On the Cost of Post-compromise Security in Concurrent Continuous Group-Key Agreement

Benedikt Auerbach(✉), Miguel Cueto Noval, Guillermo Pascual-Perez, and Krzysztof Pietrzak

ISTA, Klosterneuburg, Austria
{bauerbac,mcuetono,gpascual,pietrzak}@ista.ac.at

Abstract. Continuous Group-Key Agreement (CGKA) allows a group of users to maintain a shared key. It is the fundamental cryptographic primitive underlying group messaging schemes and related protocols, most notably TreeKEM, the underlying key agreement protocol of the Messaging Layer Security (MLS) protocol, a standard for group messaging by the IETF. CKGA works in an asynchronous setting where parties only occasionally must come online, and their messages are relayed by an untrusted server. The most expensive operation provided by CKGA is that which allows for a user to refresh their key material in order to achieve forward secrecy (old messages are secure when a user is compromised) and post-compromise security (users can heal from compromise). One caveat of early CGKA protocols is that these update operations had to be performed sequentially, with any user wanting to update their key material having had to receive and process all previous updates. Late versions of TreeKEM do allow for concurrent updates at the cost of a communication overhead per update message that is linear in the number of updating parties. This was shown to be indeed necessary when achieving PCS in just two rounds of communication by [Bienstock *et al.* TCC'20].

The recently proposed protocol CoCoA [Alwen *et al.* Eurocrypt'22], however, shows that this overhead can be reduced if PCS requirements are relaxed, and only a logarithmic number of rounds is required. The natural question, thus, is whether CoCoA is optimal in this setting.

In this work we answer this question, providing a lower bound on the cost (concretely, the amount of data to be uploaded to the server) for CGKA protocols that heal in an arbitrary k number of rounds, that shows that CoCoA is very close to optimal. Additionally, we extend CoCoA to heal in an arbitrary number of rounds, and propose a modification of it, with a reduced communication cost for certain k.

We prove our bound in a combinatorial setting where the state of the protocol progresses in rounds, and the state of the protocol in each round is captured by a set system, each set specifying a set of users who share a secret key. We show this combinatorial model is equivalent to a symbolic model capturing building blocks including PRFs and public-key encryption, related to the one used by Bienstock *et al.*

© International Association for Cryptologic Research 2023
G. Rothblum and H. Wee (Eds.): TCC 2023, LNCS 14371, pp. 271–300, 2023.
https://doi.org/10.1007/978-3-031-48621-0_10

Our lower bound is of order $k \cdot n^{1+1/(k-1)} / \log(k)$, where $2 \leq k \leq \log(n)$ is the number of updates per user the protocol requires to heal. This generalizes the n^2 bound for $k = 2$ from Bienstock *et al.*. This bound almost matches the $k \cdot n^{1+2/(k-1)}$ or $k^2 \cdot n^{1+1/(k-1)}$ efficiency we get for the variants of the CoCoA protocol also introduced in this paper.

1 Introduction

A fundamental task underlying various cryptographic protocols is to agree upon, and maintain, a secret key amongst a group of users. A prominent example is *continuous group-key agreement* (CGKA) [3], which underlies group messaging applications. Here, a group of users wants to maintain a shared secret key, that then can be used for private communication amongst the group members.

CGKA is defined in an asynchronous setting, where parties are online only occasionally, and the exchanged messages are relayed through an untrusted server (only trusted to provide liveness and thus correctness). CGKA allows for users to be added or removed from the group. Moreover users can update their keys, which allows the group to achieve forward secrecy (FS) and post-compromise security (PCS). FS guarantees that, should a user's secrets be compromised, messages sent in the past remain secure. PCS, in turn, allows the group to "heal", i.e. to recover privacy after a compromise occurs.

The most efficient existing protocols for CGKA are TreeKEM [10] and variants thereof [2,3,7,22,23], which are inspired by logical key hierarchies (LKH) [25], a popular protocol for multicast encryption (ME) [14]. The study of these protocols has received a great deal of attention recently, motivated by the IETF working group on *Message Layer Security* (MLS) [9], which aims to output standard for instant group messaging. Said standard employs TreeKEM as the underlying CGKA. These schemes all arrange keys from a public-key encryption scheme in trees, known as *ratchet trees*, where each node is associated with a key, each user is associated with a leaf, and users should know exactly the (secret) keys on the path from their leaf to the root (also known as the *tree invariant*).

A simple ratchet tree with four users is illustrated in Fig. 1. The advantage of using such a hierarchical tree structure is that replacing a user's keys in a group of size n just requires the creation of $\lceil \log(n) \rceil$ ciphertexts, while e.g. maintaining pairwise keys between the users would require $n - 1$.

Concurrent Updates. Updating keys in a ratchet tree as illustrated in Fig. 1 only works if updates are sequential. That is, if two users want to update, then they need to do it in order, with the second processing the first user's update before creating their own. TreeKEM supports concurrent updates through the "propose and commit" (P&C) paradigm, but handling concurrency in this way degrades the nice tree structure and thus efficiency of the protocol. Indeed, after several users update concurrently, all of their paths to the root but one will lose their keys, a.k.a. become *blank*, increasing the in-degrees of nodes in the tree and thus the cost of that and subsequent operations. This incurs an overhead that is linear in the number of updating parties, something which was shown to be

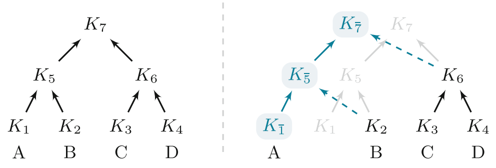

Fig. 1. Left: Illustration of a ratchet tree with $n = 4$ users $\{A, B, C, D\}$ where each key $K_i = (pk_i, sk_i)$ is a public/secret key tuple. Right: To update and achieve PCS Alice rotates keys $\{K_1, K_5, K_7\}$ by sampling new keys $\{K_{\bar{1}}, K_{\bar{5}}, K_{\bar{7}}\}$ (blue, shaded background) and encrypting each secret key under the public key of their parent (blue, dashed arrows), e.g. $K_2 \rightarrow K_{\bar{5}}$ corresponds to a ciphertext $\mathsf{Enc}_{pk_2}(sk_{\bar{5}})$. Given those ciphertexts, all users can learn the new keys on their path to the root. For example Bob must decrypt the ciphertexts $\mathsf{Enc}_{pk_2}(sk_{\bar{5}})$ and $\mathsf{Enc}_{pk_{\bar{5}}}(sk_{\bar{7}})$. This requires $2\lceil\log(n)\rceil = 4$ ciphertexts. However, by deriving the keys $\{K_{\bar{1}}, K_{\bar{5}}, K_{\bar{7}}\}$ deterministically from a single seed using a PRG as suggested in [14], we can save the ciphertexts for the solid blue arrows and only need $\lceil\log(n)\rceil = 2$ ciphertexts. (Color figure online)

optimal by Bienstock, Dodis and Rösler [12], whenever PCS is to be achieved as soon as all corrupted users update once.

CoCoA [2] takes a different approach, and simply choses a "winner" whenever there is a conflict, i.e., when two users want to concurrently replace the same key, as illustrated in Fig. 2. As opposed to the previous scenario, this does not immediately "heal" the state of the concurrently updating parties (in the Figure, key $K_{\bar{7}}$ is not secure if Dave's key K_6 was compromised). However, in [2] it is shown that the group heals (i.e., achieves PCS) after all corrupted users participate in $\log(n)$ (possibly concurrent) update rounds. This is a middle ground between the immediate concurrent healing of P&C TreeKEM, and the n sequential rounds needed for non-concurrent versions of TreeKEM.

In this work we prove a lower bound on the communication cost of CGKA protocols that heal in any number of (up to logarithmic in the group size) rounds.

A Combinatorial Model. Conceptually, our lower bound proof proceed in two steps. We first derive the lower bounds in a clean and simple combinatorial model which proceeds in rounds. The state of the protocol for n users in round t is captured by a set system $\mathcal{S}_t \subseteq 2^{[n]}$, where $S \in \mathcal{S}_t$ means that after round t there is a shared secret amongst the users S *not* known to the adversary.

In particular, $[n] \in \mathcal{S}_t$ means the group $[n] = \{1, \dots, n\}$ shares a secret, which has to be satisfied in all rounds with a secure group key.

For example, in the ratchet tree example from Fig. 1 (where users are denoted $\{A, B, C, D\}$ not $\{1, 2, 3, 4\}$), the sets corresponding to the keys $K_1, .., K_7$ are

$$\mathcal{S}_t = \{\{1\}, \{2\}, \{3\}, \{4\}, \{1, 2\}, \{3, 4\}, \{1, 2, 3, 4\}\}.$$

If a user u gets compromised, all secrets corresponding to sets containing u become known to the adversary and thus the sets must be removed, e.g., if we compromise user 1, the set system becomes

$$\mathcal{S}_{t+1} = \{\{2\}, \{3\}, \{4\}, \{3, 4\}\}.$$

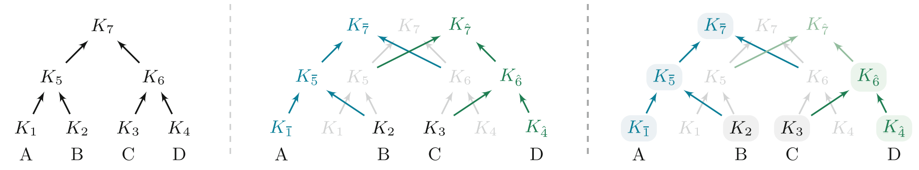

Fig. 2. Illustration of CoCoA where Alice and Dave concurrently rotate their keys. Left: state of the ratchet tree before the updates. Middle: Keys $\{K_{\bar{1}}, K_{\bar{5}}, K_{\bar{7}}\}$ and $\{K_{\hat{4}}, K_{\hat{6}}, K_{\hat{7}}\}$ generated by Alice and Dave's updates respectively. There's a collision at the (root) key K_7, and the server chooses a "winner" (any rule for choosing winners will do), in this case Alice. Right: New state of the ratchet tree (Keys in the tree depicted with shaded background). The new root key is $K_{\bar{7}}$ while Dave's $K_{\hat{7}}$ is ignored. As $K_{\bar{7}}$ was encrypted to K_6 we do not achieve PCS if Dave's state $\{K_4, K_6, K_7\}$ prior to the update was compromised. But latest once all corrupted parties updated $\log(n)$ times PCS will be achieved (in particular, if Dave updates once more PCS is achieved).

A user u can update and create new sets (keys) as follows. They can always locally sample a key, creating the singleton $\{u\}$. For two sets $S, S' \in \mathcal{S}$, where $u \in S$ (or $u \in S'$), they can create a new set $S \cup S'$, by deterministically deriving a secret from that of S using a PRF, and encrypting it under the public key of S'. This would get added to $\mathcal{S}$ in the next round. Note that indeed all users in $S \cup S'$ are able to derive the secret either deterministically from the secret associated to S or by decrypting the ciphertext. In the simplest version of TreeKEM, user 1 performs an update by creating $\{1\}$, then $\{1, 2\} = \{1\} \cup \{2\}$, then $\{1, 2, 3, 4\} = \{1, 2\} \cup \{3, 4\}$ (i.e., the keys $K_{1'}, K_{5'}, K_{7'}$ in Fig. 1). Of course, u is not restricted to create new sets as the union of only two sets, but could also encrypt the secret using the keys of sets $S_1, \ldots, S_k$ to form the set $S \cup \bigcup_{i=1}^{k} S_i$. The communication cost of this operation, i.e., the number of ciphertexts that have to be uploaded to the server to communicate the new secret to all members of the corresponding set, would in this example be k. In Sect. 3.1 we extend this idea into a self-contained combinatorial model consisting of set system $\mathcal{S}_t$ and an accompanying cost function Cost required to satisfy properties matching the intuition given above. In Sect. 3.2 we use it to prove our lower bounds.

The Symbolic Model. While the combinatorial model offers a clean model for proving lower bounds, it is not obvious how it captures real-world protocols. We show that any lower bound in the combinatorial model implies a lower bound in a symbolic model capturing pseudorandom functions and public-key encryption. Most existing CGKA protocols can be captured in this symbolic model and the fact that lower bounds in the combinatorial model carry over to the symbolic model justifies the interest of the combinatorial model we propose. Symbolic models were introduced by Dolev and Yao [18] in public key encryption, used in multicast encryption by Micciancio and Panjwani [24] and in CGKA by Bienstock, Dodis, and Rösler [12] and Alwen *et al.* [1]. In the symbolic model pseudorandom functions and public-key encryption are treated in an idealized way by seeing their inputs and outputs as variables with a data type, which, in turn, follow some grammar rules, and ignoring other considerations that an actual construction may have. The functionality and security of these primitives are captured by the grammar rules and entailment relations in Sect. 4.1.

1.1 Our Bounds

In this work we prove lower bounds on the communication cost of CGKA protocols achieving PCS. Moreover, in Sect. 5 we introduce a new protocol, a modification and generalization of CoCoA. It introduces the necessary number of rounds to heal as a parameter and, in some cases, improves over the natural generalization of CoCoA in this setting.

We measure the cost of a protocol in terms of the number of ciphertexts that users in a group must create (and upload to a server for the other users to download) to achieve post-compromise security.[1] Sometimes, we additionally put a bound on the number of rounds required for parties to heal. We do not require forward-secrecy and will also consider groups of a fixed size, i.e., without removals or additions of users, just updates. Note that both of these make the lower bounds stronger, as an adversary could always choose to not use add/removes. Additionally, FS is relatively well understood [3].

We consider the setting where the users do not know who is compromised or who else will update in any given communication round, and the adversary schedules who does updates in each round. This is similar to that of [12].

When a user is corrupted, we assume its entire secret state is leaked to the adversary, who can also observe all its local randomness. We call this the "randomness corruption" model (RC for short), but we also consider a weaker "no-randomness corruption" ($\neg$RC) model, where only the secret state is leaked. In this model, a corrupted user can still create encrypted secrets for other users. Most protocols are proven secure in the stronger RC model, whereas lower bounds are naturally stronger in the $\neg$RC model. Our lower bounds require some additional restrictions on the CGKAs discussed at the end of the introduction.

Lower Bound. The number k of updates a user is required to make before their state is guaranteed to heal plays a crucial role. Our security game is parameterized by the number of users n and k. The adversary schedules who updates in each round, and we require that, at any point, the group key is secure provided every party who was corrupted in the past was asked to update at least k times (since their last corruption). Table 1 states our lower bound and upper bound, as well as existing ones. Our lower bound is roughly $n^{1+1/k} \cdot k/\log(k)$.

The main message here is that we need to allow for logarithmically many rounds for healing (as in CoCoA) if we want a small logarithmic sender communication cost per user. In particular, if we insist on a constant number of rounds, the average cost per user will be of order $n^{1/k}$.

[1] It is possible, as in [2,5], to reduce recipient communication by introducing additional reliance on the server. We focus on sender communication.

Table 1. Upper-bounds (top) and lower-bounds (bottom) in the no-information setting for $\Omega(n)$ corrupted users. Communication is measured as total number of ciphertexts sent to recover from corruption, column "Rounds" indicates the number of update rounds after which schemes are required to recover from corruption, column "Rand. corr.", whether the security model allows the adversary to learn internal randomness of algorithms. The protocol [12] improves over TreeKEM in that concurrent operations do not degrade future performance, which is not captured in the table. Our lower-bounds require CGKA to not allow distributed work (NDW) and not use nested encryption (NNE). Our bound holds without the extra assumption requiring the protocols to have publicly-computable update cost (PCU). However, additional properties of it hold when this assumption is present. We refer the reader to the discussion in Sect. 1.3 below for more details. Here, $\alpha_\varepsilon \approx \varepsilon$ is some constant depending on ε.

Upper bounds

Scheme	Communication	Rounds	Rand. corr	See
TreeKEM and related	n^2	2	RC	[10]
Bienstock, Dodis, Rösler	n^2	2	$\neg$RC	[12]
CoCoA on $\sqrt[k-1]{n}$-ary trees	$n\,k^2\,\sqrt[k-1]{n}$	k	RC	Sect. 5
CoCoA on 2-ary trees	$n\log(n)^2$	$\log(n)$	RC	[2]
CoCoALight on $\sqrt[(k-1)/2]{n}$-ary trees	$n\,k\,\sqrt[(k-1)/2]{n}$	k	RC	Sect. 5

Lower bounds

Restrictions	Communication	Rounds	Rand. corr	See
None	n^2	2	$\neg$RC	[12]
NDW, NNE, PCU*	$n\log(n)/\log(\log(n))$	$\log(n)$	$\neg$RC	Cor. 5
NDW, NNE, PCU*	$\varepsilon \cdot n \cdot \sqrt[(1+\varepsilon)k-1]{\alpha_\varepsilon n} \cdot k/\log(k)$	k	$\neg$RC	Cor. 5

Upper Bound. We introduce in Sect. 5 the protocol CoCoALight, a modification of CoCoA that achieves PCS in $k \in [4, 2\lceil\log(n)\rceil + 1]$ rounds. This protocol has a cost $k \cdot n^{1+2/(k-1)}$, which matches the lower bound up to a factor $\log(k)/n^{1/(k-1)}$. In particular, our protocol is only a factor of $\log(\log(n))$ from optimal for k in the order of $\log(n)$. In turn, CoCoA (or rather, a straightforward generalization of it we propose for $k \in [2, \lceil\log(n)\rceil + 1]$, as opposed to $k = \lceil\log(n)\rceil + 1$ in the original protocol) has better efficiency for low values of k. The key insight in our protocol is that users do not need to update all the keys in their path to heal. In fact, it suffices for them to update keys one by one, as long as every key in the path is updated twice. We formalize and discuss this further in Sect. 5 of this paper's full version [8].

1.2 Our Proofs

The details of the proofs are omitted in this version of the paper and can be found in the full version [8]. Below we give an intuition.

Proof of the Lower Bound. To prove the lower bound we first show that, if the protocol can heal from c corruptions in k rounds, then there is some user whose cost is $c^{1/k}$. In particular, if $c = \Theta(n)$, we get a cost of $\Theta(n^{1/k})$. The intuition for this is quite simple, let us give it for $c = n$. Initially everyone is corrupted, so our set system is simply $\{1\}, \ldots, \{n\}$, after the kth round $[n] = \{1, \ldots, n\}$ is in our set system. If we denote with s_i the size of the largest set in round i, we have $s_1 = 1, s_k = n$, which means there must be a round i where $s_{i+1}/s_i \geq n^{1/(k-1)}$. Therefore, in this round, the user creating the new set of size s_i has cost $\geq n^{1/(k-1)}$. A slightly more careful argument shows that the maximum cost of a user in each round adds up to $k \cdot c^{1/(k-1)}$.

To prove our bound we will show that for $c = \Theta(n)$ corruptions, we can adversarially schedule the updates so that a $1/\log(k)$ fraction of users (and not just a single one) can be forced to pay close to the maximum cost in each round, which then adds up to $n^{1+1/(k-1)} \cdot k/\log(k)$.

This adversarial scheduling goes as follows: before each round, the adversary investigates each user's cost, should they be asked to update in the next round. Then, it simply picks a $1/\log(k)$ fraction of users, all having either very small cost or, if such a set does not exist, a set of users with roughly the same cost (we show that such a set of users always exists).

Proof of the Upper Bound. We prove our protocol secure by following the framework set by [22], which reduces the adaptive security of a CGKA protocol to that of a game played on graphs. One first defines a so-called *safe predicate*, which captures the settings in which security should be guaranteed (i.e. every corrupted user performed k updates since their last corruption, in our case). This is implicit in our security game. Then, in order to apply previous results, one needs to essentially show that key satisfying the safe predicate trivially leaked as a result of a user corruption during the execution. We do this by associating to each group key in the execution a *recovery graph*, made up of those keys that trivially allow recovery of the group key. Then, through a combinatorial argument, we show that if the safe predicate holds all keys ever leaked through a corruption cannot belong to the recovery graph of the challenge key. Security of the protocol thus follows using the aforementioned framework, in a fashion similar to that of previous works, such as [2].

1.3 Overcoming Lower Bounds

Proving lower bounds for important protocols serves several purposes. On the one hand, it can tell us when constructions *falling into the model of the lower-bound* cannot be further improved. As we identify a protocol that almost match our lower bound, this question is basically answered.

However, lower bound proofs can also hint as to where one should look for constructions overcoming them. One such possibility is to consider building blocks not captured by the bounds, or seemingly technical assumptions, which seem crucial for the lower-bound proofs to go through.

More Powerful Building Blocks. The symbolic model we consider (and which is captured by our combinatorial model) allows the basic primitives of PRFs or public-key encryption, and thus does not rule out protocols overcoming our lower bounds if they use more sophisticated tools.

The "big hammer" in this context is multiparty non-interactive key-agreement (mNIKE). With this primitive, each user could simply create a single message to be broadcast, after which any subset of users can locally compute a shared secret. While this overcomes our lower-bounds, it is just of theoretical interest, as currently no practical instantiations of mNIKE exist.

There already do exist CGKA protocols using primitives not captured by our model, in particular rTreeKEM [3] and DeCAF [7]. The variant rTreeKEM uses secretly-updatable public-key encryption [21] (skUPKE), but this primitive is used to improve the forward secrecy of the protocol, with no difference to the (asymptotic) communication cost of the protocol. The CGKA DeCAF also uses skUPKE, but in order to improve the round complexity for healing: instead of $\log(n)$ rounds as in CoCoA, DeCAF only needs $\log(c)$ rounds, with c being the number of users corrupted.

Note that our lower bound is independent of the number of corrupted users, but the proof argues based on an adversary which corrupts $c = \Theta(n)$ parties. In this setting DeCAF's cost matches that of CoCoA and thus adheres to our lower bound. However, under the promise that few, say constant, users are actually corrupted, DeCAF heals in a constant number of rounds with cost $\mathcal{O}(n\log(n))$.

Finally, two recent works [5,19], explore the use of multi-recipient multi-message PKE (mmPKE), which allows for much more efficient updates. However, the improvements save a constant factor in the ciphertext size, and do not have an influence in the asymptotic cost of the protocols.

Distributing Work. Our bound is restricted to schemes that do not "distribute the workload of communicating a secret on several users", in the following sense. We require that, if in any round a user gets access to a secret they did not previously possess, then they must have recovered it from a single update message, or sampled it themselves. All CGKA schemes we are aware of satisfy this property.

Nested Encryption. Finally, we require that users do not create layered ciphertexts, i.e., those of the form $\mathsf{Enc}(pk_1, \mathsf{Enc}(pk_2, m))$. Again, this is a property that is satisfied by all CGKA protocols we are aware of. This condition has a similar flavor as the one of distributing work, with the difference being that, instead of splitting the communication cost between several users, would enable a user to spread out communication cost over several rounds.

Publicly-Computable Update Cost. We show that there exists a sequence of updates such that the lower bound holds. However, this does not mean that the sequence can be found using only public information. We introduce this assumption to guarantee that the adversary can tell what cost a user will incur if asked to update in round t using only public information available at the end of round $t-1$ and this suffices to find the update sequence used in the proof of the lower bound. We also introduce a stronger version of this property, which we call offline publicly-computable update cost, that makes it possible to use public information available at the end of the initialization phase and the sequence of users who have performed updates in the previous rounds. While the strong property is satisfied by all protocols we are aware of, it is conceivable that protocols exist that overcome our bound, by having a user toss a coin when asked to update, with the outcome determining whether a "cheap" or "expensive" update is made.

1.4 Related Work

Protocols. The primitive of CGKA was introduced by Alwen *et al.* [3], but constructions existed earlier, notably ART [15] and TreeKEM [10]. These two were the starting point for the Message Layer Security (MLS) working group by the IETF. A variety of protocols have since been published, aiming to improve TreeKEM across different axes.

First, in the non-concurrent setting, [3] propose the use of UPKE in order to improve on FS; Klein *et al.* [22] propose an alternative way to handle dynamic operations with a lower communication cost in certain scenarios; Devigne, Duguey and Fouque [17] propose to use zero-knowledge proof to enhance the protocol robustness; Alwen *et al.* [1] initiates the study of efficiency of CGKAs in the multi-group setting; Hashimoto, Katsumata and Prest [20] provide a wrapper upgrading non-metadata-hiding CGKAs into metadata-hiding ones.

Concurrency was already mentioned in the initial TreeKEM versions, and indeed, as mentioned, its new versions allow for a certain degree of it. The first protocol to explore the idea was Weidner's Causal TreeKEM [23] with the idea of updates by re-randomizing (and combining) key material, instead of overwriting it. The work of Weidner *et al.* [26] puts forth the notion of decentralized CGKA. Alwen *et al.*'s CoCoA [2] analyzes a variant allowing for concurrent healing in $\log(n)$ rounds. A follow-up of this work by Alwen *et al.* [7] picked up the idea of [23] and extended it and formally analyzed it, showing that it allows for PCS in a logarithmic number of rounds in the number of corrupted parties.

Lowerbounds. The main approach is to make use of the symbolic security model, first introduced by Dolev and Yao [18] and later used by Micciancio and Panjwani [24] to prove worst case bounds on the update cost of multicast encryption schemes for a single group.

Regarding CGKAs, in the non-concurrent setting, Alwen *et al.* [1] provide lower bounds for the average update cost of an update in any CGKA protocol in the symbolic model, following and generalizing the approach of [24]. This shows TreeKEM or other related protocols are indeed optimal in this setting. In the concurrent setting, i.e. that where we consider the case of healing c corruptions in less than c rounds, the study of lower bounds was initiated by Bienstock, Dodis and Rösler [12], who establish lower bounds for protocols achieving PCS in exactly 2 rounds.[2] Last, Bienstock *et al.* [11] establish a lower bound on the cost of certain sequences of adds and removes. In particular, they show that any CGKA has a worst-case communication cost linear in the number of users.

[2] Here, we have a tradeoff between the time needed to achieve PCS, and the communication needed to do so. The picture is slightly more complicated, as in protocols like TreeKEM, or the protocol proposed in [12], the bigger tradeoff is in the increased cost of subsequent updates.

Security. Finally, security of CGKAs has been studied by multiple papers. Security against adaptive adversaries with a sub-exponential loss was first proved by Klein *et al.* [22]. Active security has been studied by Alwen *et al.* [4,6] in the UC model. PCS in the multi-group setting has been studied by Cremers, Hale and Kohbrok [16], who show shortcomings of a certain version of MLS compared to the (inefficient) pairwise-channels construction. Brzuska, Cornelissen and Kohbrok [13] apply the State Separating Proofs methodology to analyze the security of a certain version MLS.

2 Preliminaries

2.1 Definitions and Results from Combinatorics

Definition 1 (Minimal set cover). *Let $n \in \mathbb{N}$ and $\mathcal{S} \subseteq 2^{[n]}$. Then for $X \subseteq [n]$ we define the min cover of X with respect to $\mathcal{S}$. A* minimal set cover *(min cover)* $\text{minCover}_{\supseteq}(X, \mathcal{S})$ *of X with respect to $\mathcal{S}$ is a set $\mathcal{T} \subseteq \mathcal{S}$ of minimal cardinality such that $X \subseteq \bigcup_{T \in \mathcal{T}} T$, i.e., a minimal subset of $\mathcal{S}$ that covers X. Note that we only require S be contained in the union but no equality.*

Proposition 1 (Inequality of arithmetic and geometric means). *For $k \in \mathbb{N}$ let $x_1, \ldots, x_k \in \mathbb{R}$ be non-negative such that $\sum_{i=1}^{k} x_i = x$. Then*

$$\prod_{i=1}^{k} x_i \leq \left(\frac{x}{k}\right)^k \quad .$$

2.2 Continuous Group-Key Agreement

We now establish syntax for *continuous group-key agreement* (CGKA) schemes. A CGKA scheme allows a group G of users to agree on a group key that is to be used to secure communication within the group. In order to be able to recover from corruption users can also, possibly concurrently, send update messages, which rotate their key material. On top of this, CGKA schemes normally allow for group membership to evolve throughout the execution, by adding or removing users. However, while schemes allowing for theses additional operations are desirable in practice, the main goal of this work is to establish lower bounds on the communication complexity of recovering from corruption by concurrent updates. Thus, we restrict our view to static groups, i.e., we do not require the functionality of adding users to or removing users from the group. Not considering adds and removes allows for less technical notation, and we point out that lower bounds only profit from this restriction, as they hold even for schemes restricted to static groups. In doing so, our syntax essentially follows that of [12], with a couple of small differences mentioned below.

A continuous group-key agreement scheme CGKA specifies algorithms Setup, Init, Update, $\mathsf{Process}$, and GetKey. Algorithms Setup and Init can be used to initialize the a group G, that since we restrict our view to static groups, throughout this work we simply identify with $G = [n]$ for some $n \in \mathbb{N}$. Here, Setup is used to generate every user's initial internal state and can be thought of as the users generating a key pair and registering it with a PKI. Init, on the other hand, is called by one of the users to initialize the group. It generates a control message that, when processed by the other users, establishes the initial group-key K_G^0. Afterwards, the scheme proceeds in rounds t, in each of which a subset of users in G concurrently generate update messages using algorithm Update. The update messages are in turn processed by the group members resulting in a new group key K_G^t that can be recovered from a user's internal state using algorithm GetKey.

More formally,

- $\mathsf{Setup}(n; r)$ on input the group size n and random coins r belonging to randomness space Rnd outputs public information pub, as well as an initial state st_u for every user $u \in G = [n]$.
- $\mathsf{Init}(st_u, \mathit{pub}; r)$ receives as input a user's (initial) state, the public information pub, and random coins r. Its output (st'_u, MI_u) consists of the initializing user's updated state and a control message MI_u.
- $\mathsf{Update}(st_u, \mathit{pub}; r)$ in round t takes as input a user's current state, the public information pub, and random coins r. It returns updated state st'_u and a update message MU_u^t.
- Deterministic algorithm $\mathsf{Process}(st_u, \mathit{pub}, M)$ gets as input a user u's state, the public information pub, and a set M of control messages that either consists of a single group initialization message MI_v, or a family of update messages $(\mathit{MU}_v)_v$. Its output is the processing user's updated state st'_u.
- Deterministic algorithm $\mathsf{GetKey}(st_u)$ on input a user's state returns u's view of current group key K_G belonging to key space $\mathsf{CGKA.KS}$.

When comparing to the syntax of [12], one can find two differences. On the one hand, we chose not to merge algorithms Setup and Init, as in [12], although this would be possible since we consider the simple setting where a single static group is created. On the other hand, we include the algorithm GetKey, present in the original CGKA definition from [3]. This makes it easier to argue the

```
Game CORRECT$^{\mathsf{CGKA}}(n, u_0, (U_t)_{t=1}^{t_{\max}})$
$t \leftarrow 0$
$\mathsf{INIT}(n, u_0)$
while $t \le t_{\max}$:
    $t \leftarrow t + 1$
    $\mathsf{ROUND}(U_t)$
return $0$

Oracle $\mathsf{INIT}(n, u_0)$
$(\mathit{pub}, (st_u)_{u \in [n]}) \leftarrow \mathsf{Setup}(n)$
$(st_{u_0}, \mathit{MI}_{u_0}^0) \leftarrow \mathsf{Init}(st_{u_0}, \mathit{pub}, n; r)$
for $u \in [n]$:
    $st_u \leftarrow \mathsf{Process}(st_u, \mathit{pub}, \mathit{MI}_{u_0}^0)$
    $K_G^u \leftarrow \mathsf{GetKey}(st_u)$
if $\exists u, v \in [n] : K_G^u \neq K_G^v$:
    return $1$

Oracle $\mathsf{ROUND}(U_t)$
$\mathit{MU} \leftarrow \emptyset$
for $u \in U_t$:
    $(st_u, \mathit{MU}_u^t) \leftarrow \mathsf{Update}(st_u, \mathit{pub})$
    $\mathit{MU} \leftarrow_{\cup} \mathit{MU}_u^t$
for $u \in [n]$:
    $st_u \leftarrow \mathsf{Process}(st_u, \mathit{pub}, \mathit{MU})$
    $K_G^u \leftarrow \mathsf{GetKey}(st_u)$
if $\exists u, v \in [n] : K_G^u \neq K_G^v$:        \\disagreement on group key
    return $1$
```

Fig. 3. Correctness game for continuous group-key agreement scheme CGKA.

connection between the combinatorial and symbolic models. The two main properties that we require from a CGKA scheme are correctness and security.

Correctness. For correctness we require that, for every valid sequence of operations, in every round t, all users agree on the current group key K_G^t where $G = [n]$ is a static group of size $n \in \mathbb{N}$. We formalize the notion of correctness in the game of Fig. 3. The game gets as input n, the user $u_0 \in G$ initializing the group, and a sequence $(U_t)_t$ of updates to be applied in every round where $U_t \subseteq G$. The game returns the value 0 if the execution was correct and 1 otherwise. Accordingly, we say that a scheme CGKA is *perfectly correct* if, for every input $(n, u_0, (U_t)_{t=1}^{t_{\max}})$ and all choices of random coins, we have that $0 = \mathrm{CORRECT}^{\mathsf{CGKA}}(n, u_0, (U_t)_{t=1}^{t_{\max}})$. For a perfectly correct scheme, we denote the current group key by K_G, instead of K_G^u since all users agree on it.

Security. For security, we require that the group key of a CGKA scheme recovers from corruption assuming that every party did at least k updates since their last corruption. More formally, we consider the security notions of indistinguishability of the group key from random (IND-k-PCS$_{\text{mode}}$), and one-wayness (OW-k-PCS$_{\text{mode}}$). Here, mode $\in \{\neg\mathrm{RC}, \mathrm{RC}\}$ indicates whether the corruption of a user reveals only their private state in the current round, or also additionally the random coins they sampled in the round. Thus, we end up with 4 different

```
Game IND-$k$-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$
    $b^* \leftarrow_\$ \{0,1\}$
    $b \leftarrow \mathsf{A}^{\mathsf{INIT},\mathsf{ROUND},\mathsf{CORR},\mathsf{CHALL}}$
    return $[b = b^*] \wedge \text{safe}_{k\text{-PCS}}$

Game OW-$k$-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$
    $K \leftarrow \mathsf{A}^{\mathsf{INIT},\mathsf{ROUND},\mathsf{CORR},\mathsf{CHALL}}$
    return $[K^* = K] \wedge \text{safe}_{k\text{-PCS}}$

Oracle INIT$(n, u_0)$                    \\one call; called first
    $t \leftarrow 0$
    $\mathrm{Cor}[u] \leftarrow \emptyset$, $\mathrm{Upd}[u] \leftarrow \emptyset$ for all $u \in [n]$
    $\mathrm{Coins}[u,t] \leftarrow \emptyset$ for all $u \in [n]$, $\forall t$
    $r_{\mathbf{setup}} \leftarrow_\$ Rnd$
    $(pub, (st_u)_{u\in[n]}) \leftarrow \mathsf{Setup}(n; r_{\mathbf{setup}})$
    $r \leftarrow_\$ Rnd$; $\mathrm{Coins}[u_0,t] \leftarrow_\cup \{r\}$
    $(st_{u_0}, MI^0_{u_0}) \leftarrow \mathsf{Init}(st_{u_0}, pub, n; r)$
    for $u \in [n]$:
        $st_u \leftarrow \mathsf{Process}(st_u, pub, MI^0_{u_0})$
        $K_G \leftarrow \mathsf{GetKey}(st_u)$
    return $pub, MI^0_{u_0}$

Oracle CHALL          \\one call, Game IND-$k$-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$
    $t^* \leftarrow t$
    $K_0 \leftarrow_\$ \mathsf{CGKA.KS}$; $K_1 \leftarrow K_G$
    return $K_{b^*}$

Oracle CHALL          \\one call, Game OW-$k$-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$
    $t^* \leftarrow t$
    $K^* \leftarrow K_G$

Oracle ROUND$(U_t)$
    $t \leftarrow t + 1$
    $MU \leftarrow \emptyset$
    for $u \in U_t$:
        require $u \in [n]$
        $\mathrm{Upd}[u] \leftarrow_\cup \{t\}$
        $r \leftarrow_\$ Rnd$; $\mathrm{Coins}[u,t] \leftarrow_\cup \{r\}$
        $(st_u, MU^t_u) \leftarrow \mathsf{Update}(st_u, pub; r)$
        $MU \leftarrow_\cup MU^t_u$
    for $u \in [n]$:
        $st_u \leftarrow \mathsf{Process}(st_u, pub, MU)$
        $K_G \leftarrow \mathsf{GetKey}(st_u)$
    return $MU$

Predicate safe$_{k\text{-PCS}}$
    for $u \in [n]$:
        if $\mathrm{Cor}[u] \neq \emptyset$
            if $\exists t \in \mathrm{Cor}[u] : t \geq t^*$:          \\corruption after challenge
                return 0
            $t^c_u \leftarrow \max\{t \in \mathrm{Cor}[u]\}$
            if $|\{t \in \mathrm{Upd}[u] : t^c_u < t \leq t^*\}| < k$:        \\too few updates
                return 0
    return 1

Oracle CORR$(u)$
    $\mathrm{Cor}[u] \leftarrow_\cup \{t\}$
    return $(st_u, \mathrm{Coins}[u,t])$                  \\if mode = RC
    return $st_u$                                   \\if mode = ¬RC
```

Fig. 4. Security games IND-k-PCS$_{\text{mode}}$ for indistinguishability and OW-k-PCS$_{\text{mode}}$ for one-wayness of group keys with respect to mode $\in \{\mathrm{RC}, \neg\mathrm{RC}\}$. The game is defined with respect to a scheme CGKA and an adversary A. We require that the adversary's first call is to oracle INIT, which can only be queried once.

security notions. The weakest, OW-k-PCS$_{\neg\text{RC}}$, is used for our lower bounds in Sect. 3.2 and the strongest, IND-k-PCS$_{\text{RC}}$, for our upper bound of Sect. 5.

The security games are formally defined in Fig. 4. They provide the adversary A with an initialization oracle INIT that allows for a single query, that has to be made before using any of the other oracles. It enables A to set up a universe of users and initialize a group. Using oracle ROUND, the adversary can specify sets of users to concurrently perform updates. All operations are then processed by the members of the group, and the round counter t is increased. Further, A can, at any point in time, use the corruption oracle $\mathsf{CORR}(u)$ to reveal user u's current internal state st_u, and, in the case that mode = RC, additionally the random coins u sampled in the current round while updating. Finally, A, at an arbitrary point in time t^*, can make a single query to the challenge oracle CHALL, which in Game IND-k-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$, depending on challenge bit b^*, returns either the current group key or a uniformly random key. The adversary wins if it is able to correctly guess b^* and safety predicate safe-k-PCS holds. In Game OW-k-PCS$^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A})$, the oracle instead stores the current group key as challenge key K^*. This has to be computed by A in order to win, again with the restriction that safe-k-PCS holds. The predicate safe-k-PCS verifies that, for every user that at time t^* is a member of the group, (a) they were never corrupted after t^* and (b) since their last corruption before t^* they performed at least k updates.

Definition 2 (k-PCS security). *Let* CGKA *be a continuous group-key agreement scheme,* $k \in \mathbb{N}$, *and* mode $\in \{\text{RC}, \neg\text{RC}\}$. *Then* CGKA *is* IND-k-PCS$_{\text{mode}}$ secure, *if for every PPT adversary the advantage function* $|\Pr[\text{IND-}k\text{-PCS}^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A}) \Rightarrow 1 \mid b^* = 1] - \Pr[\text{IND-}k\text{-PCS}^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A}) \Rightarrow 1 \mid b^* = 0]|$ *is negligible.*

Further, CGKA *is* OW-k-PCS$_{\text{mode}}$ secure, *if for every PPT adversary the advantage function* $\Pr[\text{OW-}k\text{-PCS}^{\mathsf{CGKA}}_{\text{mode}}(\mathsf{A}) \Rightarrow 1]$ *is negligible.*

Remark 1. We make the following observation about the security model.

(i) In this work we are interested in the communication cost of achieving post-compromise security, and thus ignore attacks breaching forward secrecy, i.e., learning group keys from previous rounds by corrupting users. This is encoded in lines 35 and 36 of the safe predicate, which disallow corrupting users after the challenged round t^*.

(ii) Our security model is quite weak. In particular, all initialization and update operations are honestly generated and immediately processed by all users in synchronous rounds. We point out that this only strengthens our lower bounds, as they hold even for a security notion far weaker than what one would aim for in practice. While this leaves open the possibility of improving on our bounds by switching to a stronger security notion, we point out that they are closely matched by the upper bound of Sect. 5, which we expect to be easily made secure in asynchronous settings with a semi-honest server using standard techniques to ensure consistency (e.g. signatures, a key schedule, transcript and parent hashes, etc. [2,6,10]).

Restrictions. Our lower bounds apply to CGKA schemes CGKA satisfying the following two restrictions.

- CGKA does not use nested encryption (NNE). This means that users do not create layered ciphertexts of the form $\mathsf{Enc}(pk_1, \mathsf{Enc}(pk_2, m))$.
- CGKA does not distribute work (NDW). This means that, if in any round a user get access to a secret they did not previously possess, then they must have either sampled it by themselves or recovered it from the update message of a single user.

The properties are not directly exploited in our proofs in the combinatorial model. Instead, we use them to show that bounds in the combinatorial model also hold in the symbolic model. We defer the restrictions' formal definitions to Sect. 4 (Definition 5 and Definition 7), where we will also formally justify their impact on the combinatorial model. We point out that all CGKA schemes that we are aware of satisfy both properties.

We also consider an additional property. We say that CGKA has publicly-computable update cost (PCU) if it is always possible to determine the size $|MU_u|$ of an update that a user u would produce if asked to update given access only to public information, i.e., pub, as well as the sets of update messages sent so far. With this additional property we can show that not only there exists a sequence of updates for which the total communication cost is at least roughly $n^{1+1/k} \cdot k/\log(k)$, but it is also possible to find the sequence using only public information. Formally, CGKA schemes with publicly-computable update cost are defined as follows.

Definition 3 (Publicly-computable update cost). *Let* CGKA *be a CGKA scheme. Consider an execution of game* IND-k-PCS$_{\text{mode}}$ *(or* OW-k-PCS$_{\text{mode}}$*). We say that* CGKA *has* publicly-computable update cost *if, for every round* t *and for every user* $u \in G_t$ *with internal state* st_u^t *and public information* pub*, it is possible to efficiently compute* $|MU|$*, where* $(st', MU) \leftarrow \mathsf{Update}(st_u^t, pub; r)$ *would be the output of calling the update procedure, from public information at the end of round* $t-1$ *(i.e., all messages* $MI_{u_0}^0$ *and* $MU_u^{t'}$ *sent in any round* $t' \leq t-1$*, the sequence* $(U_{t'})_{t' \leq t-1}$*,* n*,* k*,* pub *and* u_0*). Note that, in particular, the size of* MU *must be independent of the random coins* r *used to generate the update message. We say that* CGKA *has* offline publicly-computable update cost *if the same property holds using only the initialization messages* $MI_{u_0}^0$*, the sequence* $(U_{t'})_{t' \leq t-1}$*,* n*,* k*,* pub *and* u_0*.*

All CGKA schemes that we are aware of have offline publicly-computable update cost. For example, for schemes based on ratchet trees, as for example TreeKEM or CoCoA, the size of every user's next update is fully determined by the position of blank and non-blank nodes in the ratchet tree, which can be determined given just the sequence of update/propose-commit operations.

3 Lower Bounds in the Combinatorial Model

In this section we define a self-contained combinatorial model capturing CGKA schemes recovering from corruption in k rounds of updates and then prove a lower bound on the communication complexity of such schemes. The model is given in Sect. 3.1, the bound in Sect. 3.2.

3.1 The Combinatorial Model

We now present a purely combinatorial model capturing an adversary interacting with a correct and secure CGKA scheme built from public-key encryption and pseudorandom functions. The interaction proceeds in rounds, during which users schedule update operations and at the end of which a set of users is corrupted.

High-Level Structure. An instance of the combinatorial model is characterized by a tuple $(n, k, t_{\max}, C_0)$ and a sequence $(U_t, C_t)_{t=1}^{t_{\max}}$, where $n, k, t_{\max} \in \mathbb{N}$, $C_0 \subseteq [n]$, and $U_t, C_t \subseteq [n]$ for all t. This corresponds to setting up the group $G = [n]$ and in round 0 corrupting the set of users C_0. The sequence $(U_t, C_t)_{t=1}^{t_{\max}}$ determines the operations performed in the following $t_{\max}$ rounds, where

- U_t is the set of users performing an update in round t, and
- C_t is the set of users corrupted at the end of round t.

Integer k determines the safety requirement imposed on the CGKA scheme. More precisely, we aim to capture CGKA schemes that recover from corruption after k updates, meaning that if every user did at least k updates since the last round in which they were corrupted, then the group must agree on a secure key. Formally, consider an instantiation of the combinatorial model with respect to $(n, k, t_{\max}, C_0)$ and $(U_t, C_t)_t$ as described above. We say that a round $t \in \{0, \dots, t_{\max}\}$ is *safe*, if for every user $u \in G$ such that $u \in C_{t'}$ for some t' there exist rounds $t_1, \dots, t_k$ such that

$$u \in U_{t_i} \text{ for all } i \in \{1, \dots, k\} \tag{1}$$

and

$$\max\{t_c \in \{0, \dots, t_{\max}\} : u \in C_{t_c}\} < t_1 < \dots < t_k \leq t \ . \tag{2}$$

Recall that since we want to only argue about post-compromise security but not forward-secrecy the condition also excludes the corruption of users *after* round t.

Set System and Cost Function. The main intuition behind the combinatorial model is to associate the secure PKE and PRF keys present in the CGKA scheme in round t to the set of users in G that have access to them at this point in time, i.e., can recover them from their current internal state. Here 'secure key' refers to keys that were established by update operations and cannot be trivially recovered from the adversary. The adversary is able to get access to keys directly by corrupting users' states, or by recovering them from protocol messages. The latter is possible if the message contains an encryption of the key under a key the adversary has access to, or if it contains the key in plain. In every round the sets $S(sk)$ of users having access to secure keys sk form a subset of 2^G. Intuitively, security and correctness of a CGKA scheme imply that the system

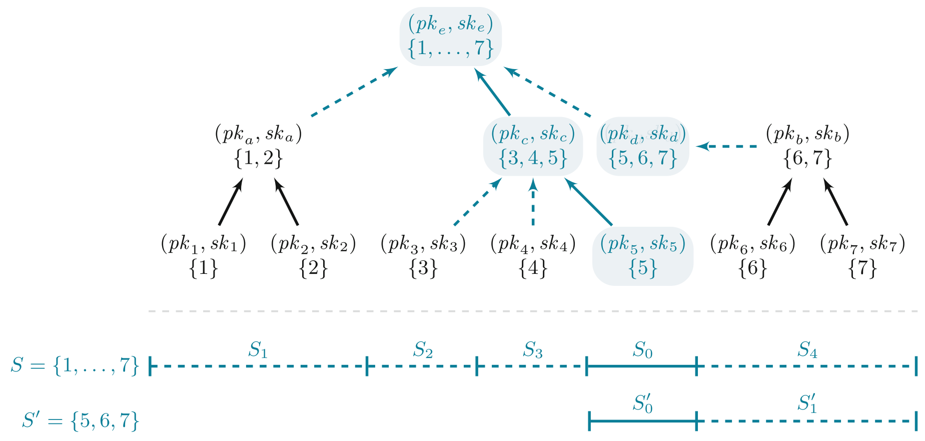

Fig. 5. Top: Illustration of a ratchet tree and its associated set system $\mathcal{S}_t$. Vertices contain key-pairs (above) and the associated set (below). Keys already present in the system at time $t-1$ are depicted in black and keys added by user 5 in round t in blue with shaded background. Edges indicate that knowledge secret key of the source implies knowledge of the one of the sink. Dashed, blue edges correspond to ciphertexts $\mathsf{Enc}_{pk_{\text{source}}}(sk_{\text{sink}})$ sent by user 5 in round t, solid edges either to keys derived using a PRF in round t (depicted in blue) or to keys communicated in a previous round (depicted in black). Accordingly, user 5 generated key-pairs (pk_5, sk_5) and (pk_d, sk_d) using fresh randomness, and (pk_c, sk_c) and (pk_e, sk_e) using a PRF. Bottom: Depiction of the sets required to exist by property (iii) using the examples $S = \{1,\dots,7\} = \bigcup_{i=0}^{k} S_i$ and $S' = \{5,6,7\} = \bigcup_{i=0}^{k'} S'_i$ with $k = 4$ and $k' = 1$ corresponding to the secret keys sk_e and sk_d respectively. We have $S_0 = \{5\} = S'_0$, $S_1 = \{1,2\}$, $S_2 = \{3\}$, $S_3 = \{4\}$, and $S_4 = \{6,7\} = S'_1$. Note that the number of ciphertexts sent to communicate the secret keys corresponding to S and S' to their members are $5 > k$ and $1 = k'$ respectively, thus satisfying the inequality on the user's cost function required by property (iii). (Color figure online)

of associated sets should satisfy certain properties, and that adding sets to it by scheduling updates comes at the cost of sending ciphertexts. These properties are stated below and, looking ahead, will serve as the main tools to derive our lower bound. For an illustration of the set system corresponding to a ratchet tree as described in the introduction see Fig. 5 (Top).

Formally, consider an instantiation of the combinatorial model with respect to $(n, k, t_{\max}, C_0)$ and $(U_t, C_t)_t$. We require the existence of a cost function Cost and a sequence $(\mathcal{S}_t)_{0 \le t \le t_{\max}}$ of set systems $\mathcal{S}_t \subseteq 2^G$. The cost function and sequences are required to satisfy three properties to be given further below. The cost function takes as input

- the user $u \in G$ performing the update operation,
- the round t with $1 \le t \le t_{\max}$, and
- the history $M_t = (n, k, (U_{t'})_{1 \le t' < t})$ of sets of users performing updates in the previous rounds.

Its output is an integer $\mathrm{Cost}(u, t, M_t)$. For better legibility, we will simply write $\mathrm{Cost}(u, t)$ whenever the third input is clear from context.

Note that while the cost of a user's update in a given round depends on the operations performed in previous rounds, it does not depend on the sets $C_{t'}$ of users corrupted in previous rounds. The latter is justified, since, looking ahead, in the security game in the symbolic model, users are not aware whether they are corrupted or not. However, if asked to update by the adversary, they may decide to create particular ciphertexts depending on the history of operations performed so far, as these may have impacted their internal state.

Requirements on Set System and Cost Function. We now give three properties to be satisfied by the cost function and the set system.

(i) Correctness of the CGKA scheme implies that group members share a common key. Further, by security, whenever a round is safe, the corresponding shared key must not be known to the adversary at this point in time.
Formally, if round t is safe we require that $G = [n] \in \mathcal{S}_t$.

(ii) If a user is corrupted in some round, all keys they currently have access to can also be recovered by the adversary and therefore should be considered insecure. This is represented by $\mathcal{S}_t$ not containing any sets that include a party corrupted in round t.
Formally, for all $t \in \{0, \dots, t_{\max}\}$ and all $u \in C_t$ we have that $S \in \mathcal{S}_t$ implies that $u \notin S$.

(iii) The third property captures how users agree on new keys when using basic cryptographic primitives (PRFs and PKE) and which cost in terms of ciphertexts sent is incurred by communicating these keys to other users. A user $u \in G$ can always sample a new key locally. Further, from such a key or one already present in the system they can derive a chain of new keys using PRF evaluations. To communicate the key sk to other users they can encrypt it under a public key pk' that must have either been present in the system at the end of round $t-1$ or been previously generated by u in round t. From the resulting ciphertext, every user with access to the corresponding sk' is able to derive sk as well as all keys derived from sk using PRF evaluations. Note that if sk' is insecure, then the adversary can recover sk.

In terms of sets this essentially means that the set $S \in \mathcal{S}_t$ of users able to recover sk can be covered by a union of sets in $\mathcal{S}_{t-1}$ (and potentially a singleton $\{u\}$ in case user u generated the starting point of the PRF evaluation chain from fresh randomness) and that the cost of the user communicating sk to the other members of S should be at least the number of sets forming the union (where sometimes one ciphertext can be saved, as the key serving as a starting point of a chain of PRF evaluations needs not be communicated).
Formally, for every $t \geq 1$ and every $S \in \mathcal{S}_t$ we require that there exist $h \in \mathbb{N}_{\geq 0}$ and $S_0, \dots, S_h$, such that either

$$S_0 = \{u\} \text{ for some } u \in U_t \quad \text{or} \quad S_0 \in \mathcal{S}_{t-1} \ , \tag{3}$$

and if $h \geq 1$ then $S_i \in \mathcal{S}_{t-1}$ for all $i \in \{1, \ldots, h\}$. Further, we require that

$$S \cap C_{\leq t} \subseteq \bigcup_{i=0}^{h} S_i \tag{4}$$

where $C_{\leq t} = \bigcup_{0 \leq t' \leq t} C_{t'}$ are the users that have been corrupted at least once in or before round t. And, regarding the cost function, we require that

$$\exists u \in S_0 \cap U_t \text{ such that } \mathrm{Cost}(u, t) \geq h \ . \tag{5}$$

Note that if in Eq. 3 we have $S_0 = \{u\}$ then the user in Eq. 5 must be u. Finally, we can connect the cost of adding a set to the set system $\mathcal{S}_t$ to its MinCover with respect to $\mathcal{S}_{t-1}$. Indeed, for $S \in \mathcal{S}_t$, if u is the user required to exist by Eq. 5, then by Eqs. 3, 4, and 5

$$\mathrm{Cost}(u, t) \geq |\mathrm{minCover}_{\supseteq}(S \cap C_{\leq t}, \mathcal{S}_{t-1} \cup \{u\})| - 1 \ . \tag{6}$$

The precise connection between the combinatorial model and the symbolic model is established in Sect. 4. There, we essentially show that an adversary playing the OW security game in the symbolic model with respect to a correct and secure CGKA scheme that satisfies the restrictions described in Sect. 2.2 implies the existence of a set system $\mathcal{S}_t$ satisfying Properties (i)–(iii) if one uses the number of ciphertexts sent by a user u in round t as cost function $\mathrm{Cost}(u, t)$.

3.2 Lower Bound in the Combinatorial Model

We now give a lower bound on the communication cost required to recover from compromise within k rounds in the combinatorial model. Conceptually, our proof proceeds in two steps. First, we lower bound the sum of the maximal per-user update cost over all rounds. This bound is a best-case bound, i.e., it holds with respect to every sequence $(U_t)_t$ of updating users. In a second step we then prove our main result, a bound on the total cost required to recover from corruption. This bound is worst case, i.e., it holds with respect to an adversarially chosen sequence of updating users. Concretely, we will exploit that the cost of a user $u \in U_t$ updating in round t does not depend on the cost of other members of U_t updating concurrently. This enables us to find a sequence $(U_t)_t$ for which all members of U_t have roughly the same update cost, which yields the desired bound as the bound on the maximal per-user update cost implies that the cost of the users in U_t in sufficiently many rounds t must be quite large.

Lower Bound on the Maximal Per-user Update Cost. We first consider the scenario that after an arbitrary setup phase of t_c rounds a set of c users in $G = [n]$ is corrupted and that after m subsequent rounds of updates we have $G \in \mathcal{S}_t$ (intuitively corresponding to the existence of a secure group key). Below, we bound the sum of the maximal per-user update cost over the m rounds. Note that this bound holds irrespective of how the sets U_t of updating users are chosen. The proposition's proof is in the full version of this paper [8].

Proposition 2. *Let $n, k, t_c, m \in \mathbb{N}$, $C \subseteq [n]$, and $c = |C|$ such that $\ln(c) \geq m - 1$. Let $t_{\max} = t_c + m$ and consider an instantiation of the combinatorial model with respect to $(n, k, t_{\max}, C_0)$, $(U_t, C_t)_t$, where $C_{t_c} = C$, $C_t = \emptyset$ for $t \neq t_c$, and $(U_t)_t$ is an arbitrary sequence.*

If $G = [n]$ is contained in the set-system $\mathcal{S}_{t_{\max}}$ at the end of round $t_{\max} = t_c + m$, then we have

$$\sum_{t=1}^{m} \max_{u \in U_{t_c+t}} \mathrm{Cost}(u, t_c + t) \geq (m-1)\left(\sqrt[m-1]{c} - 1\right).$$

From Maximal Per-user Cost to Total-Communication Cost. We now show that for an adversarially chosen sequence $(U_t)_t$ of sets of updating users actually almost all users have to adhere to the bound derived in the previous paragraph. Intuitively, after an arbitrary warm up phase of t_c rounds and corrupting a linear fraction of users in round t_c, we construct $(U_t)_t$ such that either all updating users have roughly the same update cost, or all users have a very small update cost. This procedure will then be repeated for sufficiently many rounds to force that a linear fraction of all users in the group has updated at least k times. In this case the final round $t_{\max}$ must be secure enforcing that $G \in \mathcal{S}_{t_{\max}}$. This allows us to use the bound derived in the previous paragraph to show that the communication cost of rounds corresponding to the former case must be substantial. We obtain the following theorem, its proof, as well as the one of the following corollary, being in the full version of this paper [8].

Theorem 3. *Let $k, n, t_c \in \mathbb{N}$ and $0 < \varepsilon < 2/5$ be a constant such that $(1+\varepsilon)k \in \mathbb{N}$. Set $\alpha_\varepsilon = \frac{\varepsilon - 5/2\varepsilon^2 + \varepsilon^3}{8(1+\varepsilon)} > 0$ and $t_{\max} = t_c + (1+\varepsilon)k$. If $3 \leq k \leq \ln(\alpha_\varepsilon n)$, then for every sequence $(U_t)_{t=1}^{t_c}$ there exists a set $C \subseteq [n]$ of size $\lceil \alpha_\varepsilon n \rceil$ and a sequence $(U_t)_{t=t_c+1}^{t_{\max}}$ such that the instantiation of the combinatorial model with respect to $(n, k, t_{\max}, \emptyset)$ and $(U_t, C_t)_{t=1}^{t_{\max}}$, where $C_{t_c} = C$ and $C_t = \emptyset$ if $t \neq t_c$, satisfies*

$$\sum_{t=1}^{(1+\varepsilon)k} \mathrm{Cost}(U_{t_c+t}) \geq \frac{(k-1)}{4} \cdot \left\lfloor \frac{2\varepsilon n}{5(1+\varepsilon)\lceil \log(k) \rceil} \right\rfloor \left((\alpha_\varepsilon n)^{\frac{1}{(1+\varepsilon)k-1}} - 1 \right).$$

While we phrased Theorem 3 as a single-stage experiment, i.e., only consider the communication required to recover from corruptions made in a single round, it easily carries over to a repeated experiment consisting of repeatedly corrupting a linear fraction of the users from which the group has to recover within $(1+\varepsilon)k$ rounds of updates. Note, that the setting of Theorem 3 allows for an arbitrary setup phase $(U_t)_{t \leq t_c}$ of t_c rounds. Thus, by simply applying the arguments in the proof iteratively to each recovery phase, we obtain that the derived bound holds even in an amortized sense, i.e., even in this setting the recovery from each corruption requires communication of order $nk \sqrt[(1+\varepsilon)k]{n}/\log(k)$.

Corollary 4. *Let k, n, t_c, ε, and α_ε be as in Theorem 3. Let $z_{\max} \in \mathbb{N}$ and for $0 \leq z < z_{\max}$ set $t_{c,z} = t_c + z \cdot (t_c + (1+\varepsilon)k)$ and $t_{\max} = z_{\max} \cdot (t_c + (1+\varepsilon)k)$. For all* *collections of sequences* $(U_t)_{t=t_{c,z}-t_c+1}^{t_{c,z}}$ *with* $0 \le z < z_{\max}$*, there exist sets* $C_z \subseteq [n]$ *each of size* $\lceil \alpha_\varepsilon n \rceil$ *and collections of updates* $(U_t)_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k}$ *such that for every instantiation of the combinatorial model with respect to* $(n, k, t_{\max}, \emptyset)$ *and* $(U_t, C_t)_{t=1}^{t_{\max}}$*, where* $C_{t_{c,z}} = C_z$ *and* $C_t = \emptyset$ *if* $t \notin \{t_{c,z} \mid 0 \le z < z_{\max}\}$*, we have*

$$\sum_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k} \mathrm{Cost}(U_{t_{c,z}+t}) \ge \frac{(k-1)}{4} \cdot \left\lfloor \frac{2\varepsilon n}{5(1+\varepsilon)\lceil \log(k) \rceil} \right\rfloor \left((\alpha_\varepsilon n)^{\frac{1}{(1+\varepsilon)k-1}} - 1 \right)$$

for very $0 \le z < z_{\max}$.

4 Lower Bounds in the Symbolic Model

In this section define CGKA in a symbolic model, an approach introduced for public key encryption by Dolev and Yao [18], following the work on multicast encryption by Micciancio and Panjwani [24], and generalized to CGKA by Bienstock, Dodis, and Rösler [12], who considered concurrent updates for schemes recovering in two rounds. A similar model was also used to lower bound the communication incurred by users in CGKA schemes in order to achieve PCS, in a setting of multiple groups [1]. We show how the questions we are interested in can be translated from the symbolic to the combinatorial model of Sect. 3, which allows us to conclude that the bounds derived in the combinatorial model also hold with respect to the symbolic model.

4.1 The Symbolic Model

We consider schemes constructed from pseudorandom functions and public-key encryption, both modeled as idealized primitives that take as input symbolic variables, and output symbolic variables. To more easily distinguish these from non-symbolic variables we use typewriter font. We use the following syntax.

(i) *Pseudorandom function:* Algorithm PRF takes as input a key $\mathtt{K}$ and a message $\mathtt{m}$ and returns a key $\mathtt{K}' = \mathsf{PRF}(\mathtt{K}, \mathtt{m})$.

(ii) *Public-key Encryption:* A PKE scheme consists of algorithms $(\mathsf{PKE.Gen}, \mathsf{PKE.Enc}, \mathsf{PKE.Dec})$, where $\mathsf{PKE.Gen}$ on input of secret key $\mathtt{sk}$ returns the corresponding public key $\mathtt{pk}$. $\mathsf{PKE.Enc}$ takes as input a public key $\mathtt{pk}$ and a message $\mathtt{m}$, and outputs a ciphertext $\mathtt{c} \leftarrow \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m})$ with message data type. $\mathsf{PKE.Dec}$ takes as input a secret key $\mathtt{sk}$ and a ciphertext $\mathtt{c}$, and outputs a message $\mathtt{m} = \mathsf{PKE.Dec}(\mathtt{sk}, \mathtt{c})$. We assume perfect correctness: $\mathsf{PKE.Dec}(\mathtt{sk}, \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m})) = \mathtt{m}$ for all $\mathtt{sk}$, $\mathtt{pk} = \mathsf{PKE.Gen}(\mathtt{sk})$, and messages $\mathtt{m}$.

As data types, we consider messages, public keys, secret keys, symmetric keys, and random coins, the latter being a terminal type. Which variables can be recovered from a set of messages $\mathtt{M}$, is captured by the *entailment relation* $\vdash$.

Data type		Grammar rules
Message $\mathtt{m}$	$\leftarrow$	$\mathtt{sk}, \mathtt{pk}, \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m})$
Public key $\mathtt{pk}$	$\leftarrow$	$\mathsf{PKE.Gen}(\mathtt{sk})$
Secret key $\mathtt{sk}$	$\leftarrow$	$\mathtt{K}$
Key $\mathtt{K}$	$\leftarrow$	$\mathtt{r}, \mathsf{PRF}(\mathtt{K}, \mathtt{m})$
Random coin $\mathtt{r}$		terminal type

Entailment relation		
$\mathtt{m} \in \mathtt{M}$	$\Rightarrow$	$\mathtt{M} \vdash \mathtt{m}$
$\mathtt{M} \vdash \mathtt{m}, \mathtt{pk}$	$\Rightarrow$	$\mathtt{M} \vdash \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m})$
$\mathtt{M} \vdash \mathtt{K}$	$\Rightarrow$	$\mathtt{M} \vdash \mathsf{PRF}(\mathtt{K}, \mathtt{m})$ for all $\mathtt{m}$
$\mathtt{M} \vdash \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m}), \mathtt{sk} : \mathtt{pk} = \mathsf{PKE.Gen}(\mathtt{sk})$	$\Rightarrow$	$\mathtt{M} \vdash \mathtt{m}$

Note that the entailment relation captures (ideal) correctness and (ideal) security of PRF and PKE, as recovering a PRF output or an encrypted message from a ciphertext requires knowledge of the secret key. Security is effectively captured by the of a sequence of entailment relations that recover the appropriate message. Examples and further comments (in the setting of multicast encryption) can be found in [24, Sect. 3.2]. The set of messages which can be recovered from $\mathtt{M}$ using relation $\vdash$ is denoted by $\mathsf{Der}(\mathtt{M}) := \{\mathtt{m} : \mathtt{M} \vdash \mathtt{m}\}$.

We point out that the model of [12] covers more primitives, concretely, dual PRFs, updatable PKE, and broadcast encryption. It is an interesting open question to consider whether a translation to our combinatorial model is also possible if one takes these additional primitives into account. For a brief discussion on challenges to overcome if one would allow dual PRFs see Remark 3.

Continuous Group-Key Agreement in the Symbolic Model. A CGKA scheme CGKA in the symbolic model follows the syntax of Sect. 2.2. Additionally, we require some of the inputs to CGKA's algorithms to be symbolic variables. Concretely, we require that the group keys $\mathtt{K}$, public and internal states $\mathtt{pub}$ and $\mathtt{st}$, random coins $\mathtt{r}$ as well as the control messages $\mathtt{MI}$ and $\mathtt{MU}$ are symbolic. They can also have a non-symbolic counterpart which we omit as the properties we study and the security game we consider in the symbolic model do not depend on the non-symbolic variables. However, we often distinguish between symbolic random coins $\mathtt{r}$ and non-symbolic randomness r as this is used in some of the proofs. Intuitively, symbolic randomness represents the new secrets being sampled, while non-symbolic randomness allows to capture the fact that the algorithms may flip a coin in order to determine their actions (e.g., the update algorithm might flip random coins to decide whether to generate certain ciphertexts or not). Further, we assume that the context symbolic variables, e.g., which key corresponds to a certain ciphertext, or which keys correspond to a particular set of users, are implicitly known to the algorithms.

We use the game of Fig. 3 to define correctness of CGKA, where we additionally require that, for every algorithm, each of its symbolic outputs can be derived from its symbolic inputs using the entailment relation $\vdash$. E.g., if user u computes $(\mathtt{st}'_u, \mathtt{MU}_u) \leftarrow \mathsf{Update}(\mathtt{st}_u, \mathtt{pub}; \mathtt{r}, r)$, then we require that $\mathtt{st}'_u, \mathtt{MU}_u \in \mathsf{Der}(\{\mathtt{st}_u, \mathtt{pub}, \mathtt{r}\})$, and similarly if $\mathtt{st}'_u \leftarrow \mathsf{Process}(\mathtt{st}_u, \mathtt{pub}, \mathtt{M})$ then it must hold that $\mathtt{st}'_u \in \mathsf{Der}(\{\mathtt{st}_u, \mathtt{pub}, \mathtt{M}\})$.

Regarding security, we target the notion of OW-k-PCS$_{\neg\mathrm{RC}}$ of Definition 2. As our goal is to prove lower bounds, using one-wayness as the targeted security notion only makes our results stronger compared to using indistinguishability.

We structure the game in rounds, that correspond to the oracle calls that occur between two subsequent calls to oracle ROUND. We say a query to some oracle was made in round 0 if it was made before the first query to ROUND, and in round t for $t \in \{1, \ldots, t_{\max}\}$, if it was either the tth query to ROUND, or, for calls to CHALL or CORR, if it was made after the tth and before the $(t+1)$st query to ROUND. This allows us to fully characterize adversaries A by the sequence of inputs to the oracles made in each round. For round 0, these are the input (n, G_0, u_0) to INIT and the set C_0 of corrupted users; for round t, the set U_t of updating users queried to ROUND, as well as the set C_t of users corrupted during the round; and finally, t^* indicating in which round the single call to CHALL is made. An explicit description of the OW-k-PCS$_{\neg\mathrm{RC}}$ security game in the symbolic model can be found in Fig. 7.

Definition 4 (Symbolic k-PCS security). *Let* CGKA *be a continuous group-key agreement scheme,* $k \in \mathbb{N}$*. Then* CGKA *is* OW-k-PCS$_{\neg\mathrm{RC}}$ secure*, if for all* $(n, u_0, C_0, (U_t, C_t)_{t=1}^{t_{\max}}, t^*)$ *it holds that*

$$\Pr[\text{OW-}k\text{-PCS}_{\neg\mathrm{RC}}^{\mathsf{CGKA}}(n, u_0, C_0, (U_t, C_t)_{t=1}^{t_{\max}}, t^*) \Rightarrow 1] = 0$$

where the probability is taken over the non-symbolic randomness.

This notion of security, in which for any sequence $(n, u_0, C_0, (U_t, C_t)_{t=1}^{t_{\max}}, t^*)$ the game is lost, is standard in the literature of symbolic security and used, for instance, in [24] and [12]. The requirement that the probability be zero, implies that the game is not won for every possible choice of non-symbolic randomness. The reason for this choice rather than requiring that it be a negligible function in $\log|R|$, where R denotes the set of non-symbolic randomness, is that it may very well be the case that $|R|$ is small since this is not the randomness used to sample new keys (when it would be reasonable to work with $\log|R|$ as a security parameter). For instance, one could just flip a coin (i.e., $R = \{0, 1\}$).

In the game we require that all symbolic random coins used by users are generated disjointly. More precisely, if $\mathtt{r}_u^t$ denotes the set of random coins used by user u in round t in the init/update procedures, then we require that $\mathtt{r} \in \mathtt{r}_u^t$ implies $\mathtt{r} \notin \mathtt{r}_{u'}^{t'}$ for all $(u, t) \neq (u', t')$.

We now define a property of CGKA schemes that we will require for our bounds. It essentially forbids schemes to generate layered ciphertexts of the form $\mathsf{PKE.Enc}(\mathtt{pk}_2, \mathsf{PKE.Enc}(\mathtt{pk}_2, \mathtt{m}))$. For some intuition on how it factors into our translation to the combinatorial model see Remark 3.

Definition 5 (No nested encryption). *We say a scheme* CGKA *does not use* nested encryption *if, for all ciphertexts* $\mathtt{c} \leftarrow \mathsf{PKE.Enc}(\mathtt{pk}, \mathtt{m})$*, the encrypted message is either a secret key or a random coin, i.e., of type* $\mathtt{sk}, \mathtt{K}$*, or* $\mathtt{r}$*.*

Our goal for the remainder of this section is to show that the task of deriving lower bounds on the communication complexity of correct and secure CGKA schemes translates to the analogue in the combinatorial model. To this end, we define *useful secrets*, i.e., secret symbolic variables that the adversary is not able to derive, and associate them to the set of users with knowledge of them. We prove that these sets satisfy the properties described in Sect. 3.1 for a cost function that counts the number of messages sent in a given round by a user.

Useful Secrets and Associated Sets. First, we establish some notation. Consider an adversary playing OW-k-PCS$_{\neg\text{RC}}$. We denote the set of public messages sent up to and including round t by $\mathtt{M}_t$, i.e., for $t = 0$ we set $\mathtt{M}_0 = \{\mathtt{pub}, \mathtt{MI}^0_{u_0}\}$ to be the output of oracle Init; and for every round $t \geq 1$ we extend the set by the output of oracle ROUND: $\mathtt{M}_t \leftarrow \mathtt{M}_{t-1} \cup \mathtt{MU}$, where $\mathtt{MU} = \mathsf{ROUND}(U_t)$. Further, for $t \geq 0$ we track all variables the adversary learned up to and including round t, via the corruption oracle, in a set $\mathtt{COR}_t$. I.e., at the beginning of round t, the set $\mathtt{COR}_t$ is initialized to $\mathtt{COR}_{t-1}$ and, if user u is corrupted in round t, then their current state $\mathtt{st}_u$ (meaning the one after all oracle calls of the round) is added to the set. Note that $\mathtt{COR}_t$ matches the set $\mathtt{COR}_t$, defined in game OW-k-PCS$_{\neg\text{RC}}$, that tracks the values known to the adversary via corruption. This allows us to define the notion of useful secrets $\mathtt{s}$, i.e., variables of type $\mathtt{r}$, $\mathtt{K}$, and $\mathtt{sk}$ that cannot be derived by the adversary, and associate to them the set of users that in round t have access to $\mathtt{s}$.

Definition 6 (Useful secrets and associated sets). *Consider adversary* A *playing game* OW-k-PCS$_{\neg\text{RC}}$ *in the symbolic model and let* $t \in \mathbb{N}$*, and* $\mathtt{s}$ *be a variable of type* $\mathtt{r}$*,* $\mathtt{K}$*, or* $\mathtt{sk}$ *generated during the game, before or in round* t*. We say that* $\mathtt{s}$ *is* useful *in round* t *if* $\mathtt{s} \notin \mathsf{Der}(\{\mathtt{M}_t, \mathtt{COR}_t\})$*. Let* $\mathtt{s}$ *be useful in round* t*. We define the associated set of* $\mathtt{s}$ *in round* t *as*

$$S(\mathtt{s}, t) := \{u \in [n] \mid \mathtt{s} \in \mathsf{Der}(\mathtt{st}_u^{t_{c,u}}, (\mathtt{r}_u^{t'})_{t_{c,u}+1 \leq t' \leq t}, \mathtt{M}_t)\} \subseteq [n]$$

where $t_{c,u} := \max\{\tilde{t} \mid u \in C_{\tilde{t}} \text{ and } \tilde{t} \leq t\}$ *(*$t_{c,u} = -1$ *if* u *has never been corrupted). We define the associated set of* $\mathtt{s}$ *after the setup as*

$$S(\mathtt{s}, -1) := \{u \in [n] \mid \mathtt{s} \in \mathsf{Der}(\mathtt{st}_u^{-1}, \mathtt{pub})\} \subseteq [n] \ .$$

We define the set system *in round* t *as*

$$\mathcal{S}_t := \{S(\mathtt{s}, t) \mid \mathtt{s} \text{ is useful in round } t\} \subseteq 2^{[n]} \ .$$

The intuition behind the definition of $S(\mathtt{s}, t)$ is that any user who can derive the secret $\mathtt{s}$ in a round t' such that $t_{c,u}+1 \leq t' \leq t$ (i.e., $\mathtt{s} \in \mathsf{Der}(\mathtt{st}_u^{t'}, \mathtt{M}_{t'})$) should belong to the set $S(\mathtt{s}, t)$. This is indeed the case since $\mathtt{st}_u^{t'} \subseteq \mathsf{Der}(\mathtt{st}_u^{t'-1}, \mathtt{r}_u^{t'}, \mathtt{M}_{t'})$ because symbolic outputs of algorithms can always be derived symbolically from their symbolic inputs. The notation $\mathtt{st}_u^{-1}$ refers to the state that u is assigned by the Setup algorithm.

We now define what it means for a scheme to not allow users to distribute work. Intuitively, this requirement says that whenever a secret (be it already existing or newly generated) is communicated to a set of users, who did not yet have access to it, then this communication must have been done by a *single* user. For example, this notion excludes schemes in which two users u_1, u_2, already sharing a common key, communicate this key to users u_3 and u_4 by having u_1 encrypt it to u_3, and u_2 encrypt it to u_4.

See Fig. 6 for an illustration of a scheme that *does* make use of distributed work at hand of a ratchet tree.

Definition 7 (No distributed work). *Consider a scheme* CGKA *and an execution of game* OW-k-PCS$_{\neg\mathrm{RC}}$ *with respect to* CGKA *in the symbolic model. For user u and round t let* $\mathtt{st}_u^t$ *denote the user's state in round t and* $\mathtt{r}_u^t$ *the random coins generated in round t. We say that* CGKA *does* not allow users to distribute work *if, for all t and every secret symbolic variable* $\mathtt{s}$*, we have that there exists a user u' such that for every* $u \in S(\mathtt{s}, t) \setminus (S(\mathtt{s}, t-1) \cup \{u'\})$ *it holds that* $\mathtt{s} \in \mathsf{Der}(\mathtt{st}_u^{t-1}, \mathtt{M}_{t-1}, \mathtt{MU}_{u'}^t)$ *and* $\mathtt{s} \in \mathsf{Der}(\mathtt{st}_{u'}^{t-1}, \mathtt{r}_{u'}^t, \mathtt{M}_{t-1})$.

Connection to Combinatorial Model. In the following we show that the three properties required in the combinatorial model are satisfied by the symbolic model's associated set system. The first two are quite natural observations, the last essentially corresponds to a generalization of a statement that is shown in the proof of [12, Theorem 2] and can be seen as quantifying the cost of adding new sets to the set system $\mathcal{S}_t$ by updating. We measure the cost in terms of the number of symbolic variables sent by a user u in round t and denote this quantity $|\mathtt{MU}_u^t|$. When interested in the cost of a round t, we take the sum over all users $u \in U_t$.

Intuitively, Property (i) is enforced by correctness and security, as on one hand every member of G_t must be able to derive the current group key from their state, and the safety predicate being satisfied implies that the group key at time t^* must be useful, i.e., $G_{t^*} \in \mathcal{S}_{t^*}$. Property (ii) corresponds to the simple fact that no secret derivable from $\mathtt{st}_u$ can be useful in a round in which the user gets corrupted, as in this case it can be derived by the adversary as well.

Equivalently, a set $S \in \mathcal{S}_t$ cannot contain any users in C_t. Finally, Property (iii) corresponds to the intuition, that the secret $\mathtt{s}$ belonging to a new set $S = S(\mathtt{s}, t)$ needs to be communicated to (at least) every member u of S. If $\mathtt{s}$ cannot be derived using PRF evaluations from a secret already known to u, then either it, or a secret which can be derived from using PRF, must be communicated to u by encrypting it to a useful key that was known to the party in the previous round, i.e., in round $t-1$. In other words, this determines a covering of the set S with sets in $\mathcal{S}_{t-1}$ and possibly a singleton $\{u\}$ for some updating user $u \in U_t$ with the property that the number of symbolic variables contained in the messages exchanged in round t is at least the number of sets in the said cover minus one. When we consider schemes in which users do not distribute work, we obtain a simpler statement for property (iii) and it matches Eq. 5 from the combinatorial model.

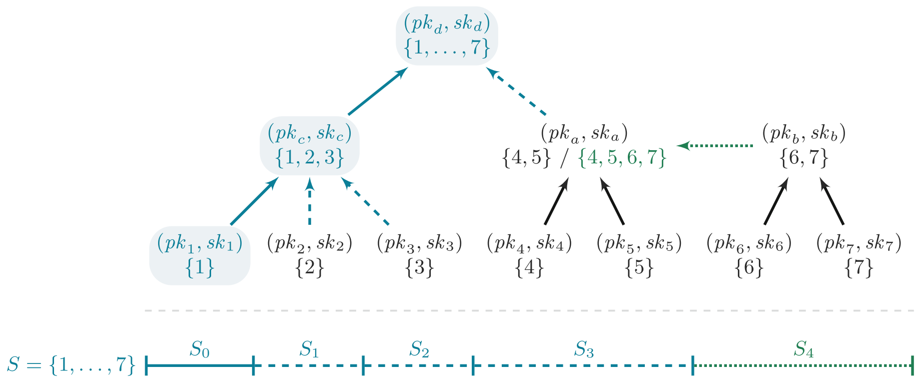

Fig. 6. Top: Example of a ratchet tree and its associated set system $\mathcal{S}_t$ making use of distributed work. Vertices contain key-pairs (above) and the associated set (below). Edges indicate that knowledge of the secret key of the source implies knowledge of the one of the sink. Dashed edges correspond to ciphertexts $\mathsf{Enc}_{pk_{\text{source}}}(sk_{\text{sink}})$ sent by user 1 in round t, solid edges either to keys derived using a PRF in round t or to keys communicated in a previous round. Keys already present in the system at time $t-1$ are depicted in black and keys added by user 1 in round t in blue with shaded background. The dotted edge corresponds to a ciphertext sent by user 5 in round t. Note that the associated set of (pk_a, sk_a) changes in round t as an effect of this ciphertext, and that users 6 and 7 need to decrypt ciphertexts sent by two different users, namely users 1 and 5, in order to recover sk_d, implying that the scheme does indeed use distributed work. Bottom: Depiction of the sets proven to exist in Lemma 1 using the example $S = \{1,\dots,7\} = \bigcup_{i=0}^{k} S_i$ in the set system depicted above. Note that $k = 4$ matches the number of ciphertexts sent in round t to establish S, which, however, stem from more than a single user (compare Lemma 1; (iii)). (Color figure online)

Lemma 1. *Let* CGKA *be a perfectly correct continuous group-key agreement scheme that is* OW-k-PCS$_{\neg\text{RC}}$*-secure and does not use nested encryption. Consider an adversary playing game* OW-k-PCS$_{\neg\text{RC}}$ *of Fig. 7 in the symbolic model.*

(i) If $\mathtt{K}_{G_t}$ *is the group key in round* t*, then* $S(\mathtt{K}_{G_t}, t) = G_t$*. In particular, if oracle* CHALL *is queried in round* t^* *and the safety predicate is satisfied then we have* $G_{t^*} \in \mathcal{S}_{t^*}$*.*

(ii) If user u *was corrupted by the adversary in round* t*, then for every* $S \in \mathcal{S}_t$ *it holds that* $u \notin S$*.*

(iii) Let $t \geq 1$*. Recall, that* $U_t \subseteq [n]$ *indicate the users that updated and for user* u *the sets* $\mathtt{MU}_u^t$ *correspond to the control messages generated by performing the corresponding update operation. Then for every set* $S \in \mathcal{S}_t$ *there exist* $k \in \mathbb{N}_{\geq 0}$ *and sets* $\{S_i\}_{i=0}^{k}$ *such that either*

$$\text{(a) } S_0 = \{u\} \text{ for some } u \in U_t, \quad \text{or (b) } S_0 \in \mathcal{S}_{t-1}$$

and, if $k \geq 1$, $S_i \in \mathcal{S}_{t-1}$ for every $i = 1, \ldots, k$. Furthermore, it holds that

$$S \cap C_{\leq t} \subseteq \bigcup_{i=0}^{k} S_i \quad \textit{and} \quad \sum_{u \in U_t} |\mathtt{MU}_u^t| \geq k \ .$$

where $C_{\leq t} = \bigcup_{0 \leq t' \leq t} C_{t'}$ are the users that have been corrupted at least once in round t or before. If CGKA *does not allow users to distribute work, then the last statement can be replaced by the following stronger expression:*

$$\exists u \in S_0 \cap U_t \ \textit{such that} \ |\mathtt{MU}_u^t| \geq k \ .$$

The lemma's proof is in the full version of this paper [8].

Remark 2. Looking ahead, we observe the following. Consider an execution of game OW-k-PCS$_{\neg\text{RC}}$ in the symbolic model, where CGKA is a perfectly correct, OW-k-PCS$_{\neg\text{RC}}$-secure CGKA scheme which does not use nested encryption and does not allow users to distribute work, and set $\text{Cost}(u, t) = |\mathtt{MU}_u^t|$ to be the number of symbolic variables sent by u in round t. Then, by Lemma 1 the associated set system $\mathcal{S}_t$ (Definition 6) and Cost satisfy all properties of the combinatorial model described in Sect. 3.1. As a consequence, to prove lower bounds on the communication cost of CGKA, i.e., the number of ciphertexts sent during the execution of the game, it is sufficient to lower bound the cost function for a scheme satisfying the combinatorial model.

Remark 3. Lemma 1 requires that CGKA not use nested encryption, i.e., not generate encryptions of ciphertexts. On a technical level, this restriction guarantees that for the graph constructed in the lemma's proof for every edge $(\mathtt{s}_1, \mathtt{s}_2)$ we have that knowledge of secret $\mathtt{s}_1$ implies knowledge of $\mathtt{s}_2$. On a more intuitive level, allowing ciphertexts of the form $\mathtt{c} = \mathsf{Enc}(\mathtt{pk}_2, \mathsf{Enc}(\mathtt{pk}_1, \mathtt{m}))$ would enable users to send ciphertext c in one round but release message $\mathtt{m}$ in a later round by at this point in time sending $\mathtt{sk}_2$ in the plain, at cost of no additional ciphertexts. While this does not seem to help with the total communication cost, it could in principle enable users to distribute their workload over several rounds. An analogous statement holds, if one allows the use of dual PRFs (as in [12]).

Following the idea outlined in Remark 2 it can be shown that the worst-case lower bound on the communication cost of CGKA schemes in the combinatorial model carries over to the symbolic model for OW-k-PCS$_{\neg\text{RC}}$-secure schemes.

Corollary 5. *Let k, n, t_c, ε, and α_ε be as in Corollary 5 and let* CGKA *be a correct and* OW-k-PCS$_{\neg\text{RC}}$*-secure CGKA scheme that does not use nested encryption, and does not allow users to distribute work. Let $z_{\max} \in \mathbb{N}$ and for every integer $0 \leq z < z_{\max}$ set $t_{c,z} = t_c + z \cdot (t_c + (1 + \varepsilon)k)$ and $t_{\max} = z_{\max} \cdot (t_c + (1 + \varepsilon)k)$.*

If $3 \leq k \leq \ln(\alpha_\varepsilon n)$, then for an arbitrary setup phase of the group $G_0 = G_t = [n]$ and $z_{\max}$ arbitrary phases of t_c rounds of updates $(U_t)_{t=t_{c,z}-t_c+1}^{t_{c,z}}$ with $0 \leq z < z_{\max}$ and any choice of non-symbolic randomness in the security game

```
**Game** $\text{OW-}k\text{-PCS}^{\mathsf{CGKA}}_{\neg\text{RC}}(n, u_0, C_0, (U_t, C_t)_{t=1}^{t_{\max}}, t^*)$
$\mathsf{INIT}(n, u_0)$
**for** $u \in C_0$: **call** $\mathsf{CORR}(u)$
**for** $t = 1, \ldots, t_{\max}$:
    $\mathsf{ROUND}(U_t)$
    $\mathtt{COR}_t \leftarrow \mathtt{COR}_{t-1}$
    **for** $u \in C_t$: **call** $\mathsf{CORR}(u)$
    **if** $t = t^*$: **call** $\mathsf{CHALL}$
**return** $[\mathtt{K}^* \in \mathsf{Der}(\mathtt{COR}_{t_{\max}}, \mathtt{M}) \wedge \text{safe}_{k\text{-PCS}}]$

**Oracle** $\mathsf{INIT}(n, u_0)$                    \\one call; called first
$t \leftarrow 0$
$\mathtt{M} \leftarrow \emptyset$, $\mathtt{COR}_0 \leftarrow \emptyset$
$\text{Cor}[u] \leftarrow \emptyset$, $\text{Upd}[u] \leftarrow \emptyset$ **for all** $u \in [n]$
$\text{Coins}[u, t] \leftarrow \emptyset$ **for all** $u \in [n]$, $\forall t$
$\mathtt{r_{setup}} \leftarrow_\$ Rnd$
$(\mathtt{pub}, (\mathtt{st}_u)_{u \in [n]}) \leftarrow \mathsf{Setup}(n; \mathtt{r_{setup}})$
$\mathtt{r} \leftarrow_\$ Rnd$; $\text{Coins}[u_0, t] \leftarrow_\cup \{\mathtt{r}\}$
$(\mathtt{st}_{u_0}, \mathtt{MI}^0_{u_0}) \leftarrow \mathsf{Init}(\mathtt{st}_{u_0}, \mathtt{pub}, n; \mathtt{r})$
**for** $u \in [n]$:
    $\mathtt{st}_u \leftarrow \mathsf{Process}(\mathtt{st}_u, \mathtt{pub}, \mathtt{MI}^0_{u_0})$
    $\mathtt{K}_G \leftarrow \mathsf{GetKey}(\mathtt{st}_u)$
$\mathtt{M} \leftarrow_\cup \{\mathtt{pub}, \mathtt{MI}^0_{u_0}\}$

**Oracle** $\mathsf{CHALL}$                    \\one call
$\mathtt{K}^* \leftarrow \mathtt{K}_G$

**Oracle** $\mathsf{ROUND}(U_t)$
$\mathtt{MU} \leftarrow \emptyset$
**for** $u \in U_t$:
    $\text{Upd}[u] \leftarrow_\cup \{t\}$
    $\mathtt{r} \leftarrow_\$ Rnd$; $\text{Coins}[u, t] \leftarrow_\cup \{\mathtt{r}\}$
    $(\mathtt{st}_u, \mathtt{MU}^t_u) \leftarrow \mathsf{Update}(\mathtt{st}_u, \mathtt{pub}; \mathtt{r})$
    $\mathtt{MU} \leftarrow_\cup \{\mathtt{MU}^t_u\}$
**for** $u \in [n]$:
    $\mathtt{st}_u \leftarrow \mathsf{Process}(\mathtt{st}_u, \mathtt{pub}, \mathtt{MU})$
    $\mathtt{K}_G \leftarrow \mathsf{GetKey}(\mathtt{st}_u)$
$\mathtt{M} \leftarrow_\cup \mathtt{MU}$

**Predicate** $\text{safe}_{k\text{-PCS}}$
**for** $u \in [n]$:
    **if** $\text{Cor}[u] \neq \emptyset$
        **if** $\exists t \in \text{Cor}[u] : t \geq t^*$:          \\corruption after challenge
            **return** 0
        $t^c_u \leftarrow \max\{t \in \text{Cor}[u]\}$
        **if** $|\{t \in \text{Upd}[u] : t^c_u < t \leq t^*\}| < k$:          \\too few updates
            **return** 0
**return** 1

**Oracle** $\mathsf{CORR}(u)$
$\text{Cor}[u] \leftarrow_\cup \{t\}$
$\mathtt{COR}_t \leftarrow_\cup \{\mathtt{st}_u\}$
```

Fig. 7. Security game OW-k-PCS$_{\neg\text{RC}}$ in the symbolic model with respect to the sequence $(n, u_0, C_0, (U_t, C_t)_{t=1}^{t_{\max}}, t^*)$ of inputs to the oracles.

OW-k-PCS$_{\neg\text{RC}}$, *there exist sequences of updates* $(U_t)_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k}$ *and sets of corrupted users* $C_z \subseteq [n]$ *each of cardinality* $\lceil \alpha_\varepsilon n \rceil$ *such that the total communication cost satisfies*

$$\sum_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k} \text{Cost}(U_{t_{c,z}+t}) \geq \frac{(k-1)}{4} \cdot \left\lfloor \frac{2\varepsilon n}{5(1+\varepsilon)\lceil \log(k) \rceil} \right\rfloor \left((\alpha_\varepsilon n)^{\frac{1}{(1+\varepsilon)k-1}} - 1 \right)$$

for every $0 \leq z < z_{\max}$.

If CGKA *has publicly-computable update cost (Definition 3), the sequences of sets* $(U_t)_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k}$ *can be computed online, i.e.,* $U_{t_{c,z}+t}$ *can be computed using public information from the previous rounds. Furthermore, if* CGKA *has offline publicly-computable update cost, the sequence of updates* $(U_t)_{t=t_{c,z}+1}^{t_{c,z}+(1+\varepsilon)k}$ *can be computed after round* $t_{c,z}$ *and is independent of the non-symbolic randomness.*

The corollary's proof is in the full version of this paper [8].

5 Upper Bound on the Update Cost

In this section we briefly outline a simple CGKA protocol, inspired by CoCoA [2], but both more general and, for certain values of k, with a lower total upload communication cost. Accordingly, we termed it CoCoALight.

This short section assumes knowledge of CoCoA and only aims to give an intuitive understanding of the ideas behind the proposed protocol. For space reasons we defer a more thorough discussion on CoCoA and our protocol, as well as the formal description and security proof to the full version of this work [8].

The key feature of the protocol is that, as opposed to in CoCoA, users will not rotate the keys for all nodes in their path in each update, but instead just rotate the key of a single node. Users keep track, by means of a counter, of which node they last refreshed, and will, in the following update, sample a new key for its child, increasing the counter by 1. In the case that two users send ciphertexts corresponding to the same node in the same round, the server will decide a winner, as in CoCoA, and thus whose key will be the next one associated to said node, according to any agreed-upon (potentially deterministic) rule. In the case of such a collision, the user losing will still "make progress" and increase their counter, and so, in the following update will attempt to rotate the key at the next node in their path.

A consequence of rotating a single key per update is that knowledge of parts of the old state might allow the recovery of this new key. In particular, the knowledge of the secret key of the parent key of v, when v's key is being refreshed, allows the recovery of the latter (as its seed will be encrypted under the former). Thus, informally, what ensures healing is the progressive rotation of all the path's keys after corruption, and *starting* from the leaf. Thus, in order to guarantee healing in k rounds, CoCoALight uses trees of depth $\approx k/2$, to ensure a rotation of all keys in the path *starting at the leaf* happens within that period.

In particular, it can recover from an arbitrary number of corruptions in k rounds and with a total communication cost in the order of $nk \sqrt[k/2]{n}$ ciphertexts, without any user coordination. While CoCoA's communication complexity is lower for low values of k, CoCoALight's improves for values of k closer to $\log(n)$. This improvement comes at the drawback of non-immediate forward secrecy, which requires at least $k/2$ updates from each user prior to their corruption. Likewise, we not prove it secure against any type of active adversary and, indeed, only describe a simple protocol satisfying IND-k-PCS$_{\mathrm{RC}}$ security. Nevertheless, it shows that the lower bound on PCS from the previous section is only $\log(k)/\sqrt[k/2]{n}$ from being tight, for $k \in [4, 2\lceil\log(n)\rceil + 1]$. Concretely, for the case $k = \log(n)$, the gap is of order just $\log(\log(n))$.

References

1. Alwen, J., et al.: Grafting key trees: efficient key management for overlapping groups. In: Nissim, K., Waters, B. (eds.) TCC 2021, Part III. LNCS, vol. 13044, pp. 222–253. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-90456-2_8
2. Alwen, J., et al.: CoCoA: concurrent continuous group key agreement. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 815–844. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-07085-3_28
3. Alwen, J., Coretti, S., Dodis, Y., Tselekounis, Y.: Security analysis and improvements for the IETF MLS standard for group messaging. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part I. LNCS, vol. 12170, pp. 248–277. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-56784-2_9

4. Alwen, J., Coretti, S., Jost, D., Mularczyk, M.: Continuous group key agreement with active security. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 261–290. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-64378-2_10
5. Alwen, J., Hartmann, D., Kiltz, E., Mularczyk, M.: Server-aided continuous group key agreement. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 69–82. ACM Press, November 2022
6. Alwen, J., Jost, D., Mularczyk, M.: On the insider security of MLS. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 34–68. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-15979-4_2
7. Alwen, J., Auerbach, B., Noval, M.C., Klein, K., Pascual-Perez, G., Pietrzak, K.: DeCAF: decentralizable continuous group key agreement with fast healing. Cryptology ePrint Archive, Paper 2022/559 (2022). https://eprint.iacr.org/2022/559
8. Auerbach, B., Noval, M.C., Pascual-Perez, G., Pietrzak, K.: On the cost of post-compromise security in concurrent continuous group-key agreement. Cryptology ePrint Archive, Paper 2023/1123 (2023). https://eprint.iacr.org/2023/1123
9. Barnes, R., Beurdouche, B., Robert, R., Millican, J., Omara, E., Cohn-Gordon, K.: The Messaging Layer Security (MLS) Protocol. RFC 9420, July 2023. https://www.rfc-editor.org/info/rfc9420
10. Bhargavan, K., Barnes, R., Rescorla, E.: TreeKEM: Asynchronous Decentralized Key Management for Large Dynamic Groups, May 2018. https://mailarchive.ietf.org/arch/attach/mls/pdf1XUH6o.pdf
11. Bienstock, A., Dodis, Y., Garg, S., Grogan, G., Hajiabadi, M., Rösler, P.: On the worst-case inefficiency of CGKA. In: Kiltz, E., Vaikuntanathan, V. (eds.) TCC 2022, Part II. LNCS, vol. 13748, pp. 213–243. Springer, Heidelberg (2022). https://doi.org/10.1007/978-3-031-22365-5_8
12. Bienstock, A., Dodis, Y., Rösler, P.: On the price of concurrency in group ratcheting protocols. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part II. LNCS, vol. 12551, pp. 198–228. Springer, Heidelberg (2020). https://doi.org/10.1007/978-3-030-64378-2_8
13. Brzuska, C., Cornelissen, E., Kohbrok, K.: Cryptographic security of the MLS RFC, draft 11. Cryptology ePrint Archive, Report 2021/137 (2021). https://eprint.iacr.org/2021/137
14. Canetti, R., Garay, J.A., Itkis, G., Micciancio, D., Naor, M., Pinkas, B.: Multicast security: a taxonomy and some efficient constructions. In: IEEE INFOCOM 1999, pp. 708–716, New York, NY, USA, 21–25 March 1999
15. Cohn-Gordon, K., Cremers, C., Garratt, L., Millican, J., Milner, K.: On ends-to-ends encryption: asynchronous group messaging with strong security guarantees. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018, pp. 1802–1819. ACM Press, October 2018
16. Cremers, C., Hale, B., Kohbrok, K.: The complexities of healing in secure group messaging: Why cross-group effects matter. In: Bailey, M., Greenstadt, R. (eds.) USENIX Security 2021, pp. 1847–1864. USENIX Association, August 2021
17. Devigne, J., Duguey, C., Fouque, P.A.: MLS group messaging: how zero-knowledge can secure updates. In: Bertino, E., Shulman, H., Waidner, M. (eds.) ESORICS 2021, Part II. LNCS, vol. 12973, pp. 587–607. Springer, Heidelberg (2021). https://doi.org/10.1007/978-3-030-88428-4_29
18. Dolev, D., Yao, A.: On the security of public key protocols. IEEE Trans. Inf. Theory **29**(2), 198–208 (1983)

19. Hashimoto, K., Katsumata, S., Postlethwaite, E., Prest, T., Westerbaan, B.: A concrete treatment of efficient continuous group key agreement via multi-recipient PKEs. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021, pp. 1441–1462. ACM Press, November 2021
20. Hashimoto, K., Katsumata, S., Prest, T.: How to hide MetaData in MLS-like secure group messaging: simple, modular, and post-quantum. In: Yin, H., Stavrou, A., Cremers, C., Shi, E. (eds.) ACM CCS 2022, pp. 1399–1412. ACM Press, November 2022
21. Jost, D., Maurer, U., Mularczyk, M.: A unified and composable take on ratcheting. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 180–210. Springer, Heidelberg (2019). https://doi.org/10.1007/978-3-030-36033-7_7
22. Klein, K., et al.: Keep the dirt: tainted TreeKEM, adaptively and actively secure continuous group key agreement. In: 2021 IEEE Symposium on Security and Privacy, pp. 268–284. IEEE Computer Society Press, May 2021
23. Weidner, M.A.: Group messaging for secure asynchronous collaboration. Master's thesis, University of Cambridge, June 2019
24. Micciancio, D., Panjwani, S.: Optimal communication complexity of generic multicast key distribution. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 153–170. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24676-3_10
25. Wallner, D., Harder, E., Agee, R.: Key management for multicast: issues and architectures. Request for Comments: 2627, Internet Engineering Task Force (1999)
26. Weidner, M., Kleppmann, M., Hugenroth, D., Beresford, A.R.: Key agreement for decentralized secure group messaging with strong security guarantees. In: Vigna, G., Shi, E. (eds.) ACM CCS 2021, pp. 2024–2045. ACM Press, November 2021