AICR · OPEN PROBLEM

Computation-Sublinear Refresh for Forward-Secure CGKA

Sublinear-time key refresh for updatable distributed broadcast encryption

Status. Statement: AI-written from James Bartusek, Nir Bitansky, Yevgeniy Dodis, Rachit Garg and David J. Wu, *Fair-Weather No More: Guaranteed Efficiency in Secure Group Messaging*, IACR ePrint 2026/1677. Not yet checked by a human. Open: the source achieves sublinear *communication* with forward secrecy, but leaves sublinear refresh *time* unresolved, and shows only that a natural way of pushing further — dropping the membership set as an input entirely — is insecure for one specific construction.

Category. *Secure Group Messaging & Key Agreement.*

Conjecture (informal). *Continuous group key agreement (CGKA) protocols let a dynamic group of users maintain a shared secret, and forward secrecy requires an explicit key-rotation ("refresh") step, so that later compromising a user's state cannot expose past group keys. The source builds the first CGKA scheme with forward secrecy whose communication cost is worst-case sublinear, in fact essentially independent of the group size, using a new primitive it calls updatable distributed broadcast encryption. But the scheme's refresh algorithms still take time polynomial in the number of users, because they take the group's whole membership as an explicit input. The source poses the question of whether this can be avoided: is there a scheme with the same security guarantees whose refresh operation also runs in time sublinear in the group size, and not merely produces a short message? The source commits to no specific target rate, such as polylogarithmic, for this question. A strictly stronger version, raised separately, asks for refresh algorithms that do not depend on the membership set at all; the source shows a natural way of achieving that stronger property is completely insecure, by an attack that cancels out the encryption randomness between two users' ciphertext components — but this attack targets only the stronger variant, not the plain sublinear-time question stated here.*

1 The setting

An *updatable distributed broadcast encryption* (updatable DBE) scheme is the cryptographic core of the source's forward-secure CGKA construction. Users are identified by indices; each user i holds a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$, and a set S of currently active users has an aggregated public key apk together with, for each $i \in S$, an aggregated secret key ask_i. A sender encrypts a single message to apk, and every $i \in S$ decrypts it with ask_i. Two update mechanisms are required: *incremental* updates that add or remove one user from S, and a *refresh* mechanism, run via algorithms UpdPK and UpdSK, that re-randomizes every active user's keys at once. The refresh mechanism is what supplies forward secrecy: after a refresh, compromising a user's new secret key does not help decrypt ciphertexts sent before the refresh.

The source's succinctness requirement (its Definition 5.6) asks only that the *sizes* of apk, of each ask_i, of the refreshed public/secret keys, and of the refresh ciphertext ctref produced by UpdPK be $\mathrm{poly}(\lambda)$, independent of $|S|$. It explicitly does *not* require the *running time* of UpdPK and UpdSK to be sublinear in $|S|$: both algorithms take the entire membership mapping S as an explicit input and are allowed to touch every entry of it. The source's own construction (its Construction 5.10) achieves compact keys and compact communication this way, built from a lattice-based matrix-commitment scheme under the decomposed learning-with-errors assumption [AMR25] in the random oracle model, but its refresh algorithms run in time $\mathrm{poly}(\lambda, |S|)$. When used to build a CGKA protocol, the resulting protocol inherits this asymmetry: its Add, Remove and post-compromise-refresh operations all run in time independent of the group size, but its forward-secrecy refresh operation runs in time polynomial in the group size, even though the message it broadcasts stays short. The only prior CGKA scheme achieving forward secrecy at all, RTreeKEM [ACDT20], does not meet even the source's weaker, communication-only efficiency requirement, so no scheme — prior or from this source — achieves forward secrecy together with a sublinear-time refresh.

The source's own Remark 5.7 records the gap and poses the question without committing to a specific target rate: "An important open problem is to obtain a scheme where the refresh operation

requires sublinear running time." A separate, strictly stronger question is raised only in a footnote: dropping S as an explicit input to UpdPK and UpdSK entirely, so that they run in time $\mathrm{poly}(\lambda)$ with no dependence on $|S|$ whatsoever. The source's Appendix A gives an attack on a natural construction achieving that stronger property — replacing each user's independent random shift with one shared shift — but this attack does not bear on the plain sublinear-refresh question below, which still permits UpdPK and UpdSK to take S as input.

2 Notation and parameters

- λ is the security parameter.
- S is a finite set of currently active group members, represented as a map from user indices to public keys; $|S|$ is the group size.
- apk is the aggregated public key associated with S; ask_i is user i's aggregated secret key.
- ctref is the refresh ciphertext produced by UpdPK and consumed by UpdSK.

3 The conjecture

Definition 1 (Updatable distributed broadcast encryption, source's Section 5). An updatable distributed broadcast encryption scheme for up to 2^λ users is a tuple $\Pi = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{IncPK}, \mathsf{IncSK}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{UpdPK}, \mathsf{UpdSK})$. $\mathsf{Setup}(1^\lambda)$ outputs public parameters pp, and $\mathsf{KeyGen}(\mathsf{pp}, i)$ outputs a key pair $(\mathsf{pk}_i, \mathsf{sk}_i)$ for user i. Writing S for a finite set of active users, IncPK and IncSK update apk and ask_i whenever a user joins or leaves S; $\mathsf{Enc}(\mathsf{pp}, \mathsf{apk}, \mu)$ and $\mathsf{Dec}(\mathsf{pp}, \mathsf{ask}_i, \mathsf{ct})$ encrypt to, and decrypt for, the current group; and the refresh pair $\mathsf{UpdPK}(\mathsf{pp}, \mathsf{apk}, S) \to (\mathsf{ctref}, \mathsf{apk}', S')$, $\mathsf{UpdSK}(\mathsf{pp}, \mathsf{ask}_i, \mathsf{ctref}, S, \mathsf{pk}_i, \mathsf{sk}_i) \to (\mathsf{ask}'_i, \mathsf{pk}'_i, \mathsf{sk}'_i, S')$ jointly rerandomize every active user's keys given only the current membership S, so that after a refresh a corrupted user's new secret key does not help decrypt ciphertexts encrypted to apk before that refresh. Π is *secure* (statically or adaptively) if no efficient

adversary can distinguish an encryption of a chosen bit from an encryption of a uniform bit, given the view produced by any admissible sequence of key-generation, corruption, add/remove and refresh queries, provided no user in the group at the time of the challenge was ever corrupted while still in the group. Π is *succinct* if $|\mathsf{apk}|$, each $|\mathsf{ask}_i|$, each refreshed key, and $|\mathsf{ctref}|$ are all bounded by a fixed polynomial in λ, independent of $|S|$.

Conjecture 1 (Sublinear-time refresh). *Fix the syntax and security notion of Definition 1 together with its succinctness requirement. Does there exist a secure, succinct updatable distributed broadcast encryption scheme* Π *for up to* 2^λ *users such that, for every set* S *of active users with* $|S| \le 2^\lambda$*, both refresh algorithms* $\mathsf{UpdPK}(\mathsf{pp}, \mathsf{apk}, S)$ *and* $\mathsf{UpdSK}(\mathsf{pp}, \mathsf{ask}_i, \mathsf{ctref}, S, \mathsf{pk}_i, \mathsf{sk}_i)$ *run in time sublinear in* $|S|$ *— rather than merely producing output of size* $\mathrm{poly}(\lambda)$ *while running in time* $\mathrm{poly}(\lambda, |S|)$*, as in the source's own Construction 5.10? (The source does not commit to a specific target rate, such as polylogarithmic, for the refresh running time itself.)*
Equivalently, at the level of secure group messaging: does there exist a continuous group key agreement protocol that simultaneously achieves post-compromise security and forward secrecy while every *one of its operations — including the forward-secrecy refresh operation, and not merely its communication — runs in time sublinear in the current group size?*

Remark 1 (A strictly stronger, separately-posed variant). The source's Footnote 8 (printed page 7) raises a strictly stronger question: dropping S as an explicit input to UpdPK and UpdSK entirely, so that they run in time $\mathrm{poly}(\lambda)$ independent of $|S|$. The source's Appendix A gives an attack — a linear-cancellation ("zeroizing") argument comparing two users' local openings and ciphertext components — against a natural construction achieving *that* stronger property, obtained by replacing each user's independent per-user random shift with one shared shift. This attack is evidence of difficulty for the stronger, input-free variant; it does not attack, and is not offered by the source as evidence against, the plain sublinear-time question of Conjecture 1, which still permits UpdPK and UpdSK to take S as input.

4 Bibliography

[BBDGW26] James Bartusek, Nir Bitansky, Yevgeniy Dodis, Rachit Garg and David J. Wu. *Fair-weather no more: Guaranteed efficiency in secure group messaging.* IACR ePrint 2026/1677. The source. The open problem is posed on printed page 6 and restated as Remark 5.7, page 35; the succinctness requirement is Definition 5.6, page 35; the stronger input-free variant is Footnote 8, page 7; the attack on it is Appendix A, pages 72–74.

[ACDT20] Joël Alwen, Sandro Coretti, Yevgeniy Dodis and Yiannis Tselekounis. *Security analysis and improvements for the IETF MLS standard for group messaging.* CRYPTO 2020. RTreeKEM, the only prior CGKA scheme with forward secrecy, cited by the source as not meeting even its weaker, communication-only efficiency requirement.

[AMR25] Damiano Abram, Giulio Malavolta and Lawrence Roy. *Key-homomorphic computations for RAM: Fully succinct randomised encodings and more.* CRYPTO 2025. Introduces the decomposed learning-with-errors assumption the source's Construction 5.10 relies on.