Revisiting Post-Compromise Security Guarantees in Group Messaging

Extended Abstract

May 9, 2019

Cas Cremers[1], Britta Hale[2⋆], and Konrad Kohbrok[3]

[1] CISPA Helmholtz Center for Information Security
cremers@cispa.saarland
[2] Naval Postgraduate School (NPS)
britta.hale@nps.edu
[3] Aalto University
konrad.kohbrok@aalto.fi

Abstract Modern secure messaging protocols such as Signal can offer strong security guarantees, in particular Post-Compromise Security (PCS) [6]. The core PCS mechanism in these protocols is inherently pairwise, which causes bad scaling behaviour and makes PCS inefficient for large groups. To address this, two recently proposed designs for secure group messaging, ART [4] and MLS Draft-04 [1], use group keys derived from tree structures to efficiently enable PCS mechanisms in large groups.

In this work we highlight a previously unexplored difference between the pairwise and group-key based approaches. We show that without additional mechanisms, both ART and MLS Draft-04 offer significantly lower PCS guarantees than those offered by groups based on pairwise PCS channels. In particular, for MLS Draft-04, it seems that the protocol does not yet meet the informal PCS security guarantees described in the draft.

We explore the causes of this problem and lay out the design space to identify solutions. Optimizing security and minimizing overhead leads us to a promising solution based on (i) global updates and (ii) *post-compromise secure signatures*. While rotating signatures had been discussed before as options for both MLS Draft-04 and ART, our work indicates that combining specific update patterns for all groups with a post-compromise secure signature scheme, may be strictly necessary to achieve any reasonable PCS guarantee.

Keywords: post-compromise security · forward secrecy · group messaging protocols · message-layer security

1 Introduction

Modern secure messaging applications, notably those built on top of the Signal protocol libraries [7], can offer very strong security guarantees, among them Post Compromise Security (PCS) [6]. At its core, PCS describes a protocol's ability to "self-heal" after a compromise. Consider the following case: a communicating pair of agents continuously exchange Diffie-Hellman keying material which is used to update a shared symmetric key. If an adversary learns the full state of an identity (or identities) it can impersonate them, but if the compromised parties manage to transfer a single new DH share to the other identity without the adversary interfering, the next symmetric keys will again be unknown to the adversary, and the session is considered "healed". In other words, if an adversary wants to eavesdrop or impersonate a compromised user at some future time, they will have to actively interfere in all of the user's communications until that time. This makes compromising a user's state a substantially less attractive attack vector.

However, the design of the Signal protocol is inherently pairwise. For group messaging, one can get similar guarantees by setting up pairwise PCS channels between all participants. This leads to a

⋆ The views expressed in this document are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

complete communication graph and therefore does not scale effectively. In particular, to send a message m to a group of N users, m must be encrypted under N different keys, scaling the communication and computation overhead for each user linearly with respect to N. Newer designs for group messaging aim to address this, notably ART [4] and MLS Draft-04 [1], which aim to efficiently offer strong PCS guarantees for large groups in addition to scaling well. To this end they use tree structures to compute group keys, which can be updated by individual group members.

In this work we revisit the PCS guarantees offered by the ART protocol and MLS Draft-04. We show that without additional mechanisms, they provide significantly weaker PCS guarantees than group messaging based on pairwise PCS channels. Our observations originate in the difference in the scope of updates: while this difference has been noted before, its implications have not been fully explored.

We explore the design space for possible solutions, isolate a candidate solution and discuss potential pitfalls and open questions. Our candidate solution revolves around two main observations: (i) the need for concurrent group updates and (ii) PCS guarantees from the authentication layer. We note that until now, the authentication layer had been left underspecified by ART and MLS, but is in fact critical for achieving stronger PCS guarantees.

We note that while both ART and MLS Draft-04 provide core building blocks for strongly secure group messaging, they do not directly achieve their informal PCS claims. We sketch how our candidate solution can be integrated with the current MLS Draft-04 design.

Our goal is to raise issues early and to sketch the way forward for achieving PCS in group messaging, but we note that this version of our work does not yet offer a full-fledged solution and formal analysis.

2 Background

In this section, we explore previous work and introduce PCS.

2.1 Post-Compromise Security

Post-Compromise Security (PCS) was introduced in [6] and refers to the security guarantees that can be expected from the communication of an identity *after* a compromise. While some designs have previously informally offered this type of guarantee, it has only been formally studied since recent works such as [3, 6].

PCS is often viewed as the counterpart to forward secrecy (FS), which is concerned with the secrecy of communication *before* the event of compromise. However, where FS for the most part is concerned with the protection of confidentiality of past communication, for PCS both authentication and confidentiality are relevant security properties.

Generally, PCS requires an "update" operation, in which an identity combines fresh key material with previously used key material and shares it with the partner. If the adversary is passive during one update, the identity is "healed" again.

2.2 Group Messaging in Signal

The core Signal protocol is a pairwise protocol [7], in which two participants continuously send each other asymmetric ratchet updates. In practice, these are ephemeral public keys of the form g^z that get combined with previous keying material using Diffie–Hellman constructions, to finally derive symmetric keys used for message transmission. If an adversary learns the state of an identity A, but the real A afterwards manages to transmit a new ephemeral public key to the peer B, then the adversary is locked out again of subsequent communications since all future keys will depend on that update.

Group messaging in the Signal protocol is done through one of two mechanisms. The first is to implement groups using the core pairwise protocol, i.e. point-to-point pairwise connections forming a complete graph over the set of group members, and the second is an alternative mechanism called *sender keys*. We summarize them in turn.

Groups Using Only Pairwise Channels. Signal groups can be implemented using only pairwise channels. In this case, a group does not correspond to a specific cryptographic mode, but is essentially a wrapper for sending each message to all group participants over individual pairwise channels. Thus, to send a message in the group $\{A, B, C, D\}$, A send the message over the three pairwise channels $A \leftrightarrow B$,

$A \leftrightarrow C$, and $A \leftrightarrow D$. Performing an asymmetric update in the group corresponds to three individual updates on the respective pairwise channels.

Groups Using Sender Keys. To improve the scaling behaviour for larger groups, Signal offers another mode called *sender keys*. In this case, when starting a group, each identity $\mathcal{I}_i$ generates their own "sender" key $k_{\mathcal{I}_i}$, which is a symmetric key that is used in a one-to-many fashion within the context of the group. $\mathcal{I}_i$ then transmits this sender key $k_{\mathcal{I}_i}$ to the other group members over the pairwise channels. When $\mathcal{I}_i$ wants to send a message to the group, they encrypt it under their own sender key and broadcast the result to the group. The other parties use the sender's key to decrypt the message. An identity will generate a sender key for each group they are in. This mechanism is very efficient for large groups, but does not provide PCS by itself: if the adversary learns the state of $\mathcal{I}_i$, they learn the symmetric sender keys of that identity and its peers, and all future messages can be decrypted or forged.

One can reintroduce some notion of PCS on top of this mechanism by frequently generating new sender keys, and sending them over the pairwise channels and updating those with asymmetric keys at the same time, at the cost of efficiency.

2.3 Group Messaging in ART

ART was the first attempt to provide a group messaging scheme that efficiently provides PCS for large groups [4, 5]. For each group that an identity is a member of, a symmetric group key is computed. This key is based on the root of a Diffie-Hellman tree and the previous group key, if it exists.

Key Derivation Using Trees in ART. Each leaf of the tree corresponds to an identity, and in particular an identity's current ephemeral public key. In general, every node in the tree is associated with a public key, where the corresponding secret key is known to every member in the subtree with that node as root. This enables all members to compute the root of the tree using their private leaf key and the public keys of the nodes on the copath, which in turn allows them to compute the session key.

To start a group communication in ART, Alice generates an asymmetric "setup key" pair (s, g^s). For each member of the group, Alice uses the member's ephemeral key x (retrieved from the server) and the public setup key g^x to compute a Diffie-Hellman secret g^{xs} that is used as the initial asymmetric secret key for that member's leaf, where $g^{\iota(g^{xs})}$ is the corresponding public key for the leaf. This enables Alice to compute the entire initial group tree even if the other members are currently offline. In subsequent updates, members replace those leaves by public keys whose private key only they know.

To achieve PCS, members can update the group key by generating a new key pair for their leaf, and broadcasting this to the other members. Additionally, the member computes and sends sufficient update information for the other members to also update their tree. Subsequently, all members update the group key in the same way.

Authentication is assumed in the ART design, but not concretely specified.

2.4 Group Messaging in MLS Draft-04

MLS Draft-04 is an ongoing standardization effort by the IETF [1]. Similar to ART, they also employ a tree-based group approach to establishing and updating a symmetric group key with the goal of better efficiency than the previously existing approach employed by Signal. In the following section, we will give a simplified overview over the key update mechanism that is used in MLS Draft-04, focusing on the parts relevant for the PCS guarantees achieved by the protocol.

Key Derivation Using Trees in MLS Draft-04. The tree used in MLS Draft-04 is a left-balanced binary tree. As in ART, every leaf corresponds to one member of the group. Every node is associated with a public key, where the corresponding secret key is known to every member in the subtree with that node as root.

To achieve PCS for a particular group, a member can perform an update operation as follows. The member generates a secret string, from which they derive two other secret strings, a path secret and a node secret. From the node secret, they derive a new key pair for their own leaf node. From the path

secret, they again derive a path secret and a node secret, using the node secret to derive a new key pair for their parent node, and using the path secret to continue this process up the tree, deriving new key pairs for every node on the direct path between their leaf node and the root. To distribute the new secret keys of a node to the members in that node's subtree, they use a hybrid public key encryption algorithm to encrypt the secrets used to derive a new key pair to the nearest public key known for every member. Finally, from the path secret in the root node, they derive a new group secret.

Authentication in MLS Draft-04. Every identity is associated with a public signature key and the protocol ensures that members agree on the list of participants in a group, each member represented by their signature key. Messages involved in performing an update operation as described above are authenticated using this signature key.

3 Security Differences between ART/MLS and Pairwise PCS Channel Groups

In this section we consider particular attacks in two group structure approaches. We base our discussion on the overall structure of existing group messaging protocols such as Signal [7], ART [4], and MLS Draft-04 [1]. Henceforth we differentiate between keys used within groups (*local-level* keys) and those used across groups/identity keys (*global-level* keys).

As this discussion is based on existing protocol designs, we follow the practice of Signal, ART, and MLS in assuming long-term signing keys per identity and symmetric keys within groups (either pairwise in *Approach 1* or as a shared group key as in *Approach 2*). Furthermore, we assume that long-term signing keys are used to sign updates to the symmetric group keys (i.e. global-level asymmetric keys authenticate local-level symmetric key updates). For Approach 1, we consider PCS with regard to the aggregation of all pairwise symmetric keys between all group members. An update to a group in that respect means individual updates to all pairwise channels. Similarly, for Approach 2, we consider PCS with regard to the actual group key, where an update operation means an update the group key shared by the respective group.

For the purposes of the following discussion in this section, *compromise* refers to full state compromise.

For illustration purposes we consider two scenarios. These are not exhaustive, but suffice to convey the main ideas.

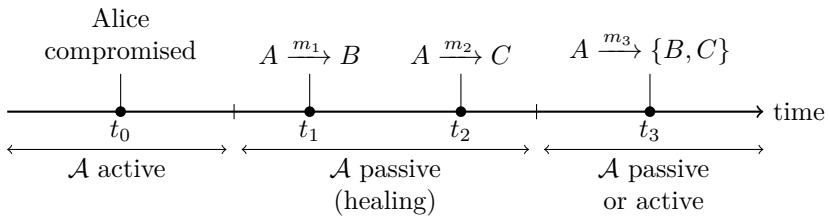

Figure 1. Scenario 1

Scenario 1: Alice gets compromised at t_0. Then, while the adversary is passive, she sends an update message m_1 to Bob and an update message m_2 to Charley over their private channels (i.e. groups of size two). At t_3 she sends a message to a group consisting of Bob and Charley. This scenario is depicted in Figure 1.

Scenarios 2a and 2b: Alice gets compromised at t_0. She then sends an update message m_1 to a large group (which includes Bob) while the adversary is passive. We consider 2a the scenario, where Alice had previously communicated with Bob individually and 2b the one where she didn't. At t_2 she sends a regular message just to Bob. The scenarios are jointly depicted in Figure 2.

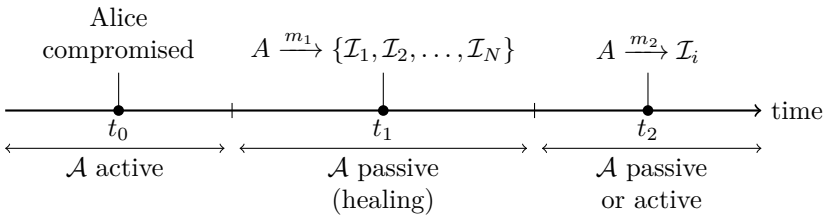

Figure 2. Scenario 2 (a and b), where $1 \leq i \leq N$.

3.1 PCS properties of Approach 1: Group Built from Pairwise Channels

1. Scenario 1: In this case, the message m_1 and m_2 "heal" the compromise for the Alice-Bob and Alice-Charley channels in the process; even if the adversary now becomes active, they can no longer eavesdrop or insert data on those channels. Furthermore, since the group with Bob and Charley is built on top of the one-to-one channels, the group has now healed as well. As a result, m_3 is secure. In terms of channels that are still vulnerable, the adversary can still start completely new conversations or groups after a compromise. If members do not have any previously established channels to heal, then the vulnerability persists.
2. Scenario 2a and 2b: In this case, sending a message m_1 to the group heals all individual channels among participants and all possible groups consisting of a subset of the original group participants, thus covering both 2a and 2b. (This does not include groups comprised of a subset of the original group participants as well as other non-participants; i.e. the pairwise channels to non-participants are not healed.) As a result, m_2 is secure again since the transmission uses the channel that was healed when m_1 was received.

3.2 PCS Properties of Approach 2: Group Channel Separate from Pairwise Member Channels

While the first approach gives good security guarantees, its lack of performance in large groups led to the design of tree-based group key exchanges such as ART [4] and the ongoing standardization efforts of the MLS working group at the IETF [1]. Both approaches claim to achieve PCS guarantees for group keys, but they don't meet the same level of PCS as the one-to-one approach detailed previously. We outline the following issues, in increasing order of nuance. Notably, the latter issues not only demonstrate that ART and MLS Draft-04 to not yet meet the intended PCS claims, but also highlight a weakened form of PCS that may be extremely difficult to communicate to an end-user.

1. Scenario 1: Alice is compromised but sends messages to Bob and Charley, healing the one-to-one channels in the process. However, since the group exists independently of the one-to-one channels, the group key remains unaffected (or is completely new) and the group channel is therefore not healed. Hence the adversary can learn or forge the final message.
   Concretely, this means that even if Alice has sent updates to each of her contacts $CON = \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$, as long as there exists a subset $S \subset CON$ to which no group update was sent, the adversary can still impersonate Alice to the group S.
2. Scenario 2a and 2b: Let Alice be compromised while being a member of a group and let Alice send an update to the group, thus healing that group from the compromise. Despite healing the group, neither for Scenario 2a nor for Scenario 2b, the subsequently created subgroup is healed.
   As a concrete example, consider a large company's messaging group that includes all employees. After Alice (the CEO) is compromised, she sends a message to the entire group. In the pairwise setting this heals all future communications. In the second approach this heals only the group, and the adversary can continue to impersonate Alice to other employees on pairwise channels, until Alice has sent an update message to each of them individually. Additionally, even a passive adversary can continue to eavesdrop on all other groups that Alice was already in (including pairwise channels).
   This variant applies to concurrent groups as well, such that Alice is compromised and heals Group 1 by sending an update, but does not heal in Group 2. Thus, while Alice may believe that she has PCS following an update, she does not actually achieve it.

Finally, it also impacts groups created at or after t_2: the adversary can create any new group impersonating to be Alice. In this particular scenario, this means any subset of the employees that haven't been in a group with Alice previously.

3.3 Observations

Even if we only consider a *passive adversary*, we can already show a significant difference between the approaches. Suppose Alice is in groups $G_1, \ldots, G_n$ when she is compromised. In Approach 1 (using only pairwise channels), once Alice sends an update message to the group $G = \bigcup_{i=1}^{n} G_i$ (or to several groups as long as their union contains G), she is completely healed. The underlying reason is that such updates cause an update of all the underlying pairwise channels, thereby essentially replacing all encryption keys that the attacker might have learned during the compromise. In Approach 2 (ART/ MLS Draft-04), Alice is only completely healed once she has sent update messages in *all* groups $G_1, \ldots, G_n$. Until that point, the adversary can still eavesdrop on all groups in which she has not yet sent an update message.

For an *active adversary*, the situation is even worse. In Approach 1, sending an update to G also heals all future groups that are subgroups of G, even if they were not yet active during the compromise. In Approach 2, while the group key of G can be updated, the re-use of authentication keys would imply that the adversary remains capable of impersonating Alice in any group unless there has been an explicit update to that group. This leaves a large window of opportunity for the adversary to impersonate Alice in the future.

These observations point to the fact that PCS per the original definition [6] is not achieved in these approaches due to the *scope of the update*: in Approach 1, each update message heals a previously compromised sender with respect to the recipient for all future communications between them, irrespective of the context. In contrast, in the group proposals in ART and MLS Draft-04, a previously compromised sender's update message only heals the group context it was sent to.

Remark on "sender keys with PCS key updates". For this version of our work we are not explicitly considering the "sender keys with PCS updates" protocol variant, in which Alice can update her group-specific sender key over the pairwise channels at some intervals. We do note that, for a passive adversary, the update mechanism for Signal's sender keys protocol heals the future messages *sent by Alice* to the updated group, but still allows a passive adversary to eavesdrop on (i) messages *sent by anyone else in the group* to which Alice just sent an update, and (ii) messages sent or received by Alice in other groups. In that sense, combining sender keys with key updates has weaker PCS guarantees than ART or MLS: in sender keys, a group of size N uses N sender keys, and an update by Alice heals only one of them. In this case, a group in which one member was compromised, the group is only healed with respect to a passive adversary once once all members of the group have updated their sender key. This is in contrast to groups based on pairwise PCS channels or ART or MLS Draft-04, in which groups can be healed by only using updates from the compromised party.

4 Solution Options

Given the attacks discussed in Section 3, claiming PCS for some such constructions is both unrealistic and misleading. Consequently, in this section we explore solutions options for messaging protocols such as MLS Draft-04 and ART.

4.1 Options

The main problem in informally claiming Signal-like PCS from MLS and ART style constructions is due to the fact that current guarantees only provide PCS within a single group following an update (local-level) instead of across all groups that an identity is a member of (global-level). Our example scenarios indicate why this may be more problematic than previously assumed. The following list outlines available options dependent on local-level and global-level security goals.

1. Remove PCS guarantees.
   In this solution, PCS claims are removed entirely. Since Section 3 demonstrates that only a highly restricted version of PCS (i.e. local-level PCS) is actually achieved, removing all claims of PCS is a valid solution.
   Effects: If PCS is not required for a group messaging protocol, then it may be possible to reduce system complexity. Consider the proposed key structure for MLS Draft-04. In absence of a PCS claim, updating individual keys inside of the group becomes unnecessary. FS, at a global-level, is achieved by updating the group key only, which can be handled separately for different groups. Succinctly, if PCS is not a desired guarantee, the tree structures in ART and MLS Draft-04 produce an unnecessary and substantial overhead that can be traded for a decrease in performance in large groups.

2. Selectively remove global-level PCS guarantees based on group size and communicate this to users.
   Let $\texttt{thres} \in \mathbb{N}$ be some application-set threshold for the groups size. In this solution, global-level PCS is claimed for groups of size n, where $n \leq \texttt{thres}$, and no PCS claims are made for groups of size $n > \texttt{thres}$.
   Effects: For groups of size $n > \texttt{thres}$, the effects of this option are the same as Option 1 above. For groups of size $n \leq \texttt{thres}$, pairwise Signal or a similar protocol can be used to establish point-to-point keys, upgrading local-level PCS to global-level PCS, thereby eliminating the issues presented in *Approach 1* in Section 3. Clearly, employing a pairwise messaging protocol across all members of a group introduces a high overhead of $n!$ pairwise keys, and communications that are linear in n.

3. Achieve global-level PCS. If PCS is truly a goal of the group messaging protocol, then neither of the above solutions are acceptable. The only alternative is to achieve global-level PCS by another means than those currently proposed in MLS Draft-04 and ART.
   Effects: We will investigate design space for achieving global-level PCS in group messaging in Section 4.2.

4.2 Design Space

In this section we provide a high-level comparison of the cross effects of key updates for achieving PCS, independent of the specific update mechanism or protocol. Following from Section 3, we consider the existence of symmetric group keys shared among group members (gk) and asymmetric signing keys of an identity $\mathcal{I}$ $(sk^{\mathcal{I}}, pk^{\mathcal{I}})$; however, for comparison we also consider group specific asymmetric signing keys, i.e. per-member asymmetric keys which are specific to a given group G, $(sk^{\mathcal{I}}_G, pk^{\mathcal{I}}_G)$. For this initial exploration, we assume that asymmetric keys are either used to sign updates to the symmetric group keys, or are used to encrypt the updates. For the purposes of the following cross comparison, we allow that any of these key types may be updated. Moreover, in the specific comparison below, we do not consider deniability, e.g. how the public keys are communicated to other group members.

Comparing the propagated effects of key updates in group messaging takes on an added layer of complexity if keying boundaries are not well preserved. Consider, for example, the case of an identity $\mathcal{I}$ which participates in multiple groups $G_1, G_2, \ldots, G_n$, and uses the same key material to update all group keys. Even if such an update was performed simultaneously across all groups (e.g. epoch update), certain less desirable properties emerge. Minimally, such an action causes poor key separation, but it could lead to full leakage of the group state to and among different groups, dependent on the update protocol. Enumerating all such cases and corresponding vulnerabilities is problematic, as the actual update protocol in use effects how catastrophic a vulnerability is. Furthermore, it is conceptually a poor design strategy to allow key reuse across groups or to not ensure key separation. To minimize obfuscation of vulnerabilities and in keeping with good design practice, we assume no key reuse and clean key separation in the following comparison, i.e. that keying material used to update one key type is not also used to update another type.

4.3 Updates vs. Key Type Breakdown

Table 1 shows a cross comparison of updates to different types of keys within the group infrastructure, where one update event (updates to keys in columns) causes further update events (updates to keys in

Add'l Affect	$\mathcal{I}$ updates key: gk^{G^*}	(sk^{G^*}, pk^{G^*})	(sk, pk)
gk^{G^*}	ParallelBreak	ParallelBreak	Trivial
gk^{G_i}	Epoch or InfoLeak	Epoch/OH or InfoLeak	OH
(sk^{G^*}, pk^{G^*})	ParallelBreak	ParallelBreak	Trivial
(sk^{G_i}, pk^{G_i})	Epoch/OH or InfoLeak	Epoch/OH or InfoLeak	OH
(sk, pk)	InfoLeak/OH	OH	Single

Table 1. Comparison of sequence of update effects, where an update in the *updates key* (column) is the catalyst for an update to *Add'l Affect* key (row). Notation: Group specific symmetric key for group G^* (gk^{G^*}), Group specific symmetric key for all i (gk^{G_i}), Group specific asymmetric key for group G^* (sk^{G^*}, pk^{G^*}), Group specific asymmetric keys for all i (sk^{G^*}, pk^{G^*}), asymmetric identity key (sk, pk). Terminology: PCS broken for parallel groups (ParallelBreak), updates occurring at an identity-set epoch and achieving PCS at epoch level (Epoch), excessive computational overhead (OH), information leakage (InfoLeak), trivial (Trivial), single update applies to all groups (Single).

rows). Note that, since we assume key separation and the absence of key reuse, the effects listed in Table 1 are about update sequencing, i.e. that the *act* of updating one key causes the *act* of updating another key.

Consider the internal (no-color) columns (C) and rows (R) in Table 1. All updates at the local-level of a given group G^* that do not have any causal effect on updates within other groups result in a concurrent group attack. This is shown in C1/R1, C1/R3, C2/R1 and C2/R3 as *ParallelBreak*. The concurrent group attack in C1/R1 is described in Section 3.

If updates at the local-level of a given group G^* act as a catalyst for updating other groups (i.e. C1/R2 and C1/R4), the concurrent group attack previous described does not hold. However, if such updates are not performed with regularity, the result is an information leakage. For example, if $\mathcal{I}$ is relatively inactive in Group A but active in Group B, the members of Group A will learn information about the activity level of $\mathcal{I}$ in Group B. To prevent this, it is possible to enforce update epochs, i.e. updates according to some frequency interval. The epoch may be set at the application level, but must be enforced such all groups are updated simultaneously. Notably, this may introduce excessive computational overhead, particularly in the case of asymmetric keys. However, such updates still do not allow PCS healing for future - not yet established - groups.

In a similar line of reasoning, we consider updates to group asymmetric keys that cause further updates across all groups (i.e. C2/R2 and C2/R4). This suffers from the similar information leakage, epoch enforcement and weaknesses for future groups as above, with two added caveats.

1. Using epochs, asymmetric keys would need to be updated per group at regular intervals, resulting in a higher computational overhead than an epoch-based update of symmetric group keys.
2. A record of asymmetric keys would need to be maintained, per group, so that new members can verify the current local-level asymmetric key for each group member.

For C1/R5 and C2/R5 we can consider updates to local-level keys that act as a catalyst for updating identity keys for $\mathcal{I}$. This incurs a notable computational overhead, as the identity key must be updated and published after each local-level update in any group. If the case that group symmetric keys cause the identity key update (C1/R5), we incur a similar information leakage as described above.

If updates to the asymmetric identity keys cause updates to per-group asymmetric keys, we incur an unnecessary overhead (i.e. C3/R2 and C3/R4). C3/R1 and C3/R3 are trivial cases of this, where updates are incurred on the local-level asymmetric key in only one group.

Finally, we can consider updates to the identity's asymmetric key pair, which apply solely to that pair and do cause updates to any local-level keys (C3/R5). Notably, this is a single update that applies across all groups simultaneously, thereby reducing overhead. Furthermore, global-level updates apply not only to all current groups but also all future groups, yielding PCS for current groups as well as groups which have not yet been established.

5 Solution Sketch and High-Level Comparison

Our analysis points to two important aspects of a solution:

1. global-level updates are necessary for effective PCS group healing, and
2. secure updates to signature keys are necessary to enable authentication after compromise, ideally accross groups.

In this extended abstract, we focus on the second point, and address the first point in detail in the extended version. At the end of this section we will nevertheless sketch how these aspects together may achieve stronger PCS guarantees.

Concretely, we introduce the notion of a *Ratcheting Digital Signature* (RSIG) scheme, which formally captures the type of rotating signatures we require, and the associated *Post-Compromise Secure Signature* (PCS2) security notion. We discuss such signatures can be constructed from standard SUF-CMA signatures. Afterwards, we show this solution can be used to achieve the PCS guarantees aimed for in the goals of MLS and ART.

5.1 Post-Compromise Secure Signatures

Forward-secure signatures were first proposed by Bellare and Miner [2]. However, as with the general distinction between FS and PCS, the security demands on the signature scheme proposed above show a different security goal than FS. Consequently we propose PCS signatures security. We employ ratcheting digital signatures below, which differ from the key evolving signatures used for forward-secure signatures in that not only the secret but also the public key is updated. Subsequently we introduce the PCS signature security game.

Definition 1 (Ratcheting Digital Signatures). *A* ratcheting digital signature *scheme is a tuple of algorithms* $\text{RSIG} = (\text{RSIG.Gen}, \text{RSIG.Update}, \text{RSIG.RcvUpdate}, \text{RSIG.Sign}, \text{RSIG.Verify})$*, where:*

- RSIG.Gen *is a probabilistic key generation algorithm which takes as input a security parameter* $\lambda \in \mathbb{N}$ *and returns a pair* (sk, pk)*, where* sk *is the initial secret key and* pk *is the initial public key.*
- RSIG.Update *is a (possibly probabilistic) key update algorithm which takes as input a secret and public key pair* (sk, pk) *and returns a new secret and public key pair* (sk', pk') *and an update message* m_u*.*
- RSIG.RcvUpdate *is a deterministic update receiving algorithm which takes as input a public key* pk *and an update message* m_u*, and returns the updated public key* pk'*.*
- RSIG.Sign *is a (possibly probabilistic) signing algorithm which takes as input the secret key* sk *and a message* M *and returns* σ*, a signature on* M*.*
- RSIG.Verify *is a deterministic verification algorithm which takes as input a public key* pk*, a message* M*, and a signature* σ_M *and outputs bit* verify *such that* $\text{verify} = 1$ *if* σ_M *is a valid signature of* M *and* $\text{verify} = 0$ *otherwise.*

Correctness is left to the full version.

Definition 2 (PCS2 Security). *We say that a ratcheting signature scheme* RSIG *is* PCS-signature (PCS2) secure *if there exists a negligible function* $\mathsf{negl}(\lambda)$ *such that for all security parameters* $\lambda \in \mathbb{N}$ *and all polynomial-time adversaries* $\mathcal{A}$ *interacting according to the experiment* $\mathsf{Exp}^{\mathsf{pcs-sig}}_{\text{RSIG}}(\mathcal{A})$ *in Fig. 3, it holds that*

$$\Pr[\mathsf{Exp}^{\mathsf{pcs-sig}}_{\text{RSIG},\mathcal{A}}(\lambda) = 1] \leq \mathsf{negl}(\lambda) \ .$$

The above notion of PCS signature security captures not only general forgeries, but an adversary's ability to forge signatures *following* a key compromise. Intuitively, an adversary wins if they can forge a signature that is valid under the current key of the verifier, and where an update has occurred after all past key corruptions (if any) during which the adversary was passive.

Experiment Details. Throughout the experiment, a sequence number sqn increments according to the number of PCS2.Update and PCS2.Corrupt queries the adversary makes. This is used for global ordering.

- PCS2.Update may be called by the adversary to operate on the current key. The oracle checks that this is the current value and proceeds to process the update request, updating the current sender keys to the new values. Additionally, the generated update message is recorded according to the current sequence value, with the sequence value also being recorded separately and incremented. The new public key and update message are returned to the adversary.

$\mathsf{Exp}^{\mathsf{pcs-sig}}_{\mathrm{RSIG},\mathcal{A}}(\lambda)$:

$(sk, pk) \xleftarrow{\$} \mathrm{RSIG.Gen}(\lambda)$
$\mathsf{rcv.pk} \leftarrow pk, \mathsf{snd.pk} \leftarrow pk, \mathsf{snd.sk} \leftarrow sk$
$\mathsf{decision} \leftarrow 0$
$\mathsf{sqn} \leftarrow 0$
$\mathsf{m}_{\mathsf{sqn}} \leftarrow \perp$
$\mathsf{Corrupted} \leftarrow \emptyset$
$\mathsf{SigList} \leftarrow \emptyset$
$\mathsf{ValidPairs} \leftarrow \{(sk, pk)\}$
$\mathsf{UpdateMsgIndex} \leftarrow \emptyset$
$\mathsf{RecdUpdateMsgs} \leftarrow \emptyset$
$\mathcal{A}^{\mathsf{PCS2.Update}(\cdot),\mathsf{PCS2.RcvUpdate}(\cdot),\mathsf{PCS2.Sign}(\cdot),\mathsf{PCS2.Verify}(\cdot),\mathsf{PCS2.Corrupt}(\cdot)}()$
return decision

PCS2.Update():

$((sk', pk'), m) \leftarrow \mathrm{RSIG.Update}(\mathsf{snd.sk}, \mathsf{snd.pk})$
$\mathsf{ValidPairs} \leftarrow \mathsf{ValidPairs} \cup \{(sk', pk')\}$
$(\mathsf{snd.sk}, \mathsf{snd.pk}) \leftarrow (sk', pk')$
$\mathsf{m}_{\mathsf{sqn}} \leftarrow m$
$\mathsf{UpdateMsgIndex} \leftarrow \mathsf{UpdateMsgIndex} \cup \{\mathsf{sqn}\}$
$\mathsf{sqn} \leftarrow \mathsf{sqn} + 1$
return $(\mathsf{snd.pk}, m)$ to $\mathcal{A}$

PCS2.Sign(M):

$\sigma \leftarrow \mathrm{RSIG.Sign}(\mathsf{snd.sk}, M)$
$\mathsf{SigList} \leftarrow \mathsf{SigList} \cup \{(\sigma, M, \mathsf{snd.pk})\}$
return σ to $\mathcal{A}$

PCS2.Corrupt(pk):

if $\nexists sk, (sk, pk) \in \mathsf{ValidPairs}$ **then**
  return $\perp$ to $\mathcal{A}$
$\mathsf{Corrupted} \leftarrow \mathsf{Corrupted} \cup \{\mathsf{sqn}\}$
$\mathsf{sqn} \leftarrow \mathsf{sqn} + 1$
return sk to $\mathcal{A}$

PCS2.RcvUpdate(m):

$pk' \leftarrow \mathrm{RSIG.RcvUpdate}(\mathsf{rcv.pk}, m)$
if $pk' \neq \perp$ **then**
  $\mathsf{RecdUpdateMsgs} \leftarrow \mathsf{RecdUpdateMsgs} \cup \{m\}$
  $\mathsf{rcv.pk} \leftarrow pk'$
return rcv.pk to $\mathcal{A}$

PCS2.Verify(M, σ):

$\mathrm{verify} \leftarrow \mathrm{RSIG.Verify}(\mathsf{rcv.pk}, M, \sigma)$
if
$\big(\forall sqn' \in \mathsf{Corrupted},$
$\exists j \in \mathsf{UpdateMsgIndex},$
$\bullet j > sqn' \wedge$
$\bullet \mathsf{m}_j \in \mathsf{RecdUpdateMsgs}\big)$
$\wedge$
$\big((\sigma, M, \mathsf{rcv.pk}) \notin \mathsf{SigList}\big)$
$\wedge$
$\bullet\big(\mathrm{verify} = 1\big)$ **then**
  $\mathsf{decision} \leftarrow 1$
  return decision from experiment
return verify to $\mathcal{A}$

Figure 3. PCS2 signature security experiment.

- PCS2.Sign may be called by the adversary on any message of its choosing. The signature, message, and current public key of the signer are recorded and the signature is returned to the adversary.
- PCS2.Corrupt may be called by the adversary on any public key of its choosing. If the public key is not part of a key pair previously generated with a call to PCS2.Update, the oracle terminates. Otherwise, the current sequence number is recorded and the sequence number is updated be the secret key is returned to the adversary.
- PCS2.RcvUpdate may be called by the adversary on any update message (generated by PCS2.Update or forged). The oracle processes the update message under the currently recorded public key of the sender and checks if the output is a valid public key. If so, the receiver side updates its recorded public key for the sender and adds the update message to its record list, returning the updated public key to the adversary.

  Note that this requires only that a valid public key is output from RSIG.RcvUpdate, not that it matches the actual updated sender public key. Consequently, an adversary may forge an updated that is accepted, resulting in $\mathsf{snd.pk} \neq \mathsf{rcv.pk}$.
- PCS2.Verify may be called by the adversary on any message and signature pair (generated by PCS2.Sign or forged). The oracle processes the signature verification and allows a win to the adversary if all of the following hold:
  - An update message has been both correctly generated and received following any and all corruption queries.

- The signature was not generated on the message under the recorded public key of the sender using the PCS2.Sign oracle. Note that this requirement fails if the signature, message, or public key do not match (i.e. strong unforgeability inclusive of public key).
- The verification processes correctly.

If all of the above hold, the adversary wins. Note that the first clause of the conjunction requires that m_j was both correctly generated and processed, since $\mathsf{m}_j \in \mathsf{RecdUpdateMsgs}$ implies that it is in the ancestral update path of snd.pk.

Remark 1. Intuitively, RSIG keys are simply signature keys and an update message may simply be a new public key that is verifiably linked to the previous signing key. Thus a natural construction of an RSIG signature scheme exists from a SUF-CMA signature scheme (strong unforgeability being a requirement) where updates are signed. In order to prevent the adversary from forging update messages by having a party sign a given public key, regular signatures and update messages must be easily distinguishable. The concrete construction and proof of security are left for the full version.

5.2 Summary ART and MLS Draft-04 Solutions and RSIG

As discussed with the example of two specific scenarios in Section 3, there are significant differences between the PCS guarantees achieved by ART and MLS Draft-04 and those achieved by groups based on pairwise channels. In this section, we give an intuition how an RSIG scheme can be used to close this gap. Additionally, we introduce a new Scenario 3 that showcases how a thusly improved ART or MLS Draft-04 provides PCS guarantees beyond what is currently provided by groups based on pairwise channels.

Recall that in the pairwise approach, if an identity is a member in multiple groups, these groups do not exist independently of each another, but instead rely on the same pairwise channels between their participants. In contrast, in MLS Draft-04 and ART, groups exist independently from one another with the exception of the signature key, which is used in all groups to authenticate each identity.

Let π be the protocol that behaves like MLS Draft-04, using an RSIG scheme instead of a standard signature scheme to authenticate updates.

We can create a tighter binding in terms of PCS guarantees between all groups without a change to the protocol itself by parallelizing group updates, such that updates to all groups are concurrent (i.e. epoch updates). In the following discussion we step through the scenarios of Section 3 for Approach 1 and Approach 2 using epochs, and Approach 2 with π, for comparison. However, before we begin, we also introduce a new Scenario 3, which showcases the additional PCS that π achieves over pairwise Signal as follows:

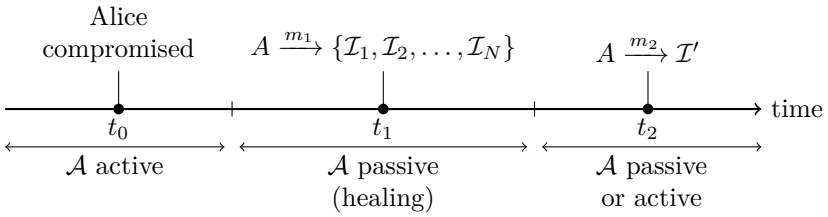

Figure 4. Scenario 3, where $\mathcal{I}' \notin \{\mathcal{I}_1, \ldots, \mathcal{I}_N\}$.

Scenario 3: Alice gets compromised at t_0. She then sends an update to all her groups in t_1. Subsequently, she starts a new group with Bob and Charley at t_2, where there exist no previous communication between the three (meaning that Alice, Bob and Charley to not share membership in any group). This scenario is depicted in Figure 4.

Scenario 1

Suppose that Alice is compromised, and then sends updates to each group member. Alice then sends a message to the group.

Approach 1 Epochs do not affect the security of Signal – new groups and conversations are still vulnerable in the example of Section 3.1.

Approach 2 This prevents cases where a group consisting of Alice, Bob and Charley is healed after Alice updates following a compromise, but not a second group consisting of just Alice and Bob. This fixes the problem exposed in Scenario 1 of Section 3.2. However, new groups and conversations are still vulnerable as they are not affected by the update.

Approach π This fixes the problem exposed in Scenario 1 of Approach 2 in Section 3.2 similarly to using epochs, but also yields PCS for new groups and conversations.

Scenarios 2a and 2b

Suppose that Alice is compromised, and then sends updates in each group signed with her identity key. Alice then communicates with Bob, a member of one of those groups. In 2a, Alice has previously communicated with Bob, in 2b, she hasn't.

Approach 1 Signal protects against both 2a and 2b. 2a is covered, as the group update also updates the previously existing pairwise channel with Bob. 2b is covered, because even if no conversation with Bob took place before, a pairwise channel still exists between Alice and Bob due to the group conversation, which is updated by the group update.

Approach 2 In 2a, epochs allow MLS Draft-04 and ART to achieve PCS in the communication with Bob, as in conjunction with the update to the group, the channel to Bob is also updated due to the epoch update procedure. However, in 2b, neither MLS Draft-04 nor ART heal the channel to Bob (i.e. the attack in Section 3.2 still holds despite epochs). This is due to the fact that no channel to Bob existed previous to the update and thus nothing can be updated.

Approach π The ratcheting signature of π allows it to achieve PCS in both 2a and 2b. 2a is covered due to epoch updates in the same way as in Approach 2. 2b is covered, because the rotating signature prevents an active adversary from impersonating Alice any conversation, even if no previous conversation took place. In other words, the authentication of Alice is globally healed, whereas in the pairwise approach, the authentication heals only with respect to existing communication partners.

Scenario 3

Suppose that Alice is compromised, and then sends updates in each group signed with her identity key. Alice then creates a new group with Bob and Charley, both of which she has not communicated before and both of which neither share group membership with each other, nor with Alice.

Approach 1 Signal does not give PCS guarantees in this Scenario, as even after updating all existing channels, an adversary can still impersonate Alice towards Bob and Charley.

Approach 2 Neither MLS Draft-04 nor ART achieve PCS guarantees here, as similar to Signal, no updated key material is shared with Bob and Charley.

Approach π The ratcheting signature of π allows it to achieve PCS in Scenario 3 in the same way as in Scenario 2b. The updated signature key prevents the adversary from impersonating Alice towards Bob and Charly and thus allows Alice to achieve PCS.

6 Conclusion and Future Work

We have shown that achieving post compromise security by updating keying material internally to groups at a local-level leaves a significant window of opportunity for an adversary and fails to meet global-level PCS guarantees. In practice, this means there is a substantial difference between the global-level PCS guarantees offered by groups implemented over point-to-point PCS channels and the current proposals for more efficient group key solutions such as ART and MLS Draft-04 which only achieve local-level PCS. Notably, this shows that the informally described PCS guarantees in MLS Draft-04 are not met by the current design.

For MLS, this implies three options as it moves towards standardization:

1. Explicitly reduce claimed PCS guarantees to a local-level, thereby explicitly allowing attacks on PCS at the global-level.

2. For large groups, explicitly reduce PCS guarantees to a local-level or remove the guarantees entirely, and guarantee global-level PCS for small groups only.
3. Achieve global-level PCS to meet the current goals. This can be done by, for example, (i) concurrent global updates together with (ii) ratcheting signatures, as discussed in Section 4.2 and Section 5.

In addition to the above conclusions about PCS within groups, this work highlights the distinction between local-level and global-level security guarantees, and the relationship between FS signatures and PCS signatures.

References

1. Barnes, R., Millican, J., Omara, E., Cohn-Gordon, K., Robert, R.: The Messaging Layer Security (MLS) Protocol. Internet-Draft draft-ietf-mls-protocol-04, IETF Secretariat (January 2019), http://www.ietf.org/internet-drafts/draft-ietf-mls-protocol-04.txt
2. Bellare, M., Miner, S.K.: A forward-secure digital signature scheme. In: Advances in Cryptology - CRYPTO '99 Proceedings. pp. 431–448 (1999)
3. Cohn-Gordon, K., Cremers, C., Dowling, B., Garratt, L., Stebila, D.: A formal security analysis of the Signal messaging protocol. In: 2017 IEEE European Symposium on Security and Privacy (IEEE EuroS&P). pp. 451–466 (April 2017)
4. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. Cryptology ePrint Archive, Report 2017/666 (2017), http://eprint.iacr.org/2017/666
5. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: Asynchronous group messaging with strong security guarantees. In: ACM CCS 18. pp. 1802–1819. ACM Press (2018). doi: 10.1145/3243734.3243747
6. Cohn-Gordon, K., Cremers, C.J.F., Garratt, L.: On Post-compromise Security. In: IEEE 29th Computer Security Foundations Symposium, CSF 2016. pp. 164–178 (2016). doi: 10.1109/CSF.2016.19
7. Moxie Marlinspike, T.P.: The Signal Protocol. Tech. rep. (November 2016), https://signal.org/docs/specifications/x3dh/