Security Analysis and Improvements for the IETF MLS Standard for Group Messaging

Joël Alwen
Wickr Inc.
jalwen@wickr.com

Sandro Coretti
IOHK
sandro.coretti@iohk.io

Yevgeniy Dodis
New York University
dodis@cs.nyu.edu

Yiannis Tselekounis
New York University
tselekounis@cs.nyu.edu

Speaker: Joël Alwen

Abstract

Secure Messaging. End-to-end Secure Messaging (SM) allows people to exchange messages without compromising their authenticity and privacy. In contrast to common secure communication protocols such as TLS and SSH, SM protocols are designed for settings in which messages are exchanged *asynchronously* and in which sessions exist for long periods of time. Consequently, participants can be offline at times and their state is more likely to be exposed at some point during the lifetime of a session. SM protocols are therefore expected to satisfy so-called *forward secrecy* and *post-compromise security (PCS)* (aka backward secrecy). The former means that even when a participant's key material is compromised, past messages (delivered before the compromise) remain secure. Conversely, PCS means that once the compromise ends, the participants will eventually recover full security as a side effect of continued normal protocol usage.

The rigorous design and analysis of *two-party* SM protocols has received considerable attention in recent years. This is, in no small part, due to advent of the Double Ratchet paradigm. Forming the cryptographic core of a slew of popular messaging applications (e.g., Signal, who first introduced it, as well as WhatsApp, Facebook Messenger, Skype, Google Allo, Wire and more); Double Ratchet protocols are now regularly used by over a billion people worldwide.

Unfortunately, it turns out that the Double Ratchet paradigm doesn't scale well in the number of users. So we are left with the natural question of how to build secure *group* messaging protocols (SGM) that enjoy similar security properties to the two-party ones but whose efficiency scales (say) logarithmically in the group size.

Forward secrecy of TreeKEM. In order to address the lack of satisfactory SGM protocols, the IETF has launched the message-layer security (MLS) working group, which aims to standardize an eponymous SGM protocol. Following in the footsteps of the Double Ratchet, the MLS protocol promises to be widely deployed and heavily used. Indeed, the working group already includes various messaging companies (Facebook, Google, Cisco, Wickr, Wire, Twitter, etc.) whose combined messaging user base includes everything from government agencies to companies both large and small; not to mention a major chunk of the world's consumer population.

The heart of the MLS protocol is the TreeKEM continuous key-agreement protocol. Not only is it the most novel and intricate part of the MLS draft, understanding TreeKEM is also central to understanding the security and efficiency properties of MLS itself.

In our work, we analyze (and improve on) the latest version of TreeKEM. On a positive note, we show rigorously that TreeKEM indeed achieves PCS (in isolation). However, we also observe that there are serious issues with TreeKEM's forward secrecy, stemming from the fact that its users do not erase old keys sufficiently fast.

To better understand the problem, recall that TreeKEM aims to achieve forward secrecy and PCS by having members of a messaging group periodically perform *update* operations. Whenever a party performs an update, all group members obtain a fresh so-called *update secret* I. In order to efficiently perform updates (with logarithmic packet size), TreeKEM arranges all members in a binary tree and uses public-key encryption (PKE) to encrypt information about I to specific subsets of members (determined by their position in the tree). After processing the update, however, parties do not erase or modify the PKE secret keys used to decrypt the update information, since they might need them to process future updates. Hence, corrupting any party other than the update initiator will completely reveal I to an attacker, thereby violating forward secrecy. In fact, in order for I to remain secret upon state compromise, logarithmically many additional updates are required in the best case, and linearly many in the worst case (depending on the order of updates). Even worse, unless the sibling (in the tree) of I's initiator performs an update, I is never forward-secret. And in a fully asynchronous setting, requiring members to update their state frequently might not be easily enforceable.

Fixing TreeKEM. In order to remedy TreeKEM's issues with forward secrecy, we devise a new type of public-key encryption (PKE) (based on work by Jost *et al.* [2] and suggestions by Konrad Kohbrok on the MLS mailing list [3]) and show that using it in lieu of the (standard) PKE within TreeKEM results in a protocol with *optimal* forward secrecy. Specifically, with the new flavor of PKE, *public and secret keys suitably change with every encryption and decryption*, respectively. This kind of key evolution ensures that after decryption, the (evolved) secret key leaks no information about the original message, thereby thwarting the above attack. We also provide a very efficient instantiation of the new PKE notion, thereby ending up with a practical fix and *going from very loose to optimal security at negligible cost.*

Exact security of TreeKEM and continuous group key agreement. To precisely capture the security properties of the original and modified TreeKEM protocols, we first introduce a new primitive called *continuous group key agreement (CGKA)* along with corresponding security notions. In particular, the security definitions include PCS, and a flexible range for forward secrecy, from no forward secrecy to optimal forward secrecy. We then proceed to characterize *precisely* the limited type of forward secrecy achieved by the current TreeKEM (which includes PCS, meaning TreeKEM is at least backward-secure), and also show that our modified version of TreeKEM is optimally forward secret.

The concept of CGKA distills out the key agreement component of MLS (and potentially of other group messaging protocols) and allows it to be analyzed in isolation. To illustrate that CGKA is a useful abstraction, we sketch how any CGKA scheme can be combined with symmetric primitives to obtain a full-fledged secure messaging scheme inheriting the CGKA efficiency properties. In particular, this modular design allows CGKA to be analyzed in a setting with authenticated channels, which greatly simplifies the security proofs. Furthermore, it allows for a better understanding of the full MLS protocol in terms of simpler building blocks.

Adaptive security. Finally, we assess the security TreeKEM in the face of an *adaptive* attacker, i.e., an attacker that may corrupt group members on-the-fly (as opposed to at the beginning). This very important setting turns out to be quite intricate to analyze. The naïve methods to extend our non-adaptive security proofs leads to exponentially loose security. Fortunately, using a very clever reduction technique of Jafargholi *et al.* [1], we end up getting substantially better and closer to practically usable security bounds for this important setting.

References

[1] Zahra Jafargholi, Chethan Kamath, Karen Klein, Ilan Komargodski, Krzysztof Pietrzak, and Daniel Wichs. Be adaptive, avoid overcommitting. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part I*, volume 10401 of *Lecture Notes in Computer Science*, pages 133–163. Springer, 2017.

[2] Daniel Jost, Ueli Maurer, and Marta Mularczyk. Efficient ratcheting: Almost-optimal guarantees for secure messaging. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 159–188. Springer, 2019.

[3] Konrad Kohbrok. Subject: [MLS] Improve FS granularity at a cost. MLS Mailing List 24, January 2019 09:51 UTC, 2019. `https://mailarchive.ietf.org/arch/msg/mls/WRdXVr8iUwibaQu0tH6sDnqU1no`.