 6, 2018.
Chair for Network and Data Security.

1st Supervisor: Prof. Dr. Jörg Schwenk
2nd Supervisor: Prof. Dr. Tibor Jager
1st Advisor: Prof. Dr. Jörg Schwenk
2nd Advisor: Dr. Christian Mainka

hgi Lehrstuhl für Netz- und Datensicherheit

Acknowledgments

The reason for finishing my master thesis more than half a year later than it was planned, is my enthusiasms for doing (new) research. I am grateful to the ones who supported me in gaining the opportunity to do research. At first I want to thank colleges Martin and Christian, who introduced me to my supervisor Jörg. Then I want to thank Jörg for providing a liberal environment of self-responsibility that allows me to explore many interesting areas of cryptography and IT security. I am also thankful to Christian and Jörg for starting my PhD studies with a collaboration that resulted in this thesis. Furthermore I want to thank my second supervisor Eike for introducing me to (other researchers but especially to) Bertram from whom I learned to be precise and formal in my work. Also I want to thank all my colleges (especially my office neighbor Robert) for jointly enabling a friendly and welcoming office atmosphere. Then I thank the best learning group for having wonderful six years of studies (and much more). I honestly hope that we maintain our friendship. Finally I thank my whole family and my girlfriend Nadine for always supporting me mentally and – if necessary – letting me work long hours.

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.

In this thesis, we investigate group communication security mechanisms of three major massaging applications: Signal, WhatsApp, and Threema. We first introduce the scientific background of theoretical and practical approaches to define and analyze secure group instant messaging. Then this thesis mainly consists of the results, the author (together with Mainka and Schwenk) published as the article "*More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema*" in the proceedings of *3rd IEEE European Symposium on Security and Privacy (Euro S&P 2018)*.

To approach an investigation of group instant messaging protocols in this publication, we provide a comprehensive and realistic attacker model. This model combines security and reliability goals from various related literature to capture relevant properties for communication in dynamic groups. Thereby we also consider the satisfiability of the definitions with respect to the instant delivery of messages. Since the analyzed protocols and their implementations are mostly undocumented for the public and two out of three applications among them are closed source, we are the first to describe the group protocols employed in Signal, WhatsApp, and Threema. By applying our model on these protocols, we reveal several shortcomings with respect to our 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 group communication.

Official Declaration

Hereby I declare, that I have not submitted this thesis in this or similar form to any other examination at the Ruhr-Universität Bochum or any other Institution of High School.

I officially ensure, that this paper has been written solely on my own. I herewith officially ensure, that I have not used any other sources but those stated by me. Any and every parts of the text which constitute quotes in original wording or in its essence have been explicitly referred by me by using official marking and proper quotation. This is also valid for used drafts, pictures and similar formats.

I also officially ensure, that the printed version as submitted by me fully confirms with my digital version. I agree that the digital version will be used to subject the paper to plagiarism examination.

Not this English translation, but only the official version in German is legally binding.

Eidesstattliche Erklärung

Ich erkläre, dass ich keine Arbeit in gleicher oder ähnlicher Fassung bereits für eine andere Prüfung an der Ruhr-Universität Bochum oder einer anderen Hochschule eingereicht habe.

Ich versichere, dass ich diese Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen benutzt habe. Die Stellen, die anderen Quellen dem Wortlaut oder dem Sinn nach entnommen sind, habe ich unter Angabe der Quellen kenntlich gemacht. Dies gilt sinngemäß auch für verwendete Zeichnungen, Skizzen, bildliche Darstellungen und dergleichen.

Ich versichere auch, dass die von mir eingereichte schriftliche Version mit der digitalen Version übereinstimmt. Ich erkläre mich damit einverstanden, dass die digitale Version dieser Arbeit zwecks Plagiatsprüfung verwendet wird.

Date	author

Contents

1 Introduction

Instant messaging has emerged to be one of the most relevant means of communication besides E-Mail, phone calls, and personal communication. Among young users, it is already the most popular communication medium [77]. Especially from the security perspective, instant messaging has a special role in comparison to other means of communication. One major difference is that, in contrast to E-Mail, phone calls, or short messages service (SMS), instant messaging protocols are stateful. That means, a connection between communicating partners stays alive such that it can be used continuously. It is important to emphasize here that a stateful connection does not require all communication partners to participate in the communication simultaneously (that means communication can be conducted asynchronously). Due to statefulness, one can achieve (and hence may require) stronger security guarantees from instant messaging in comparison to stateless protocols. Nevertheless, our results show that still not all secure instant messaging protocols take advantage of the maintained state.

In addition to this property, compared to other stateful protocols (such as short-term chat protocols like IRC [61]), instant messaging sessions (i.e., connections) have a long life time: A session among two parties is usually setup with the first message between the parties and is afterwards only terminated and freshly setup as soon as one of the parties changes the device. Since the respective devices (e.g, smartphones) are changed in time periods of multiple years (for smartphones this is between six months and two years for 60% of US citizens [78]), sessions in instant messaging are faced with a stronger threat model than other named communication protocols. One of the main contributions of this thesis is the design of a model that aims to depict the respectively stronger capabilities of adversaries in instant messaging.

While there has been related work that already provided important steps in modeling and analyzing security as well as designing secure protocols for instant messaging, group communication in instant messaging has, until now, received relatively little attention in the literature. Despite the lack of scientific work regarding the security of group instant messaging protocols, practical protocols for communication in groups are widely implemented in instant messaging applications. Even though, these protocols are equipped with mechanisms to protect the confidentiality and authenticity of the communication, it is not clear against which attackers these security goals are defended (and the security goals themselves are not precisely defined, either).

This thesis, as a result, makes steps into two distinct directions with respect to the security of group instant messaging:

1. A model for the definition of security in group instant messaging is developed. Therefore, first common goals of secure channels (such as confidentiality, integrity, and authenticity) are regarded. Furthermore, since dynamic groups (in which members can be added and removed to or from groups) are considered, security properties that regard the modification of the group member set are regarded. Finally, special features, common to instant messaging (such as receipt acknowledgments), are considered in the definition of security properties.

   Additionally we define the capabilities of attackers that aim to break these security properties. This includes both, the access to the network (e.g., manipulation of ciphertexts) and the ability to obtain secret information from the attacked users (like a session's local state information).

2. Widely distributed group instant messaging protocols are described in this thesis. The description required reverse engineering of the respective applications as for none of their protocols a detailed documentation nor sufficient description existed before. On the one hand, this enables us to analyze these protocols with respect to out attacker model. We thereby find security insufficiencies and propose countermeasures to strengthen the applications' protocols. On the other hand, our description may support further analyses of these protocols.

The thesis then continues by analyzing the protocols with respect to the model. At the end (since the analysis reveals shortcomings of the analyzed protocols' security), countermeasures for protecting the protocols are recommended.

To reduce redundancy between the introducing chapters of this thesis and the *Introduction*, *Approach*, *Attacker Model*, and *Related Work* sections of the paper that is embedded as the main body of the thesis, this first part of the thesis sheds a broader light on the approach of formally modeling instant messaging. For gaining an overview of the topic of instant messaging in general, the reader is referred to the *Introduction* section of the paper (in the main body).

1.1 Motivation

As described before, instant messaging is one of the most important means of communication in the (digital) world but, despite this importance, has received very little attention in the literature. As a side effect, instant messaging providers built their own protocols for group communication. The approach for their design thereby bases on the combination of known and secure building blocks. Combining these building blocks is, however, non-trivial and, since no precise attacker model is defined, it is unclear how far this combination provides *security*.

While for pairwise instant messaging communication, the de-facto standard is Signal's protocol (based on the *X3DH key exchange* [54] and the *Double Ratchet algorithm* [53]), no such widely deployed protocol exists for group instant messaging. The Signal protocol was designed by Moxie Marlinspike and Trevor Perrin without formal analysis nor precise security definition. Cohn-Gordon et al. [25] later formally analyzed the security of it. One can see that, in comparison to related work, the security model of their formal analysis is very extensive and complex, which results from both, the strong adversary capabilities, but especially from the fact that the model precisely grasps the security properties that Signal provides. Very recently, Poettering and Rösler [67, 68] proposed a *natural* security model for pairwise *ratcheting*, which is the abstract primitive behind the *Double Ratchet algorithm*. One can compare their approach to the first *natural* security definitions of encryption [10, 41]: both provide the adversary full control over the respective set of algorithms (for ratcheting this is sending and receiving and for ciphers this is encryption and decryption) and they require *full security* except for these cases in which it is certain that an adversary can win without breaking any hard assumptions, which are called *trivial attacks* (letting the adversary decrypt the challenge ciphertext is a trivial attack in the CCA game for ciphers). The model by Cohn-Gordon et al. [25] does not base on trivial attacks only, because the analyzed protocol (i.e., Signal) was not designed with such a strong security model in mind, but rather with an abstract idea of security.

When designing secure protocols for instant messaging in dynamic groups, more aspects than only the establishment of a shared secret, and based on that, the establishment of a channel need to be considered. Combining primitives to obtain a secure protocol is consequently more complex. As our analysis reveals, practical protocols either employ trivial compositions of the pairwise protocols to obtain group protocols, or develop group management mechanisms that undermine the group channel's security properties.

Our analysis shows that there exists a need for research in this area: practically used protocols for group instant messaging are designed based on best practices. Our definition of security thereby is a first step that captures practically relevant properties. Based on this, the design of future protocols can benefit by considering a comprehensible definition of security. In order to derive strong and natural notions of security in groups (similar to Poettering and Rösler [67, 68]) further research, however, is certainly necessary.

1.2 Organization of this Thesis

According to the requirements for a master thesis as part of the *TopING* program, this thesis is organized as follows: after giving an overview over the research question in this chapter, the next chapter provides further background for this work. This

background consists of prior analyses of practical instant messaging protocols as well as the formal modeling of secure and reliable group communication. We note that the original publication already entails an overview over related work and discusses modeling of group instant messaging. Consequently overlaps cannot be avoided.[1] Finally, the extended version of our original publication (and thereby the main body of this thesis) is provided with appendices that were omitted in the proceedings due to space limitations.

[1] The idea of enriching the publication with scientific background stems from short conference publications that are usual for other research fields in the electrical engineering faculty of Ruhr University Bochum. In such publications, a sufficient introduction into the scientific background is – according to the responsible examination office – seldom.

2 Background

This chapter provides an overview over the scientific fields that are related to the approach and methods used in the main body (i.e., the publication), as well as insights into constructions and techniques of the analyzed protocols, and the modeling of security and reliability, which is the basis for evaluating these protocols.

First, previous work with similar aim and approach is presented and summarized. Thereby analyses that generically evaluated applications, implementations, and protocols for instant messaging, and specific analyses, focusing on either a particular application or a certain protocol, are described. Also literature is presented that proposes and introduces new protocol constructions.

Subsequently, two important aspects of secure instant messaging are examined: reliable group communication and secure channels. The analysis of reliability and security in instant messaging demands that the definitions and properties of both fields are reviewed and validated with respect to compatibility. Then definitions and requirements of these two – partially independent – areas of research must be sensibly combined. Even though there are publications that consider fragments of both, there exists no standard notion for the purpose of applying it to real-world protocols. Therefore main results of each of both areas, as well as literature with an intersection of them are presented. It is important to note that models and constructions for secure and reliable two-party communication cannot trivially be enhanced to suffice the requirements for communication among multiple parties nor groups (where the latter can be considered as a special case of the former).

Our original publication in the main body of this thesis already entails an introduction and a section that describes related work. As a consequence, redundancies in references and explanations cannot be fully avoided. Nonetheless, this chapter shall deepen the understanding of the background and therefore broaden the explanations.

2.1 Analyses of Instant Messaging Protocols

In this section we survey analyses of instant messaging applications and protocols. Even though earlier work already touched group communication, neither a proper

definition of an attacker model was proposed nor were the most important real-world protocols described.

We classify (and accordingly present) analyses of instant messaging protocols as follows:

1. application specific or implementation focused works that reveal weaknesses independent of the actual transmission of messages,
2. positive and negative results on the security of the transmission protocols in real-world applications, and
3. systematization and modeling of instant messaging regarding security, reliability, and usability.

2.1.1 Implementation Focused Analyses

Schrittwieser et al. [74] analyze WhatsApp among other instant messaging applications regarding the initial authentication and the account management and describe found weaknesses accordingly. The described weaknesses enabled an attacker to crawl the user database or to use foreign phone numbers.

Another application specific analysis [6] focused on the information that can be retrieved from *artifacts* (i.e., user specific data) that are stored by the WhatsApp's Android application.

Finally, a newspaper article described that, even though the verification of a communication partner's public key is implemented in WhatsApp, this verification can partially be bypassed for usability reasons [52, 60].

2.1.2 Results on Security of the Transmission Protocols

The analysis of Signal's secure channel protocol started with Frosch et al. [38]. They analyze TextSecure v2, the predecessor of the Signal key exchange protocol, and identified its three main components: key exchange (*X3DH Key Agreement Protocol* [54]), key derivation (*Double Ratchet algorithm (DR algorithm)* [53]), and authenticated encryption. In their analysis they identify an Unknown Key-Share (UKS) attack and propose fixes. Although Signal now uses a slightly improved protocol (e.g., using signatures during a device's registration), the core protocol is very similar to TextSecure v2.

Kobeissi et al. [48] describe the application of formal verification methods for analyzing a slightly modified version of the Signal protocol and other real-world protocols. They derive a statement on security from an automatic cryptographic verification tool but also confirm the UKS of Frosch et al. and present attacks

on the protocol that go beyond their security definition (i.e., against further features).

Cohn-Gordon et al. [25] conduct a formal analysis on the Signal key exchange protocol. Therefore, they develop a new multi-stage key exchange security model and identify security properties that precisely capture the security that is provided by the analyzed protocol. They then apply this model to the Signal protocol, and prove it to be secure.

An initial analysis of the Threema protocol was conducted by Ahrens [3]. Independent of our work Schilling and Steinmetz presented a detailed description of the Threema message format and developed an open source implementation [71].

2.1.3 Systematization of Security in Instant Messaging

The most generic and broadest analysis and systematization of knowledge regarding trust establishment, conversation security, and transport privacy of instant messaging is by Unger et al. [86]. They split these three evaluation categories into features and properties regarding *security* and *usability and adoption*. These features and properties are then used as criteria for the analysis. The base of their analysis (with respect to information on the considered protocols) is literature research, considering publications of the developers (e.g., whitepapers), as well as scientific analyses. Consequently, in contrast to the main body of this thesis (i.e., our publication), their analysis includes no reverse engineering, source code analysis, nor protocol observation. Positive statements on the analyzed protocols in their work are thus mostly not based on cryptographic proofs. The selection of analyzed applications and protocols in [86] includes *TextSecure* (now known as *Signal*), that is evaluated regarding trust establishment and conversation security with respect to its underlying protocol *Axolotl* (now known as *Double Ratchet Algorithm*). Also *Threema* is regarded in the evaluation of trust establishment between communicating users. Since *WhatsApp* did not employ end-to-end security measures at the time of their analysis, it is not included. Regarding group communication, they conduct only a high level investigation on basic concepts and features of the protocols. This mainly includes efficiency and usability properties of the protocols and the members' role within a group.

Independent of real-world applications, Cohn-Gordon et al. [24] published a work on definitions and constructions for Future Secrecy of key exchange protocols. As there are different terms (such as *backward secrecy* or *post-compromise security*) and different definitions for this property, their work is an initial survey of the underlying concept. Especially since Signal's protocol achieves some form of Future Secrecy in the two-party communication, a generic treatment and understanding was necessary. The core idea of stateful communication with Future Secrecy (as in Signal) was then extended and deepened by Bellare et al. [12] who 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. While this first definition of ratcheting only considers unidirectional communication, the bidirectional setting was defined by Poettering and Rösler [67]. Subsequent definitions of ratcheting [5, 32, 47] deviate from the approach of defining strongest possible security (as in [12, 67]), but propose security definitions that can be efficiently instantiated.

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

Concurrent to our work a construction for ratcheting in static groups in a semi-asynchronous setting was proposed by [26]. Their security notion – similarly to recent definitions for ratcheting in the two-party setting – grasps the level of security provided by their proposed protocol rather than the best security one can hope for. As they target pure key establishment (but not messaging) and neither fully capture asynchronous communication nor dynamic groups, also their security model is not applicable for our analysis.

2.2 Modeling of Reliable Group Communication

Independent of security, group communication demands mechanisms that maintain a common understanding of *what the group is* and *which communication takes place within a group*. Reliability properties known from concepts like broadcast or group communication systems (GCS) formalize this abstract idea. In this section we provide a brief overview over notions that are related to reliability and that are relevant for our work.

Bracha and Toueg introduce the notion of *reliable broadcast* in the asynchronous setting [18]. While in the synchronous setting, all involved processes (e.g., receivers or members of a group) receive messages within the same time frame (such that communication proceeds in scheduled, isolated rounds), in the asynchronous setting each message might be received at a different time by every involved process individually (which does not hinder the participating processes to continue the protocol). Especially in instant messaging, the former setting is unrealistic as one cannot assume all (involved) users to be always online.

Abstractly, a reliable broadcast ensures for a sent message m that, if sufficiently many processes eventually participate in the protocol, then all these processes output (i.e., deliver to the user) the sent message m. Otherwise, these processes either all output the same value m^* or nothing. Mapped to group instant messaging, this property ensures that all members of a group see the same transcript (i.e., message history). Even though this property might be desired, reaching consensus on the

transcript within a group is (as described above) only possible if sufficiently many members participate in the protocol. In fact, Fischer et al. prove the impossibility of correct deterministic byzantine consensus protocols that terminate [35]. Consequently, reliable broadcast protocols are either probabilistic (resulting in an error probability), or do not always terminate. It is additionally easy to see that consensus cannot be achieved if messages are delivered instantly.

After Bracha's and Toueg's seminal work [18], many papers introduced and improved algorithms to solve the problem of validly and consistently delivering messages in a multi user setting [20, 21, 22, 50] but also refined the notion and definition to provide realistic attacker models [20].

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 [72]. Consequently 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 (as argued above).

Cachin et al. defined a *secure causal atomic broadcast* [20]. This definition requires confidentiality of sent messages until they are 'scheduled'. As soon as a message is supposed to be delivered by the receiving processes, it is scheduled to be decrypted and until then no party, including the adversary, should be able to see the message's plaintext. This notion is one step towards a confidential group communication but it implies the lack of confidentiality after the delivery. In fact, the proposed protocol reveals the plaintext during the delivery to the network and thereby to the adversary.

Chockler et al. [23] give an overview on various models and results regarding GCS (e.g., [58, 87]). They systematize different notions and definitions regarding the reliability and security of group communication in the literature. One focus and major enhancement of GCS towards reliable broadcast is the membership management and thereby the consistency of the members' view on the set of group members. A necessary property of GCS protocols is thereby to reach consensus on communicated messages as well as group membership (which is combined in the time-dependent concept of the members' *view* on the respective group).

Conclusively, the above mentioned established definitions 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 setting of our analysis but 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 listed literature but also on the satisfiability of real-world requirements such as asynchronous communication and instant message delivery.

2.3 Modeling of Secure Channels

As our summary of related work on real-world protocol analyses indicates, past analyses of instant messaging only vaguely defined the analyzed security properties. One primary component of instant messaging is the channel over which messages are securely transmitted. This channel is a key feature of instant messaging in contrast to, for example, E-Mail or SMS, in which communication consists of single unrelated messages. Here we review and discuss the definitional background of secure channels.

The simplest and widely adopted idealization and abstraction of integrity and confidentiality of single ciphertexts in a channel is authenticated encryption (AE) [8]. Bellare and Namprempre [8] systematically analyze the relations among properties and notions that are linked to (and necessary for) AE. By transferring stateless security definitions (such as indistinguishability against chosen ciphertext attackers or integrity of ciphertexts) into their stateful equivalent, Bellare et al. [11] provide the first notion of secure stateful unidirectional channels. It is remarkable that this work as well as many successors were used as abstractions for channels even though they only allow to consider each communication direction in a two-party channel separately and independently (which is corrected by Marson and Poettering [55], see below).

The core idea of stateful AE is that, with each encrypted plaintext (or decrypted ciphertext respectively), the state is changed such that a reliable communication among encryptor (sender) and decryptor (receiver) is enforced. Reliability implies that ciphertexts must be decrypted in the same order as they were encrypted (preventing replays, reordering, or loss of ciphertexts). As these strong properties may not always be necessary (or in some scenarios even restrict functionality), Kohno et al. [49] describe five levels of stateful AE. While the first level essentially is stateless AE and the fourth level is stateful AE, the remaining levels define different behavior and effects on the (decryptor's) state regarding manipulated, replayed, reordered, and lost ciphertext. For a practical reason, Paterson et al. [65] add *length hiding* as a property to AE. They show that, even though TLS employs padding within the encryption, it does not reach this *length hiding* property. Consequently ciphertexts from different plaintexts (with different length) can be distinguished.

One of the most important analyses of real-world channel (establishment) protocols adopts this notion of length hiding AE to define channel security: Jager et al. [46] analyze one of the TLS 1.2 handshake protocols. Because TLS 1.2 uses the established channel already during the handshake, the introduction of a security notion that covers the handshake as well as the actual channel protocol simultaneously was necessary. As Jager et al. assume – but provide no evidence – that the *record layer* of TLS 1.2 (i.e., the channel protocol) provides length hiding AE, Boyd et al. [17] approach this issue. They therefore describe (analogously to [49]) a hierarchy of

levels for stateful length hiding AE notions. Rogaway and Zhang [69] formalize a (common) methodology to derive a security definition for a primitive from its correctness predicate. They apply their formalization to stateful AE and thereby revisit the results of Kohno et al. [49] and Boyd et al. [17] regarding a hierarchy of stateful AE notions.

In order to capture another practically relevant feature of channels, Fischlin et al. [36] regard channel notions for streamed data. As many real-world protocols allow the processing of chunks of messages or ciphertexts, they formulate security for such channels accordingly. Günther and Mazaheri [42] enhance the security notion of channels by considering state updates. Their notion regards attackers that obtain access to one of the channel users' local state. Along the standardization of TLS 1.3, their notion requires that an exposed state must not allow an adversary to obtain information on earlier plaintexts (i.e., that were encrypted or decrypted with a state from which this exposed state originates via state updates). Essentially this notion defines forward secrecy.

Similar to the first notion of channels by Bellare et al. [11], Marson and Poettering [55] investigate security definitions related to confidentiality and integrity of channels but now consider communication bidirectionally. Interestingly, for most unidirectional notions (INT-PTXT, INT-CTXT, IND-CPA) a secure construction can trivially be duplicated (i.e., one instance for each direction) to reach the respective bidirectional security notion. For IND-CCA security, this composition does, however, not lift a unidirectionally secure construction to the bidirectional equivalent. The bidirectional channel notion by Marson and Poettering [55] is then further enriched by considering adversaries who can temporarily obtain information on the local states of the communicating parties: Jaeger and Stepanovs [45] combine the concept of secure bidirectional channels with strong security (under state exposures) of two-party communication. Thereby they directly compose the establishment of keys (which was defined by Poettering and Rösler [68] for bidirectional communication) and their usage within a channel. Alwen et al. [5] relax this notion in order to capture the security of Signal's channel protocol. This relaxation is conducted twofold: first they explicitly allow unreliable delivery of messages (which is necessary for instant messaging), then they restrict the adversary's ability in exposing the communication participants' local states (because they aim to formalize the strength of Signal's channel protocol rather than defining strongest possible security).

While all above named notions work with game based definitions Degabriele and Fischlin [29] introduce universal composable channel definitions. They thereby also consider various variants with respect to real-world requirements, such as reliability features, and attacks, such as leakage of secrets from decryption.

Channel Notion for Multi-Party Communication The first important step towards secure channels among groups of communicating parties was conducted by Eugster et al. [34]. They lift the bidirectional channel notion of Marson and Poettering [55] to the multi-party setting. However, their notion has several drawbacks when applying it to scenarios such as instant messaging.

Firstly, they only consider an a priori fixed set of communication participants. Real-world group messaging protocols, however, allow the modification of membership dynamically. Dynamic groups would introduce enormous complexity as they would require to define *membership* (and how it is changed).[1] Membership cannot be defined globally as each communication participant may have a different view of the group (since ciphertexts are not delivered to all participants simultaneously). Finally, different views of participants may result in contradicting sets of members at the same time. It is yet unclear how to formally define security precisely under these circumstances.

Another required property in the channel notion of Eugster et al. for multiple users is the *reliable* delivery of messages. This implies in-order delivery and no tolerance for message loss. Both properties are incompatible with *instant* delivery of messages. A hierarchy of levels of security in group channels, equivalent to the pairwise setting (as in [17, 49, 69]), would be necessary to obtain appropriate definitions for group instant messaging.

2.4 Modeling in this Thesis

Modeling and defining security of interactive protocols (such as authenticated key exchange) is often conducted rather intuitively than formally and precisely. As our analysis focuses on finding deviations from the security definition and not on proving accordance to it, an intuitive approach may suffice. In order to propose enhancements for *insecure* protocols, such that they are called *secure* again, it is, however, necessary to determine the target (i.e., the security definition) sufficiently precisely. In the following we describe how our model is influenced by introduced related work and how far it improves previous notions.

Group Instant Messaging as a Primitive In order to provide a definition of *secure and reliable group instant messaging*, a syntax of this primitive first needs to be defined.[2] As this primitive transforms user interaction (with the communicat-

[1] Please note that the earlier cited result of Schiper and Toueg [72] (simply treating membership via the consensus of the group's communication transcript) cannot be applied as it requires consensus, which is incompatible with instant delivery of messages.

[2] The necessity for a syntax definition is explained below. A prominent example for a tradition of 'formal' security analyses that get along without a syntax definition is *authenticated key exchange*.

ing device) into network traffic (among the communicating devices), we define the primitive to consist of two sets of algorithms:

User Interface: algorithms that take commands from the users (e.g., sending a message to a group or modifying the set of group members) and then translate them into ciphertexts, sent over the network, and notifications, delivered to the calling user, and

Network Interface: algorithms that receive ciphertexts from the network (from other users' *User Interface*) and then translate them into notifications, delivered to the receiving user, and ciphertexts, sent over the network.

One can imagine that, in a network layer model (e.g., the OSI model), the User Interface enables communication with higher layers while the Network Interface provides communication with lower layers.[3]

The outputting algorithms (i.e., the actual sending to the network and the notification delivery to the users) are not part of the protocol specification but of the environment. Theoretically, the network delivery is conducted by the adversary. Practically, ciphertexts are sent via TCP/IP to a central service provider (e.g., the WhatsApp servers) that forwards them via TCP/IP to the receiving devices (e.g., Smartphones). Similarly, the adversary initiates the sending of ciphertexts and obtains the notifying feedback in the theoretic model, while in practice a graphical user interface allows for the interaction with the algorithms.

We consider explicit delivery acknowledgments as a special notification feature of instant messaging in our syntax. The main motivation behind this is the conflict of instant delivery of messages and the maintenance of a (semi-)reliable communication. When defining security and reliability, this feature is hence of special importance.

Many models for interactive protocols disregard an explicit syntax and only describe the attacker's ability to interact with the protocol (in a game that defines security). This is insufficient for two reasons: if the respective primitive's syntax is not defined (independent of the attacker), it's functionality does not become clear. It is consequently unclear for which class of protocols the resulting security definition is applicable. Secondly, without a proper syntax, correctness cannot be defined. As correctness is often the basis for defining security (cf. [69]), the adequacy of the proposed definition is ambiguous.

Definition of Adversarial Power In our definition we consider various threats that we translate into adversarial capabilities. Commonly a real-world attacker has access to an open protocol such that it can participate as a normal user (and therefore interact with the attacked victims). Furthermore, no security mechanism would be

[3]This idea stems from countless discussions with Bertram Poettering.

employed on the network communication protocol, if the considered network would be trusted. As a result, we allow the adversary to actively manipulate the traffic among the user devices and the central service provider.[4] Finally, end-to-end security mechanisms only make sense if an evil central service provider should be defended. Therefore, we define the adversary to be able to manipulate the end-to-end protected ciphertexts – and all additional meta data –, sent within any transport layer protection among user devices and the central server.

As it is described in our initial motivation, a special kind of threat is relevant in instant messaging particularly: attackers that temporarily obtain access to the local states of the attacked users become a practical risk since sessions in instant messaging usually last a long time. Furthermore, the statefulness of instant messaging can be utilized to defend against such attacks. Consequently we also consider adversaries who can compromise session states of users.

Often also users' long term secrets are an objective of attacks. These static secrets can be compromised as long as the identity of a user exists. In contrast to client-to-server scenarios, physical mechanisms (such as hardware security modules) to protect long term secrets in instant messaging are additionally unrealistic, which increases the relevance of this attacker type.

Definition of Reliability and Security Properties In our definition of reliability and security, we describe which events must (or must not) occur in the presence of the described attackers with respect to the syntax of group instant messaging. Apart from standard goals such as confidentiality (also against compromising attackers) and authentication, properties specific to group instant messaging are introduced. The property that only group members can contribute massages to a group (called *No Creation*) describes a distinction towards multicast protocols (in which everyone can send, but only a specified set of users can receive). Furthermore, we define *Closeness* as an important property for secure group communication (meaning that only *special* group members can modify a group's set of members).

As the reliable delivery of messages in groups (i.e., with consensus) is not achievable if these messages should be delivered (i.e, displayed) immediately to the user, no guarantees on the validity of the transcript (as the concatenation of communicated group messages) would exist. Real-world protocols therefore introduced explicit delivery acknowledgments to indicate messages' receipt status via the application's graphical user interface (e.g., towards the original sender). The reliability property that we deduce for this feature is called *Traceable Delivery*: an acknowledgment must not be indicated if the respective message was not delivered to all designated receivers (i.e., all group members). This definition clearly deviates from common reliability

[4] As the next described attacker is more powerful and more relevant in the context of end-to-end protected protocols, we do not explicitly regard attackers that only have access to the traffic among users and service provider.

definitions as it only requires the unforgeability of acknowledgments. We recall that no consensus can be achieved in instant messaging and consequently no guarantees on the delivery of acknowledgments can be required.

Finally we define *soft* ordering definitions. If instant delivery of messages is the first priority of instant messaging, neither FIFO order nor causal order can be achieved (as messages can be lost in transmission this would result in the termination of communication). We therefore propose weaker forms of ordering that allow instant message delivery but enforce the rejection of belatedly received ciphertexts.

Subsumption of Modeling in the Publication The previous sections provide a broad introduction into various aspects of modeling of secure and reliable communication in groups. As we describe, many theoretic negative results on reliability, such as *consensus*, show that requirements regarding reliability and efficiency need to be balanced in order to gain a desirable user experience for real-world scenarios.

Our notion of security and reliability is consequently the first one that explicitly considers the core functionalities of group communication and at the same time respects the characteristics of instant messaging. Even though this definition suffices the needs of our practical analysis of real-world protocols, yet there exists no standard definition that is precise enough to be used for proving security of group instant messaging.

As described above, our publication chooses reliability requirements by also respecting the specific applications' user interfaces and the developers' assertions. In addition to that, the paper further evaluates additional goals of reliability and discusses the effect of not reaching them.

This is the full version of the article [70] published in the proceedings of
3rd IEEE European Symposium on Security and Privacy (EuroS&P 2018).

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

Paul Rösler, Christian Mainka, Jörg Schwenk
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. Since these protocols and their implementations are mostly undocumented for the public and two out of three applications among them are closed source, we describe the group protocols employed in 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 [1, 19, 33].

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

With the revelation of mass surveillance activities by intelligence agencies, new IM applications incorporating end-to-end encryption launched, as well es established IM applications added encryption to their protocols to protect the communication towards the message delivering servers. Hence analyses, investigating these protocols, also include malicious server-based attacks [40, 48].

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 [59], analyzed [38], and proven [25]. *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 [64], WhatsApp [89], Threema [84], Google Allo [62], and Facebook Messenger [63]. 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 [28], 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 harder to analyze and compare their security properties. This leads to two major challenges. First, the applications must be reverse engineered [13, 38, 39] for retrieving a protocol description. Second, third-party implementations are often blocked by providers [88] 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, 43] and related formalizations like group communication systems (GCS) [23] 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 [64], Threema [84], and WhatsApp [89]. 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 [76]. 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 [75, 83]. 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 (§8).

  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 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.

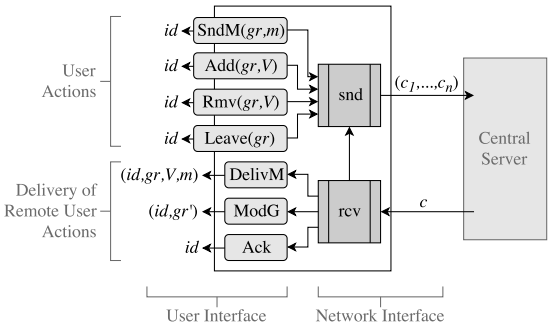

Figure 1: Overview over syntax of group instant messaging protocols showing the interacting user's interfaces on left and the interfaces of the application to the network on the right.

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

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

The first two algorithms (snd, 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 $\text{algA}^{\text{algB}}$.

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.

- $\text{snd} \to \vec{c}$: Outputs a vector of ciphertexts, designated to the central server, to the network.
- $\text{rcv}^{\text{snd,DelivM,ModG,Ack}}(c)$: 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:

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

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

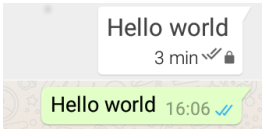

Figure 2: Double checkmarks in Signal (upper screenshot) and WhatsApp (lower screenshot) indicating that a group message was successfully delivered to *all* members' devices.

Every algorithm 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 $\text{Actn}(gr) \to id$.

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:

- $\text{DelivM} \to (id, gr, V, m)$: Stores m with reference string id from sender V in group gr for displaying it to user U.
- $\text{ModG} \to (id, gr')$: Updates the description of group gr with $ID_{gr} = ID_{gr'}$ to gr' after the remote modification with reference string id.
- $\text{Ack} \to id$: 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 $\text{Actn}(gr) \to id$, we denote both algorithms DelivM and ModG as $\text{Deliv} \to (id, gr)$.

2.3 Threat Model

We consider three types of adversaries against secure instant messaging protocols. Thereby we define the adversaries 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 [40, 48], this adversary models attacks against the transport layer security between users and the central server [4, 7, 56].

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 [31, 43]. As we will argue in section 8, 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 $\mathrm{DelivM} \rightarrow (id, gr, V, m_r)$.

▶ *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* [25] – 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 [12, 25, 26] 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. [25] 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 $\mathrm{DelivM} \rightarrow (id, gr, U, m)$, 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 $\mathrm{DelivM} \rightarrow (id, gr, V, m)$ with sender V, then $V \in \mathcal{G}_{gr}$ holds.

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

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].

- *No Duplication.* For every user action $\text{Actn}(gr) \rightarrow id$ initiated by user U for group gr the resulting delivery algorithm $\text{Deliv} \rightarrow (id, gr)$ is invoked at most once by each group member $V_i \in \mathcal{G}_{gr}$.
- *Traceable Delivery.* If the delivery of user action $\text{Actn}(gr) \rightarrow id$ by user U is acknowledged to user $V' \in \mathcal{G}_{gr}$ by invoking $\text{Ack} \rightarrow id$, then the respective $\text{Deliv} \rightarrow (id, gr)$ 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 $\text{Actn}_1(gr) \rightarrow id_1$ before $\text{Actn}_2(gr) \rightarrow id_2$, then member $V \in \mathcal{G}_{gr}$ will not invoke $\text{Deliv}_1 \rightarrow (id_1, gr)$ after she invoked $\text{Deliv}_2 \rightarrow (id_2, gr)$.
- *Weak Causal Order.* If user U_1 calls $\text{Actn}_1(gr) \rightarrow id_1$ and user U_2 calls $\text{Actn}_2(gr) \rightarrow id_2$ after she invoked $\text{Deliv}_1 \rightarrow (id_1, gr)$, then member $V \in \mathcal{G}_{gr}$ will not invoke $\text{Deliv}_1 \rightarrow (id_1, gr)$ after she invoked $\text{Deliv}_2 \rightarrow (id_2, gr)$.

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 $(id, gr') \leftarrow \text{ModG} : |\mathcal{G}_{gr}^{old}| < |\mathcal{G}_{gr'}|, ID_{gr} = ID_{gr'}$, then an administrator $U^* \in \mathcal{G}_{gr}^*$ called $\text{Add}(gr, V)$ 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 $(id, gr') \leftarrow \text{ModG} : |\mathcal{G}_{gr}^{old}| > |\mathcal{G}_{gr'}|, ID_{gr} = ID_{gr'}$, then either an administrator $U^* \in \mathcal{G}_{gr}^*$ called $\text{Rmv}(gr, V)$ to remove member $V = \mathcal{G}_{gr}^{old} \setminus \mathcal{G}_{gr'}$ from the group, or member V called $\text{Leave}(gr)$ 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

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

[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.

made: if a member invokes 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 8.

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 [79, 80]. 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 [13, 39]. 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 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. An overview on the results of our protocol evaluation can be found in section 7. 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.

In order to provide a comparable description of the protocols, Figures 4 (Signal), 6 (WhatsApp) and 8 (Threema) depict an example protocol run of each protocol containing direct and group communication. The figures are meant to highlight the differences in the three group messaging 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 instant messaging application available for Android, iOS, and as a Google Chrome extension [82]. 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 [25, 38].

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.

The Signal application implements *Curve25519* [15] and *HMAC-SHA256* [9] for the key derivation (*Double Ratchet algorithm*). The *HMAC* is also used for message authenticity in combination with *AES-CBC-PKCS5Padding* [27] 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 model. Figures 4, 6, and 8 depict an example 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* [54] that uses static and ephemeral Diffie-Hellman shares of both parties. This root key initializes the *Double Ratchet algorithm (DR algorithm)* [53], which can be seen as a stateful encryption algorithm [10]. 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 5). 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.

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

A schematic description of the DR algorithm when used in the Signal and WhatsApp messaging protocol can be seen in Figure 9. The usage of the key streams can be seen in the example protocol run in Figures 4 and 6.

4.2 Group Protocol

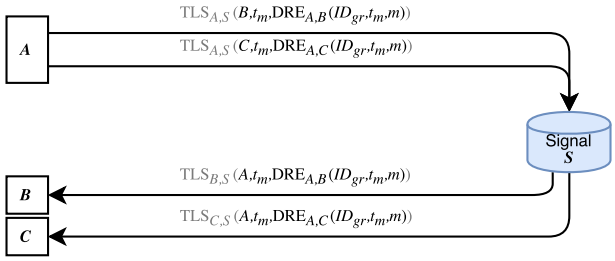

Figure 3: Schematic depiction of Signal's traffic, generated for a message m from sender A to receivers B and C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$. Transport layer protection is not in the analysis scope (gray).

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 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 $\mathcal{S}$[5].

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

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

[5]We omitted irrelevant fields regarding our evaluation in the message format. The whole format can be found in the format description [81]. We also left out Google's Cloud Messaging (GCM) service for clarity.

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 $info_{gr}$. 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.

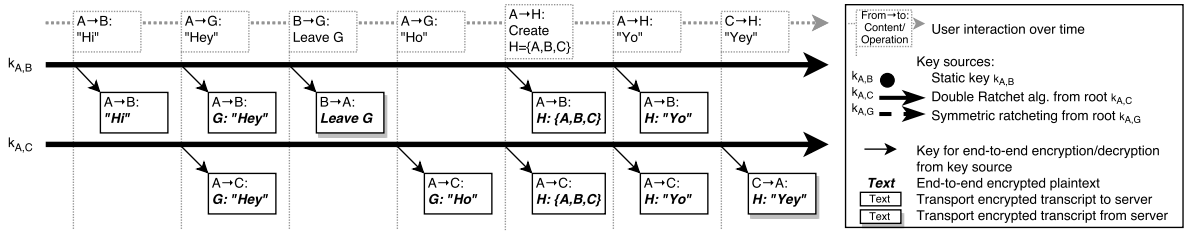

Figure 4: Schematic depiction of key streams of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in Signal. The legend of the graphic also regards to Figures 6 and 8.

4.2.3 Example Protocol Run

Figure 4 depicts an example 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 4 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 verified two weaknesses of Signal and created proof-of-concepts for them. First, we burgle into a group by writing group management messages into it. Second, we make a victim believe that a message is delivered while it is not.

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* $m = (\{\mathcal{A}\}, info_{gr}))$ 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.

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.
- *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 [6]. 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. The decryption of received messages follows this order. Since old omitted keys are removed after a 2000 new keys are derived, reordering by the server is restrictedly possible. Henceforth neither *FIFO Order* nor *Causal Order* are provided by Signal.

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[7].

Even though WhatsApp is a closed source application, there exist open source implementations [39, 57] whose usage is forbidden and aimed to be prevented by WhatsApp [88]. We used a fork[8] of Galal's

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

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

[8] https://github.com/colonyhq/yowsup

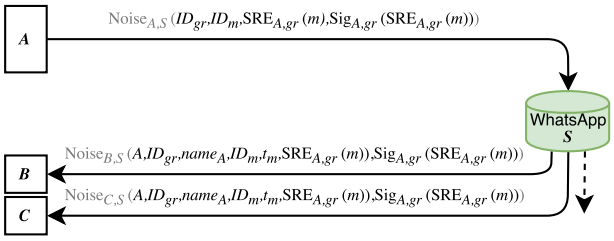

Figure 5: Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in WhatsApp.

implementation [39] to analyze the traffic, generated by the official WhatsApp Android application[9].

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* [15].

Our analysis confirms the description of WhatsApp's technical white paper [44] 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* [66] to protect the communication between the clients and the server on the transport layer [44]. 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* [54] and the *DR algorithm* [53], is integrated in WhatsApp in order to establish a confidential channel for messaging between two users [44]. A detailed description of these building blocks can be found in section 4 and Figure 9.

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.

[9] Version 2.17.107 from the Google Play Store

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 5). 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 as it can be seen in Figures 6 and 9. 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.

5.2.3 Example Protocol Run

Figure 6 depicts an example 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.

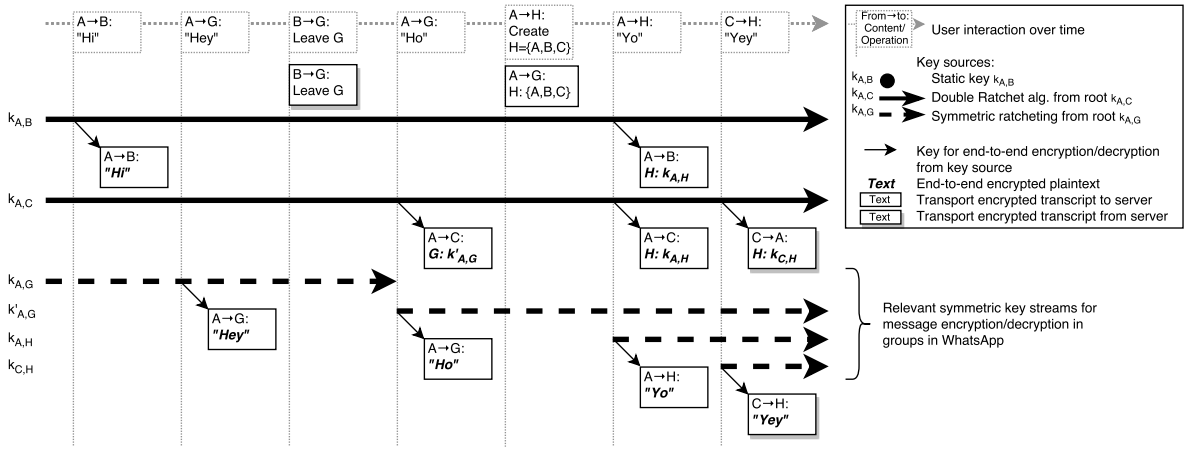

Figure 6: Schematic depiction of key streams of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in WhatsApp.

5.3 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[10]:

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

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.

[10]Schematic representation of modification message for adding a new member to 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 $\mathcal{G}_{gr}^{new} := \mathcal{G}_{gr} \cup \{\mathcal{A}\}$, (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.

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* 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. By employing a reference to a previous message, *Causal Order* is at least preserved for this reference.

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 operating systems [84]. 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 [13] which was based on an analysis of Ahrens [3]. 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.[11]

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 [16] with integrity and authenticity protection using the *Poly1305-AES* MAC [14].

We identified that Threema implements *Curve25519* for all DHKEs, which is also described in [85].

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 8.

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

[11] Version v.2.92.323

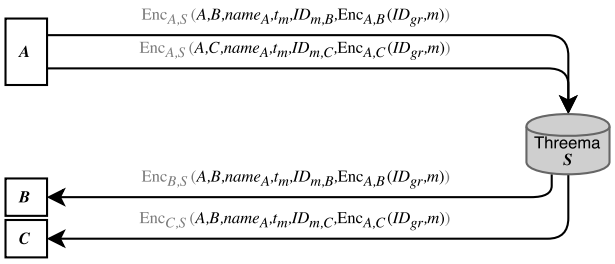

Figure 7: Schematic depiction of traffic, generated for a message m from sender A to receivers B, C in group gr with $\mathcal{G}_{gr} = \{A, B, C\}$ in Threema.

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 8). 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 7 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.

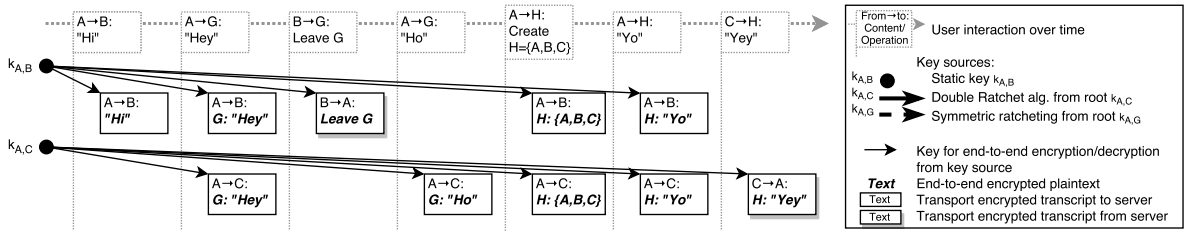

Figure 8: Schematic depiction of keys of A and ciphertexts from and to A that are used when sending and receiving direct and group messages and modifying the groups in Threema.

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 Example 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 6, 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 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. [12]

7 Evaluation Summary

Table 1 summarizes our evaluation results. The gray cells indicate, that a security goal is not reached because another security goal is not provided as well. Since a compromising Malicious User can break the Additive Closeness of Signal, Future Secrecy is implicitly violated as well. Signal is directly attackable regarding Additive Closeness and No Creation. In contrast, breaking No Creation in WhatsApp results from breaking Additive Closeness. Threema is vulnerable to No Duplication. This can consequentially be used for breaking Additive Closeness by rewinding group management operations and breaking Additive Closeness again breaks No Creation. Threema updated their application in response to our responsible disclosure. Consequently No Duplication, No Creation, and Additive Closeness are not attackable anymore.

The descriptions of the attacks against the protocols regarding our security model always selects the weakest successful attacker. Consequently a *Compromising Malicious Server* can break the same goals as a *Malicious User* with compromising access to the victim's secrets.

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

	E2E Confidentiality	Forward Secrecy	Future Secrecy	Msg. Authentication	Traceable Delivery	No Duplication	No Creation	Closeness
Signal								
WhatsApp			✗					
Threema		✗	✗		✗			

Table 1: ✗: Not implemented security goal;
: Not reached against *Malicious User* who can compromise victim;
: Not reached against *Malicious Server*;
Gray symbols: not provided as side effect of another unreached security goal

8 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.

8.1 Fixing the Protocols

8.1.1 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 WhisperSystems, 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[13]. 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 8.2.1).

8.1.2 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.

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

8.1.3 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[14].

8.2 General Outcomes

8.2.1 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 [2, 30, 37, 51]. 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.

8.2.2 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 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 [72]. 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

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

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. [26].

8.2.3 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 Eugster et al. [34] 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.

9 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. [74] analyze IM applications regarding the initial authentication and the account management and describe weaknesses accordingly. Unger et al. [86] 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. [38]. 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. [48] 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. [25] 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 [24]. Independent of our work, Schliep et al. [73] very recently conducted an analysis of Signal in which they revealed, among other shortcomings, consistency weaknesses in the group protocol. Bellare et al. [12] 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. [74] analyzed WhatsApp among other IM applications regarding the authentication and account management and found several vulnerabilities. Another application specific analysis [6] 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 [52, 60]. In addition to these analyses, the WhatsApp protocol was implemented and published as open source projects [39, 57].

Analyses of Threema. An initial analysis of the Threema protocol was conducted by Ahrens [3]. Based on this Berger [13] 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 [71].

Security in Multi User Settings.

Cohn-Gordon et al. [26] 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 [18]. Since then many works introduced and improved algorithms to solve the problem of validly and consistently delivering messages in a multi user setting [20, 21, 22, 50] but also refined the notion and definition to provide realistic attacker models [20].

Chockler et al. [23] give an overview on various models and results regarding group communication systems (GCS) like [58, 87] and others. They systematize different notions and definitions regarding the reliability and security of group communication in the literature.

Eugster et al. [34] 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 8, their model requires stronger notions of reliability at costs of the *instant* message delivery.

10 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 [25, 38, 48], 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 [26]. 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.

A Signal Key Exchange Protocol and its Usage

Figure 9 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.

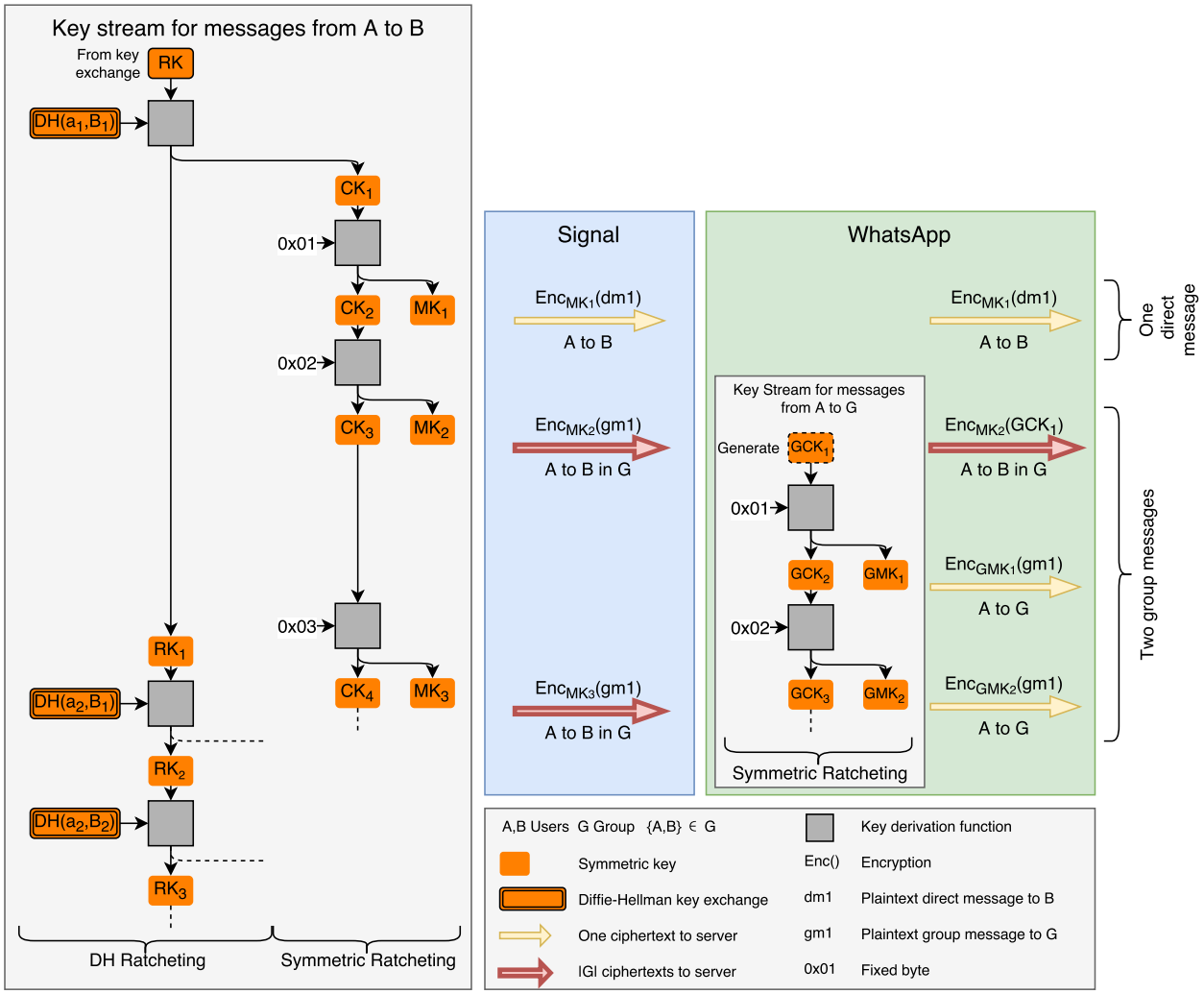

Figure 9: Sender key stream from A to B and ciphertexts from A to the server when sending one direct message to B and two group messages to a group G of which A and B are members in Signal and WhatsApp.

Bibliography

[1] Communications market report, 2017. https://www.ofcom.org.uk/__data/assets/pdf_file/0017/105074/cmr-2017-uk.pdf.

[2] B. Adamson, C. Bormann, M. Handley, and J. Macker. Multicast Negative-Acknowledgment (NACK) Building Blocks. RFC 5401, IETF, November 2008. URL https://tools.ietf.org/html/rfc5401.

[3] Jan Ahrens. Threema protocol analysis, 2014. URL http://blog.jan-ahrens.eu/files/threema-protocol-analysis.pdf.

[4] Martin R. Albrecht and Kenneth G. Paterson. Lucky microseconds: A timing attack on amazon's s2n implementation of TLS. In Marc Fischlin and Jean-Sébastien Coron, editors, *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*, volume 9665 of *Lecture Notes in Computer Science*, pages 622–643. Springer, 2016.

[5] Joël Alwen, Sandro Coretti, and Yevgeniy Dodis. The double ratchet: Security notions, proofs, and modularization for the signal protocol. Cryptology ePrint Archive, Report 2018/1037, 2018. https://eprint.iacr.org/2018/1037.

[6] Cosimo Anglano. Forensic analysis of whatsapp messenger on android smartphones. *Digital Investigation*, 11(3):201–213, 2014.

[7] 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. DROWN: breaking TLS using sslv2. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 689–706. USENIX Association, 2016.

[8] Mihir Bellare and Chanathip Namprempre. Authenticated encryption: Relations among notions and analysis of the generic composition paradigm. In *Advances in Cryptology - ASIACRYPT 2000, 6th International Conference on the Theory and Application of Cryptology and Information Security, Kyoto, Japan, December 3-7, 2000, Proceedings*, pages 531–545, 2000.

[9] Mihir Bellare, Ran Canetti, and Hugo Krawczyk. 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*, pages 1–15, 1996.

[10] Mihir Bellare, Anand Desai, E. Jokipii, and Phillip Rogaway. 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*, pages 394–403, 1997.

[11] Mihir Bellare, Tadayoshi Kohno, and Chanathip Namprempre. Authenticated encryption in SSH: provably fixing the SSH binary packet protocol. In *Proceedings of the 9th ACM Conference on Computer and Communications Security, CCS 2002, Washington, DC, USA, November 18-22, 2002*, pages 1–11, 2002.

[12] Mihir Bellare, Asha Camper Singh, Joseph Jaeger, Maya Nyayapati, and Igors Stepanovs. Ratcheted encryption and key exchange: The security of messaging. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 619–650, 2017.

[13] Philipp Berger. Open source implementation of a threema desktop client, 2016. URL `https://github.com/blizzard4591/openMittsu`. Based on the descriptions of Jan Ahrens' paper.

[14] Daniel J. Bernstein. The poly1305-aes message-authentication code. In *Fast Software Encryption: 12th International Workshop, FSE 2005, Paris, France, February 21-23, 2005, Revised Selected Papers*, pages 32–49, 2005.

[15] Daniel J. Bernstein. 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*, pages 207–228, 2006.

[16] Daniel J. Bernstein. The salsa20 family of stream ciphers. In *New Stream Cipher Designs - The eSTREAM Finalists*, pages 84–97. 2008.

[17] Colin Boyd, Britta Hale, Stig Frode Mjølsnes, and Douglas Stebila. From stateless to stateful: Generic authentication and authenticated encryption constructions with application to TLS. In *Topics in Cryptology - CT-RSA 2016 - The Cryptographers' Track at the RSA Conference 2016, San Francisco, CA, USA, February 29 - March 4, 2016, Proceedings*, pages 55–71, 2016.

[18] Gabriel Bracha and Sam Toueg. Asynchronous consensus and broadcast protocols. *J. ACM*, 32(4):824–840, 1985.

[19] Business2Community. Are instant messaging apps the future of the (mobile) internet?, August 2015. URL `http://www.business2community.com/mobile-apps/instant-messaging-apps-future-mobile-internet-01313577`.

[20] Christian Cachin, Klaus Kursawe, Frank Petzold, and Victor Shoup. 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*, pages 524–541, 2001.

[21] Christian Cachin, Klaus Kursawe, and Victor Shoup. Random oracles in constantinople: Practical asynchronous byzantine agreement using cryptography. *J. Cryptology*, 18(3):219–246, 2005.

[22] Ran Canetti and Tal Rabin. 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*, pages 42–51, 1993.

[23] Gregory V. Chockler, Idit Keidar, and Roman Vitenberg. Group communication specifications: a comprehensive study. *ACM Comput. Surv.*, 33(4):427–469, 2001.

[24] Katriel Cohn-Gordon, Cas J. F. Cremers, and Luke Garratt. On post-compromise security. In *IEEE 29th Computer Security Foundations Symposium, CSF 2016, Lisbon, Portugal, June 27 - July 1, 2016*, pages 164–178, 2016.

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

[26] Katriel Cohn-Gordon, Cas Cremers, Luke Garratt, Jon Millican, and Kevin Milner. On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 1802–1819, 2018.

[27] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, pages 277–284, 1998.

[28] Sergej Dechand, Dominik Schürmann, Karoline Busse, Yasemin Acar, Sascha Fahl, and Matthew Smith. An empirical study of textual key-fingerprint representations. In Thorsten Holz and Stefan Savage, editors, *25th USENIX Security Symposium, USENIX Security 16, Austin, TX, USA, August 10-12, 2016.*, pages 193–208. USENIX Association, 2016.

[29] Jean Paul Degabriele and Marc Fischlin. Simulatable channels: Extended security that is universally composable and easier to prove. In *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and*

Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III, pages 519–550, 2018.

[30] Christophe Diot, Walid Dabbous, and Jon Crowcroft. Multipoint communication: A survey of protocols, functions, and mechanisms. *IEEE Journal on Selected Areas in Communications*, 15(3):277–290, 1997.

[31] Sisi Duan, Lucas Nicely, and Haibin Zhang. Byzantine reliable broadcast in sparse networks. In *15th IEEE International Symposium on Network Computing and Applications, NCA 2016, Cambridge, Boston, MA, USA, October 31 - November 2, 2016*, pages 175–182, 2016.

[32] F. Betül Durak and Serge Vaudenay. Bidirectional asynchronous ratcheted key agreement without key-update primitives. Cryptology ePrint Archive, Report 2018/889, 2018. `https://eprint.iacr.org/2018/889`.

[33] eMarketer. Mobile messaging to reach 1.4 billion worldwide in 2015, November 2015. URL `https://www.emarketer.com/Article/Mobile-Messaging-Reach-14-Billion-Worldwide-2015/1013215`.

[34] Patrick Eugster, Giorgia Azzurra Marson, and Bertram Poettering. A cryptographic look at multi-party channels. In *31st IEEE Computer Security Foundations Symposium, CSF 2018, Oxford, United Kingdom, July 9-12, 2018*, pages 31–45, 2018.

[35] Michael J. Fischer, Nancy A. Lynch, and Mike Paterson. Impossibility of distributed consensus with one faulty process. *J. ACM*, 32(2):374–382, 1985.

[36] Marc Fischlin, Felix Günther, Giorgia Azzurra Marson, and Kenneth G. Paterson. Data is a stream: Security of stream-based channels. In *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part II*, pages 545–564, 2015.

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

[38] Tilman Frosch, Christian Mainka, Christoph Bader, Florian Bergsma, Jörg Schwenk, and Thorsten Holz. How secure is textsecure? In *IEEE European Symposium on Security and Privacy, EuroS&P 2016, Saarbrücken, Germany, March 21-24, 2016*, pages 457–472, 2016.

[39] Tarek Galal. Open source implementation of a whatsapp client, 2016. URL `https://github.com/tgalal/yowsup`. This code is not maintained anymore but some of its forks are still under development.

[40] Christina Garman, Matthew Green, Gabriel Kaptchuk, Ian Miers, and Michael Rushanan. 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.*, pages 655–672, 2016.

[41] Shafi Goldwasser and Silvio Micali. Probabilistic encryption and how to play mental poker keeping secret all partial information. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing, May 5-7, 1982, San Francisco, California, USA*, pages 365–377, 1982.

[42] Felix Günther and Sogol Mazaheri. A formal treatment of multi-key channels. In *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part III*, pages 587–618, 2017.

[43] Vassos Hadzilacos and Sam Toueg. Distributed systems (2nd ed.). chapter Fault-tolerant Broadcasts and Related Problems, pages 97–145. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1993. ISBN 0-201-62427-3.

[44] WhatsApp Inc. Whatsapp encryption overview, 2016. URL `https://www.whatsapp.com/security/WhatsApp-Security-Whitepaper.pdf`. Technical white paper.

[45] Joseph Jaeger and Igors Stepanovs. Optimal channel security against fine-grained state compromise: The safety of messaging. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 33–62, 2018.

[46] Tibor Jager, Florian Kohlar, Sven Schäge, and Jörg Schwenk. On the security of TLS-DHE in the standard model. In *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, pages 273–293, 2012.

[47] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. Cryptology ePrint Archive, Report 2018/954, 2018. `https://eprint.iacr.org/2018/954`.

[48] Nadim Kobeissi, Karthikeyan Bhargavan, and Bruno Blanchet. 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*, 2017.

[49] Tadayoshi Kohno, Adriana Palacio, and John Black. Building secure cryptographic transforms, or how to encrypt and mac. Cryptology ePrint Archive, Report 2003/177, 2003. `https://eprint.iacr.org/2003/177`.

[50] Klaus Kursawe and Victor Shoup. Optimistic asynchronous atomic broadcast. In *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, pages 204–215, 2005.

[51] Brian Neil Levine and J. J. Garcia-Luna-Aceves. A comparison of reliable multicast protocols. *Multimedia Syst.*, 6(5):334–348, 1998. URL `http://link.springer.de/link/service/journals/00530/bibs/8006005/80060334.htm`.

[52] Tobias Boelter Manisha Ganguly. Whatsapp vulnerability allows snooping on encrypted messages. *The Guardian*, 2017. URL `https://www.theguardian.com/technology/2017/jan/13/whatsapp-backdoor-allows-snooping-on-encrypted-messages`.

[53] Moxie Marlinspike and Trevor Perrin. The double ratchet algorithm, 11 2016. URL `https://whispersystems.org/docs/specifications/doubleratchet/doubleratchet.pdf`.

[54] Moxie Marlinspike and Trevor Perrin. The x3dh key agreement protocol, 11 2016. URL `https://whispersystems.org/docs/specifications/x3dh/x3dh.pdf`.

[55] Giorgia Azzurra Marson and Bertram Poettering. Security notions for bidirectional channels. *IACR Trans. Symmetric Cryptol.*, 2017(1):405–426, 2017.

[56] Christopher Meyer, Juraj Somorovsky, Eugen Weiss, Jörg Schwenk, Sebastian Schinzel, and Erik Tews. Revisiting SSL/TLS implementations: New bleichenbacher side channels and attacks. In *23rd USENIX Security Symposium*, pages 733–748, 2014.

[57] mgp25. Open source implementation of a whatsapp php api, 2016. URL `https://github.com/mgp25/Chat-API`.

[58] Louise E. Moser, P. M. Melliar-Smith, Deborah A. Agarwal, Ravi K. Budhia, and Colleen A. Lingley-Papadopoulos. Totem: A fault-tolerant multicast group communication system. *Commun. ACM*, 39(4):54–63, 1996.

[59] Moxie Marlinspike. Advanced cryptographic ratcheting, 2013. URL `https://whispersystems.org/blog/advanced-ratcheting/`.

[60] Moxie Marlinspike. There is no whatsapp 'backdoor', 2017. URL `https://whispersystems.org/blog/there-is-no-whatsapp-backdoor/`.

[61] Jarkko Oikarinen and Darren Reed. Internet relay chat protocol. Technical report, 1993.

[62] Open Whisper Systems. Open whisper systems partners with google on end-to-end encryption for allo, 2017. URL `https://whispersystems.org/blog/allo/`.

[63] Open Whisper Systems. Facebook messenger deploys signal protocol for end to end encryption, 2017. URL `https://whispersystems.org/blog/facebook-messenger/`.

[64] Open Whisper Systems. Signal website, 2017. URL `https://signal.org/`.

[65] Kenneth G. Paterson, Thomas Ristenpart, and Thomas Shrimpton. Tag size does matter: Attacks and proofs for the TLS record protocol. In *Advances in Cryptology - ASIACRYPT 2011 - 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, December 4-8, 2011. Proceedings*, pages 372–389, 2011.

[66] Trevor Perrin. The noise protocol framework, 2016. URL `http://noiseprotocol.org/noise.pdf`.

[67] Bertram Poettering and Paul Rösler. Towards bidirectional ratcheted key exchange. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part I*, pages 3–32, 2018.

[68] Bertram Poettering and Paul Rösler. Asynchronous ratcheted key exchange. Cryptology ePrint Archive, Report 2018/296, 2018. `https://eprint.iacr.org/2018/296`.

[69] Phillip Rogaway and Yusi Zhang. Simplifying game-based definitions - indistinguishability up to correctness and its application to stateful AE. In *Advances in Cryptology - CRYPTO 2018 - 38th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2018, Proceedings, Part II*, pages 3–32, 2018.

[70] Paul Rösler, Christian Mainka, and Jörg Schwenk. More is less: On the end-to-end security of group chats in signal, whatsapp, and threema. In *2018 IEEE European Symposium on Security and Privacy, EuroS&P 2018, London, United Kingdom, April 24-26, 2018*, pages 415–429, 2018.

[71] Roland Schilling and Frieder Steinmetz. A look into the mobile messaging black box, 12 2016. URL `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`.

[72] André Schiper and Sam Toueg. From set membership to group membership: A separation of concerns. *IEEE Trans. Dependable Sec. Comput.*, 3(1):2–12, 2006.

[73] Michael Schliep, Ian Kariniemi, and Nicholas Hopper. Is bob sending mixed signals? In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society, Dallas, TX, USA, October 30 - November 3, 2017*, pages 31–40, 2017.

[74] Sebastian Schrittwieser, Peter Frühwirt, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, and Edgar R. Weippl. 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*, 2012. URL http://www.internetsociety.org/guess-whos-texting-you-evaluating-security-smartphone-messaging-applications.

[75] Signal. Signal private messenger in google play, 2017. URL https://play.google.com/store/apps/details?id=org.thoughtcrime.securesms.

[76] Statista. Most popular messaging apps, 2017. URL https://www.statista.com/statistics/258749/most-popular-global-mobile-messenger-apps/.

[77] Statista, Inc. What are the three methods of communication that you use most often to talk to your friends or stay in touch with them?, 2014. URL https://www.statista.com/statistics/455029/teens-popular-means-of-communication-by-age-group-germany/.

[78] Statista, Inc. Duration of current smartphone ownership in the u.s. 2017, 2017. URL https://www.statista.com/statistics/716124/duration-of-use-of-smartphones-in-the-us/.

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

[80] Open Whisper Systems. Source code of signal-service library, 11 2016. URL https://github.com/WhisperSystems/libsignal-service-java/commit/460cd7559caa74bb6539c72865c71de660a69bac. Java Library Version 2.4.1.

[81] Open Whisper Systems. Message format in the signal protocol, 11 2016. URL https://github.com/WhisperSystems/libsignal-service-java/blob/4cedb5c31c11c1e8811b3bb7cd68d56ff7e0c03f/protobuf/SignalService.proto. Specified with Google Protocol Buffers.

[82] Open Whisper Systems. Signal github repository, 05 2017. URL https://github.com/WhisperSystems/.

[83] Threema. Threema in google play, 2017. URL https://play.google.com/store/apps/details?id=ch.threema.app.

[84] Threema. Threema website, 2017. URL https://threema.ch/en.

[85] Threema GmbH. Threema cryptography whitepaper, 2016. URL `https://threema.ch/press-files/2_documentation/cryptography_whitepaper.pdf`.

[86] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. Sok: Secure messaging. In *2015 IEEE Symposium on Security and Privacy, SP 2015, San Jose, CA, USA, May 17-21, 2015*, pages 232–249, 2015.

[87] Robbert van Renesse, Kenneth P. Birman, and Silvano Maffeis. Horus: A flexible group communication system. *Commun. ACM*, 39(4):76–83, 1996.

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

[89] WhatsApp. Whatsapp security, 2017. URL `https://www.whatsapp.com/security/`.