

Secure Multiparty Communication Systems

Finn Paul Lippok

Seminar on Internet Technology

Chair of Communication and Distributed Systems

RWTH Aachen University

Aachen, Germany

finn.lippok@rwth-aachen.de

Abstract—Placeholder.

Index Terms—Secure Multiparty Communication

I. INTRODUCTION

- Motivation: groups are harder than pairs.
- Which techniques and protocols are most relevant in current systems? → Section III
- How practical are these to deploy? → Section IV
- How to protect against misbehaving peers? → Section V

II. BACKGROUND

A. Threat Model

- Network adversary: passive, active
- Malicious server, semi-trusted server, DS
- Member: state compromise, malicious insider.

B. Security Goals

In this section, we provide clear definitions for the security notions used in this paper. Group membership and metadata privacy are out of scope.

1) *Confidentiality*: Sent group messages should only be decryptable by current members of the group. In particular, this means that new group members cannot decrypt messages sent before they joined and can no longer decrypt messages sent after they left. In the group key management literature, these two properties have been called Backward and Forward Secrecy, respectively [1], [2]. We instead use Forward Secrecy in the sense of the secure-messaging literature [3], [4] (see Section II-B2).

2) *Forward Secrecy*: Forward Secrecy (FS) is met if, after a member's state is compromised, all old, already decrypted messages stay secure [3], [4].

3) *Post-Compromise Security*: Post-Compromise Security (PCS) is met if, after a member's state is compromised by a passive attacker, all messages sent in the future are secure [4], [5].

4) *Group Membership Authentication*:

5) *Sender Authentication*:

6) *Consistency*: Group members should eventually agree on the group membership and group key, regardless of the order in which messages arrive [4]. Thus, it should not be possible for a single member or the DS to permanently split the group [3], [5].

7) *Availability*: We assume that all messages are eventually delivered, because an active network adversary or a DS can always drop them [5]. Neither a misbehaving insider nor the DS should be able to disrupt honest group members' message processing or key agreement [6], [7].

C. Two-Party End-to-End Encryption

The most widely deployed protocol for two-party messaging is the Signal Protocol [8], which uses PQXDH [9] for initial key agreement and the Double Ratchet Algorithm [10]. In the following, we outline how the protocol works and how it provides FS and PCS (called break-in recovery in the specification [10]). For simplicity, we omit key authentication and the recently added post-quantum ratchet [10].

1) *Setup*: We assume Alice wants to initiate a conversation with Bob while Bob is offline. Bob has previously published a prekey bundle to a server, which contains Bob's public key. Alice fetches it and uses the PQXDH key agreement protocol [9] to derive a root key. In the first message, Alice includes the values Bob needs to derive the same root key.

2) *Symmetric-key ratchet*: Alice and Bob use a KDF to further derive sending and receiving chain keys. For each message they send or receive, they derive a one-time message key, used to encrypt or decrypt the message, and a new sending or receiving chain key, respectively. Because the KDF is one-way and all old keys are deleted after use, an attacker that compromises the state of either or both parties cannot compute previous chain keys and thus cannot derive the keys used to encrypt old messages. This provides FS. PCS, on the other hand, is not provided because an attacker can advance the chain exactly as the legitimate parties do and thus derive all future keys [10].

3) *Diffie-Hellman ratchet*: To provide PCS, new randomness needs to be mixed into the keys. This is achieved by the DH ratchet. Whenever the sending direction changes, the new sender includes a new ratchet public key in their first message. From that and the current root key, a new shared root key and chain keys can be derived for the following messages [10].

D. Client-Side Fanout

The naive approach to scale the existing Signal Protocol to group chats would be client-side fanout. By creating pairwise channels between all group members, a de facto group chat is created. For a group of size n , each individual is required

to create $n - 1$ channels and every sent message needs to be encrypted $n - 1$ times [11]. Because every message is sent over such pairwise channels, client-side fanout inherits their FS and PCS properties [11], [12]. On the other hand, because every message is encrypted independently, malicious group members could send completely different messages to different members of the group without being caught [13]. Detecting such misbehaving peers would require additions to the protocol, like making all parties share the hash of their transcripts with each other to catch any discrepancies, as described by Goldberg et al. [13].

III. TECHNIQUES IN CURRENT SYSTEMS

In this section, we will discuss protocols used in current systems, their current shortcomings, and proposed improved variants. We will first discuss the currently most widely adopted approach, Sender Keys, and then go into more detail on the current research direction of Continuous Group-Key Agreement (CGKA), with a focus on tree-based approaches, while also briefly discussing other approaches.

A. Sender Keys

Sender Keys are currently adopted by many group messaging platforms, most notably WhatsApp and Signal [8]. We will first discuss their functionality as formally described by Balbás et al. [8] and then ...Like client-side fanout, Sender Keys rely on pairwise channels between group members, but unlike client-side fanout, they do not exchange messages over these channels directly [8]. Instead, a group member i who wants to send a message must have previously sent their sender key to every other group member over such a two-party channel. Such a sender key consists of a symmetric chain key ck_i used to derive message keys and a signing public key pk_i , while the corresponding signing sk_i remains secret to member i . When sending a message, it is encrypted with a message key derived from the current ck_i and signed with sk_i . All group members receive the same ciphertext, verify its signature using pk_i and decrypt it by also deriving the same current message key using their copy of ck_i . After sending and receiving, respectively, each party performs one ratchet step on ck_i using an KDF.

Because Sender Keys only use a symmetric chain ck_i and no second ratchet, PCS can only be achieved by redistributing new sender keys and thus new chain keys. Because this is implementation-dependent and often only done after a group member leaves, PCS is only provided over long periods of time [8]. If an attacker also compromises the state of the pairwise channels and the new keys are distributed before these channels have fully healed, the newly distributed sender keys could also leak [8].

- Similar: Megolm (Matrix) [8]
- complexity

B. Continuous Group-Key Agreement

Well-designed CGKAs improve upon the weakness of Sender Keys by no longer requiring keys to be shared ... with every group member ...

1) *Definitions:* Epochs, commits, Operations

2) *Tree Based:*

- Predecessors (Not Strictly CGKA)
 - LKH: Server holds all keys
 - TGDH: not asynchronous
- ART [11]
- TreeKEM/MLS [3] (used by RCS, Discord VC)
 - Improved variants:
 - * RTreeKEM: updates private keys (UPKE) [14]
 - * Tainted TreeKEM: avoids blanks [15]

3) *Other Approaches:*

- Chained CmpKE [16]
- DCGKA: decentralized (no ordering server) [4]
- DeCAF [17]

IV. PRACTICALITY OF DEPLOYMENT

A. Complexity

- see Table I
- Best vs. worst case
- Bandwidth asymmetry [16]

B. Other Challenges

- Post-Quantum
- Forks and state loss
- Interoperability (EU DMA)
- Dependence on the ordering server.

V. PROTECTING AGAINST MISBEHAVING PEERS

- What insiders cannot be prevented from doing
- Inconsistent commits → split group. (prevent with Zero-Knowledge Proofs [6])
- Insider DoS: abort vs. guaranteed output delivery [18]
- Cheater identification

VI. CONCLUSION

REFERENCES

- [1] Y. Kim, A. Perrig, and G. Tsudik, "Tree-based group key agreement," *ACM Transactions on Information and System Security*, vol. 7, no. 1, pp. 60–96, Feb. 2004.
- [2] T. Prantl, T. Zeck, A. Bauer, P. Ten, D. Prantl, A. E. B. Yahya, L. Ifflaender, A. Dmitrienko, C. Krupitzer, and S. Kounev, "A Survey on Secure Group Communication Schemes With Focus on IoT Communication," *IEEE Access*, vol. 10, pp. 99 944–99 962, 2022.
- [3] R. Barnes, B. Beurdouche, R. Robert, J. Millican, E. Omara, and K. Cohn-Gordon, "The Messaging Layer Security (MLS) Protocol," RFC Editor, Tech. Rep. RFC9420, Jul. 2023.
- [4] M. Weidner, M. Kleppmann, D. Hugenroth, and A. R. Beresford, "Key Agreement for Decentralized Secure Group Messaging with Strong Security Guarantees," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 2024–2045.
- [5] B. Beurdouche, E. Rescorla, E. Omara, S. Inguva, and A. Duric, "The Messaging Layer Security (MLS) Architecture," RFC Editor, Tech. Rep. RFC9750, Apr. 2025.
- [6] J. Alwen, S. Coretti, D. Jost, and M. Mularczyk, "Continuous Group Key Agreement with Active Security," in *Theory of Cryptography*, R. Pass and K. Pietrzak, Eds. Cham: Springer International Publishing, 2020, vol. 12551, pp. 261–290.

TABLE I
COMPARISON

Scheme	FS	PCS	Asynchronous	Ordering requirement	Add	Remove	Send	Receive	Update
Fanout									
Sender Keys									
ART									
TreeKEM									
RTreeKEM									
Tainted TreeKEM									
CmPKE									
DCGKA									

- [7] J. Devigne, C. Duguey, and P.-A. Fouque, “MLS Group Messaging: How Zero-Knowledge Can Secure Updates,” in *Computer Security – ESORICS 2021*, E. Bertino, H. Shulman, and M. Waidner, Eds. Cham: Springer International Publishing, 2021, vol. 12973, pp. 587–607.
- [8] D. Balbás, D. Collins, and P. Gajland, “WhatsApp with Sender Keys? Analysis, Improvements and Security Proofs,” in *Advances in Cryptology – ASIACRYPT 2023*, J. Guo and R. Steinfeld, Eds. Singapore: Springer Nature Singapore, 2023, vol. 14442, pp. 307–341.
- [9] E. Kret and R. Schmidt, “The PQXDH Key Agreement Protocol,” Signal, Specification Revision 3, May 2023.
- [10] T. Perrin, M. Marlinspike, and R. Schmidt, “The Double Ratchet Algorithm,” Signal, Specification Revision 4, Nov. 2025.
- [11] K. Cohn-Gordon, C. Cremers, L. Garratt, J. Millican, and K. 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*. Toronto Canada: ACM, Oct. 2018, pp. 1802–1819.
- [12] P. Rösler, C. Mainka, and J. 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)*. London: IEEE, Apr. 2018, pp. 415–429.
- [13] I. Goldberg, B. Ustaoglu, M. D. Van Gundy, and H. Chen, “Multi-party off-the-record messaging,” in *Proceedings of the 16th ACM Conference on Computer and Communications Security*. Chicago Illinois USA: ACM, Nov. 2009, pp. 358–368.
- [14] J. Alwen, S. Coretti, Y. Dodis, and Y. Tselekounis, “Security Analysis and Improvements for the IETF MLS Standard for Group Messaging,” in *Advances in Cryptology – CRYPTO 2020*, D. Micciancio and T. Ristenpart, Eds. Cham: Springer International Publishing, 2020, vol. 12170, pp. 248–277.
- [15] K. Klein, G. Pascual-Perez, M. Walter, C. Kamath, M. Capretto, M. Cueto, I. Markov, M. Yeo, J. Alwen, and K. Pietrzak, “Keep the Dirt: Tainted TreeKEM, Adaptively and Actively Secure Continuous Group Key Agreement,” in *2021 IEEE Symposium on Security and Privacy (SP)*. San Francisco, CA, USA: IEEE, May 2021, pp. 268–284.
- [16] K. Hashimoto, S. Katsumata, E. Postlethwaite, T. Prest, and B. West-erbaan, “A Concrete Treatment of Efficient Continuous Group Key Agreement via Multi-Recipient PKEs,” in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*. Virtual Event Republic of Korea: ACM, Nov. 2021, pp. 1441–1462.
- [17] J. Alwen, B. Auerbach, M. Cueto Noval, K. Klein, G. Pascual-Perez, and K. Pietrzak, “DeCAF: Decentralizable CGKA with Fast Healing,” in *Security and Cryptography for Networks*, C. Galdi and D. H. Phan, Eds. Cham: Springer Nature Switzerland, 2024, vol. 14974, pp. 294–313.
- [18] A. Dalskov, D. Escudero, and A. Nof, “Fast Fully Secure Multi-Party Computation over Any Ring with Two-Thirds Honest Majority,” in *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. Los Angeles CA USA: ACM, Nov. 2022, pp. 653–666.