POSTER: End-to-End Secure Mobile Group Messaging with Conversation Integrity and Minimal Metadata Leakage

Mike Schliep
University of Minnesota
schli116@umn.edu

Nicholas Hopper
University of Minnesota
hoppernj@umn.edu

ABSTRACT

Existing End-to-End secure messaging applications trust a single service provider to deliver messages in a consistent *order* to a consistent *group* of conversation members. We propose a protocol that removes this single point of failure by using multiple service providers, enforcing conversation integrity as long as one service provider out of N behave honestly. However, this approach could potentially increase the number of entities that learn the metadata for a conversation. In this work we discuss the challenges and provide a protocol that limits the metadata leakage to that of existing messaging applications while still providing strong conversation integrity.

KEYWORDS

End-to-End Encryption, Secure Messaging, Cryptographic Protocol

1 INTRODUCTION

Texting and social media-based messaging applications have become prolific as a means of every day communication. The popularity of these messaging applications stems in part from their convenience, allowing users to communicate even in a mobile and asynchronous setting, where their network availability may be unreliable and they may come online and go offline at different times. Due to increasing privacy concerns of users, some of the most widely deployed messaging applications, including WhatsApp [11], Facebook [5], and Signal [9], have been deploying end-to-end encryption to protect the confidentiality and integrity of messages in users' conversations.

Message confidentiality and integrity are not sufficient to protect a conversation. While current applications protect the integrity of individual *messages* — an adversary cannot modify a message while in transit from Alice to Bob — they do not protect the integrity of the *conversation*. Consider the following conversation between Alice and Bob, in which the order that messages are displayed can drastically affect the meaning of the conversation, even if the individual messages cannot be modified:

Alice's View:

> **Alice:** Are you going to the protests?
> **Alice:** Have you had lunch yet?
> **Bob:** No... Yes.

Bob's View:

> **Alice:** Have you had lunch yet?
> **Alice:** Are you going to the protests?
> **Bob:** No... Yes.

We refer to the security property that a conversation must be displayed consistently to all participants as *conversation integrity*. This is an example of an additional security property we deem necessary for any future protocols to achieve end-to-end secure messaging.

To support conversation progress while some users are offline the service provider handles routing and caching messages in the conversation. This service provider can break the conversation integrity property of a protocol that allows conversation to progress while some participants are offline. The service provider simply needs to fork the conversation after a target message and can partition the group into multiple views of the same conversation. We illustrate this with an example. Consider a conversation between Alice, Bob, Charlie, and Dave. The service provider forks the conversation after Alice's second message. The group is partitioned into two views, one where Alice and Bob believe they are the only participants online and the other where Charlie and Dave believe they are the only participants online.

Alice's and Bob's View:

> **Alice:** Lets go to the protest if 3 people want to?
> **Alice:** I want to go.
> **Bob:** I cannot make it.

Charlie's and Dave's View:

> **Alice:** Lets go to the protest if 3 people want to?
> **Alice:** I want to go.
> **Charlie:** I am in.
> **Dave:** Yes, me too.

Another security property we believe is importation for secure messaging is deniability. Consider the following conversation:

> **Report:** What is your company doing illegally?
> **Whistleblower:** They are dumping poison into the water.

Message deniability guarantees there is no cryptographic proof that the whistleblower authored the message. Now consider the following conversation:

> **Whistleblower:** My SSN is 123-45-6789.
> **Report:** What is your company doing illegally?
> **Whistleblower:** They are dumping poison into the water.

A protocol that provides message deniability allows the whistleblower to argue that they did not author the messages. But only the whistleblower knows their social security number so a protocol must also provide message unlinkability, guaranteeing there is

CCS '18, October 15–19, 2018, Toronto, ON, Canada

ACM ISBN 978-1-4503-5693-0/18/10.
https://doi.org/10.1145/3243734.3278506

no cryptographic proof to a third party that both messages were authored by the same participant.

Most deployed secure messaging applications are based on the Signal two-party protocol, which is non-trivial to extend to group settings. Recently, multiple vulnerabilities [6, 7] have been discovered in the way these applications implement end-to-end secure messaging for groups. These vulnerabilities allow an adversary to drop or reorder messages in two-party and group conversations. Other messaging applications ignore end-to-end security of group conversations entirely. We consider group conversations just as important as two-party conversations and future protocols must be designed with that in mind.

On the other hand, recent literate literature [2–4, 8] proposing secure messaging protocols make assumptions that are not realistic in the modern mobile internet which makes them not practical for real world deployments. Most of these works require synchronous communication or provide little to no guarantees about conversation integrity.

We propose a new protocol (GrOMPI) that provides end-to-end secure messaging in the mobile model with conversation integrity and does not leak metadata to any additional entities compared to existing applications.

2 SECURITY GOALS

We quickly discuss the threat model and security goals of end-to-end secure mobile messaging.

2.1 Threat Model

The security provided by GrOMPI needs to withstand strong adversaries. We must consider an adversary that may compromise multiple servers and multiple users long-term and ephemeral keys. The adversary also has full network control and may drop, modify, and reorder the network traffic.

2.2 Security Properties

We now discuss the security goals of GrOMPI.

- **Message Confidentiality**: Only conversation participants can read a message.
- **Message Integrity**: Messages are guaranteed to not have been modified in transit.
- **Message Authentication**: Conversation participants can verify the author of a message.
- **Forward Secrecy**: Past messages are confidential even if future key material is revealed.
- **Backward Secrecy**: Future messages are confidential if past key material is revealed, also know as future secrecy or post-compromise secrecy.
- **Participant Authentication**: Participants can verify other participants are really who they claim to be.
- **Participant Consistency**: All participants of a conversation agree on all the participants of the conversation.
- **Conversation Integrity**: All participants see the same conversation. This includes the order of messages in a conversation and the order of participant changes in a conversation.
- **Deniability**: Participants must be able to deny taking part in a conversation. Unger et. al. [10] refer to this as participant repudiation. They also discuss two additional deniability properties; message repudiation and message unlinkability. Participant repudiation implies message repudiation. Message unlinkability is the property that if a distinguisher can be convinced a user authored one message this should not prove the authorship of any other message. Off-the-Record messaging should provide all of these forms of deniability.
- **Metadata Privacy**: Minimal metadata is leaked to the fewest entities that are not participants in the conversation.
- **Anonymity Preserving**: The protocol should not undermine the anonymity features of the underlying transport.
- **Computation and Trust Equality**: All users perform similar computations and no user is trusted more than any other.
- **Untrusted Service Provider**: Any single service provider is not trusted to provide any of the security properties.
- **Dynamic Groups**: Participants can be added and removed from conversations without restarting the protocol.

3 GROMPI: IN SUBMISSION

At a high level GrOMPI is designed as follows. To avoid the conversation partitioning attack discussed previously GrOMPI depends on a routing server used to route messages and multiple mirrors to enforce a consistent order. Users register with a routing server out-of-band. This registration links a user identity, a long-term public key, and multiple single use pre-keys. The routing server shares this registration information with the mirrors. When messages are sent as part of a conversation they are uploaded to the routing server and distributed to the mirrors. The server and mirrors then send the messages to the participants. The participants do not process a message until it has been received from the server and every mirror. As long as a single server or mirror is honest conversation integrity and participant consistency is enforced.

There are four types of messages; Setup, Receipt, Message, and Update. The Setup message indicates a new conversation. Receipt indicates a users has processed all previous Setup, Message, and Update messages. Message is used to send a conversation message to display to users. And Update is used to update the set of participants in a conversation. The rules of who is authorized to make participant changes as well as what kind of changes they are allowed to make are left up to the implementation but must be enforceable by the servers.

When Alice wishes to send a message in a conversation with Bob and Charlie she first fetches Bob's and Charlie's long-term public keys and a single use public pre-key for each of Bob and Charlie. She then uploads the message of the form:

$$Sid, \text{“}TYPE\text{”}, Alice, c, c_{ab}, c_{ac}, auth_{as_1}, \ldots, auth_{as_m}$$

to the routing server. Sid is a unique session identifier for the session. TYPE is the type of message, c is the message ciphertext or set of participants. c_{a*} are per-user ciphertext blocks that authenticate the message and contain key ratchet material for Bob and Charlie. The $auth_{a*}$ blocks are authentication blocks allowing the servers to authenticate the entire message from Alice.

The servers then distribute the following to Alice, Bob and Charlie respectively.

$$Sid, \text{“}TYPE\text{”}, Alice, c$$
$$Sid, \text{“}TYPE\text{”}, Alice, c, c_{ab}$$
$$Sid, \text{“}TYPE\text{”}, Alice, c, c_{ac}$$

All messages in a conversation are processed in this manner. The pairwise ciphertext blocks provide message authentication and integrity. The pairwise ciphertext blocks use a protocol that provides forward and backward secrecy as well as deniability, similar to OTR or Signal. As long as a single mirror is honest the protocol provides participant consistency and conversation integrity. Participant authentication is provided by verifying the long-term keys out-of-band in the same manner as existing secure messaging applications. Finally, GrOMPI does not compromise the anonymity of any network protocol by using anonymous Diffie-Hellman to encapsulate setting up the client secure connections to the server.

4 METADATA PROTECTION: WORK IN PROGRESS

The disadvantage of GrOMPI compared to other secure messaging protocols is that with GrOMPI more servers learn the metadata of a conversation. This is necessary for the mirrors to send the messages to the clients. We are working to reduce the metadata leaked to the mirrors. We achieve this by removing the mirrors and replacing them with order enforcing servers.

When a client registers pre-keys with the routing server, the client also registers conversation order pre-keys that are associated with an ephemeral client identity. Now, when a conversation is setup the routing server chooses a conversation order pre-key and sends it to the order enforcing servers. Then the order enforcing server can derive a symmetric key with the ephemeral client identity and produce a Message Authentication Code (MAC) associating the hash of the setup message as the first message in a conversation. The MAC is returned to the routing server and delivered to the client through the routing server. This way the order enforcing servers do not learn who participated in a conversation. The order of all conversation messages are enforced this way with the MAC associating the hash of the message with the index of the message in the conversation.

This approach still reveals how many participants are in a conversation and how often messages are sent. To avoid this we pad all conversations to a constant number of participants and the routing server can send noise messages in a conversation. This does increase the size of setup and participant change messages and also increases the number of messages a client must process but significantly reduces the amount of metadata leaked to the order enforcing servers. Finally, all conversation order key material must be replaced when the participants of a conversation are changed. This avoids leaking the number of users added or removed.

In addition to protecting the metadata of a conversation this also eases deployment of the protocol. With this new model only the routing server must be highly available. As long as a majority of the order enforcing servers are honest and available a conversation can continue to progresses even if some order enforcing servers are offline.

5 RELATED WORK

Off-The-Recorcd (OTR) [2] is the first academic work to look at providing private instant messaging. OTR provides message confidently, integrity, authentication, repudiation, and unlinkability. However OTR does not provide participant repudiation or conversation integrity. The main limitation of OTR is it only supports conversations between two individuals. There is not a straight forward mechanism to apply OTR in a group setting.

Multiparty OTR (mpOTR) [3] tries to provide the properties of OTR for group conversations. However, it only achieves conversation integrity after a conversation has ended by executing a byzantine agreement of the completed transcript. Group Off-The-Recored (GOTR)[4] and SYM-GOTR [8] provide the properties of OTR for group conversations with conversation integrity but require all users to be online at the same time to make conversation progress. They achieve conversation integrity by performing a consistency check after every message.

Signal [9] formerly TextSecure is the most widely deployed protocol for secure mobile messaging. Recently multiple vulnerabilities [6, 7] have been discovered affecting participant consistency and conversation integrity vulnerabilities in two-party and group conversations. Signal lacks conversation integrity of messages and receipts, Charlie can not verify if Alice has received Bob's message and no order of messages is enforced.

Recently the Internet Engineering Task Force initiated a working group to develop a standard for Messaging Layer Security (MLS) [1]. The working group has focused on developing a group key agreement protocol that can be setup and ratcheted while some users are offline. They do not provide a mechanism for message authorship authentication, deniability, or message ordering.

6 ACKNOWLEDGMENTS

This work was sponsored by the National Science Foundation under grant 1814753.

REFERENCES

[1] Richard Barnes, Jon Millican, Emad Omara, Katriel Cohn-Gordon, and Raphael Robert. 2018. *Messaging Layer (MLS) Secuirty.* https://tools.ietf.org/html/draft-barnes-mls-protocol-01.
[2] Nikita Borisov, Ian Goldberg, and Eric Brewer. 2004. Off-the-record communication, or, why not to use PGP. In *Proceedings of the 2004 ACM workshop on Privacy in the electronic society.* ACM, 77–84.
[3] Ian Goldberg, Berkant Ustaoğlu, Matthew D Van Gundy, and Hao Chen. 2009. Multi-party off-the-record messaging. In *Proceedings of the 16th ACM conference on Computer and communications security.* ACM, 358–368.
[4] Hong Liu, Eugene Y Vasserman, and Nicholas Hopper. 2013. Improved group off-the-record messaging. In *Proceedings of the 12th ACM workshop on Workshop on privacy in the electronic society.* ACM, 249–254.
[5] Moxie Marlinspike. 2016. Facebook Messenger deploys Signal Protocol for end to end encryption. https://whispersystems.org/blog/facebook-messenger/
[6] Paul Rösler, Christian Mainka, and Jörg Schwenk. 2018. More is Less: On the End-to-End Security of Group Chats in Signal, WhatsApp, and Threema. (2018).
[7] Michael Schliep, Ian Kariniemi, and Nicholas Hopper. 2017. Is Bob Sending Mixed Signals?. In *Proceedings of the 2017 on Workshop on Privacy in the Electronic Society.* ACM, 31–40.
[8] Michael Schliep, Eugene Vasserman, and Nicholas Hopper. 2018. Consistent Synchronous Group Off-The-Record Messaging with SYM-GOTR. *Proceedings on Privacy Enhancing Technologies* 2018, 3 (2018), 181–202.
[9] Open Whisper Systems. [n. d.]. *Open Whisper Systems.* https://whispersystems.org/.
[10] Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. 2015. SoK: Secure Messaging. In *Security and Privacy (SP), 2015 IEEE Symposium on.* IEEE, 232–249.
[11] WhatsApp. 2017. https://www.whatsapp.com/security