Collaborative Rate Limiting Nullifier Signaling

Yağmur Gürel, Uğur Şen, and Oğuz Yayla

Institute of Applied Mathematics, Middle East Technical University,
Çankaya 06800, Ankara, Türkiye
yagmurg@metu.edu.tr, ugursen187@gmail.com, oguz@metu.edu.tr

Abstract. Rate Limiting Nullifier (RLN) is a privacy-preserving, decentralized spam-prevention mechanism for anonymous broadcast networks: each member may emit at most r signals per epoch, and any violation reveals a secret enabling the member's stake to be slashed. The standard construction binds each membership to a single secret key sk, coinciding identity, authorization, and slashing in one party, which rules out settings where a *group* should speak with one voice, share one rate budget, and stand behind one collective bond without any member acting unilaterally. We introduce *Collaborative RLN Signaling (coRLN)*, which lets n parties register as a single RLN member and signal only by acting jointly. The group secret sk_G is held as additive shares under SPDZ and never reconstructed; the identity (or rate) commitment, the per-epoch RLN evaluation, and the broadcast proof are produced inside an MPC network using collaborative zk-SNARKs. The group occupies one leaf in the membership Merkle tree, locks one aggregated stake stake_G, and is bound by one rate limit. We present the construction in both the rate-limit-1 and general $r \geq 1$ settings, with a collaborative withdrawal that lets the group exit without reconstructing sk_G, and prove three properties by reduction to the collaborative-SNARK composition: (i) no PPT adversary corrupting up to $n-1$ parties recovers sk_G while one party is honest; (ii) two same-epoch signals with the same $\mathsf{messageId}$ yield an extractor that recovers sk_G and forfeits stake_G; and (iii) no strict subset of G can produce a verifying signal. The verifier interface and signal shape match classical RLN at the byte level, so coRLN deploys on existing RLN-aware infrastructure with only the verification key updated.

Keywords: Rate Limiting Nullifier · collaborative SNARKs · anonymous group signaling · multi-party computation.

1 Introduction

Anonymous broadcast networks need a way to limit how often each participant speaks without identifying who is speaking. Rate Limiting Nullifier (RLN) [17,14] solves this by issuing each member a secret sk, committing $H(\mathsf{sk})$ in a Merkle tree, and forcing every signal to expose a polynomial share of sk with a zero-knowledge proof that it was computed correctly under some leaf. Within the per-epoch budget the polynomial is underdetermined and sk stays hidden; exceeding it lets

anyone interpolate, recover sk, and slash the member's stake, giving anonymous, Sybil-resistant, rate-limited speech with cryptographic abuse deterrence.

The construction is single-party by design: one sk, one commitment, one stake, one rate budget. This fits individual users but not applications where authority to speak is distributed across parties: a newsroom or NGO whose statements should bind the whole organization, a multi-operator bridge committee whose updates must reflect joint authorization, or a whistleblower coalition where no strict subset should publish alone.

What classical RLN cannot supply here is *joint authorization*: producing a signal should require every member, not any single keyholder. The naive workarounds fail. Sharing one sk creates a single point of failure and lets any keyholder exhaust the budget alone; issuing n parallel memberships fragments the stake, multiplies the effective rate limit by n, and breaks the single-speaker view. These applications need one externally-visible identity, one shared rate budget, and one collective stake behind joint authorization, which coRLN provides while retaining RLN's public interface and slashing semantics.

In this paper, we propose *Collaborative RLN Signaling (coRLN)*, a protocol that lifts RLN from a single-party to a group-authorized setting while keeping the external identity model unchanged. A group $G = \{U_1, \ldots, U_n\}$ registers as one RLN member: each U_i submits a local secret sk_i to an MPC network as an additive share of $\mathsf{sk}_G = \sum_i \mathsf{sk}_i$, and $H(\mathsf{sk}_G)$ is computed inside MPC and inserted as one leaf bound to a single aggregated stake stake_G. To broadcast, the group jointly evaluates the RLN line $y = \mathsf{sk}_G + a_1 x$ on shared inputs and produces a collaborative zk-SNARK [12], opening only y and $\mathsf{nullifier} = H(a_1)$; two same-epoch signals let the standard interpolation recover sk_G and slash the entire stake_G. We give the $r = 1$ construction and generalize it to an arbitrary budget $r_G \geq 1$, with a collaborative withdrawal that lets the group exit without reconstructing sk_G. Because sk_G exists only as additive shares, never reconstructed in honest execution, signaling requires every member, no strict subset can produce a valid signal, and slashing forfeits the pooled stake as a whole, making G a single unit for authority and accountability.

In our construction, coRLN composes SPDZ [3] for additive sharing and MPC evaluation of the RLN polynomial with collaborative zk-SNARKs [12] yielding a single proof from the distributed witness. Because this proof verifies under the standard verifier, the public signal $(m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi)$ matches classical RLN and only the verification key differs, so existing RLN-aware infrastructure is reused at the network and contract layer.

A group that holds one anonymous credential jointly and presents it so that the result verifies as an ordinary single-party presentation is an established notion, developed in the multi-holder credentials of Flamini, Lee and Lysyanskaya [6] and, on the issuance side, in Coconut [16] and threshold BBS+ [5]. Our contribution is not that idea but its instantiation for rate-limited signaling: an issuer-free membership admitted by stake, a jointly held identity carrying a self-punishing rate budget and one slashable bond, and verifier compatibility

obtained generically over an existing R1CS rather than from the algebra of a particular signature scheme.

The outline of our works is as follows. Section 2 surveys related work; Sections 3 and 4 give the construction for $r = 1$ and general $r \geq 1$; Section 5 proves the security properties; Section 6 surveys applications; and Section 7 analyzes the cost of joint signaling.

2 Related Work

We position coRLN at the intersection of four lines of work: anonymous group-signaling primitives, multi-party control of a single anonymous credential, rate-limited anonymous broadcast, and collaborative proving over distributed witnesses.

The classical primitive for "one member of a group signs anonymously" is the *group signature* [2], made practical by the short pairing-based construction of Boneh, Boyen, and Shacham [1], in which a manager may open signatures and revoke anonymity. coRLN is the opposite design point: *collective* accountability with no opening authority and no individual blame, suiting permissionless settings where no party holds unilateral authority over the group identity. In the Ethereum ecosystem, anonymous signaling is dominated by *Semaphore* [9], which proves group membership and emits a signal under an external nullifier without revealing the signer; the external nullifier prevents double-signaling within a scope and is the conceptual ancestor of RLN's epoch-based rate limit. coRLN inherits Semaphore's identity-commitment-in-Merkle-tree structure but lifts both membership and rate-limited evaluation to the group.

Distributing control of one anonymous credential across several parties is the subject of the threshold and multi-holder credential line, which is the closest prior work to coRLN. Coconut [16] and Doerner et al. [5] distribute the *issuer*: t-out-of-n authorities jointly issue a credential that its holder later presents alone. Flamini, Lee and Lysyanskaya [6] distribute the *holder*: shares of a single BBS credential sit with n parties who run a threshold presentation protocol whose output verifies as an ordinary BBS presentation, which is the same output-compatibility goal we pursue. On the axes the two constructions share, theirs is the stronger one, being t-out-of-n with identifiable abort where SPDZ affords coRLN n-out-of-n with abort. That line presupposes an issuer, whereas a coRLN membership is self-created and admitted by stake. None of it carries a rate budget whose violation publishes the group secret and forfeits a bond, which is the mechanism coRLN exists to lift to a group. And each obtains verifier compatibility from the algebra of one signature scheme, whereas coRLN obtains it generically, by proving a deployed R1CS over a witness split across parties, so the verification routine is reused as it stands and only its key changes.

Rate Limiting Nullifier extends Semaphore by attaching a Shamir share of the member's secret to each signal, so exceeding the per-epoch budget exposes the secret and enables slashing (§1). We follow the RLN-v1 specification [18] and the formal circom-rln constraint system [13,14]; RLN-v2 lifts the budget to a per-

leaf parameter r via a messageId counter and range check. Waku-RLN-Relay [17] integrates RLN into a GossipSub relay for proof-of-work-free spam protection, and later work [15] measures its latency and adds an rln_identifier separating co-located applications; coRLN preserves these network-layer choices. Closest to a "collective-stake RLN" is the deployed Waku model where a member pre-pays for budget proportional to its deposit; coRLN is the multi-party analogue, with n members sharing one membership whose budget tracks the pooled stake and whose secret is split across them.

The collaborative zk-SNARK framework of Ozdemir and Boneh [12] compiles standard provers (Groth16, PLONK, Marlin) to a witness split across parties, generating the proof by MPC while leaving the verifier and verification key unchanged. The MPC layer is typically an actively-secure linear secret-sharing protocol; we instantiate it with SPDZ, whose security-with-abort yields coRLN's n-out-of-n semantics (§3). The original framework is not optimized for large prover counts; Liu *et al.* [11] construct a scalable collaborative zk-SNARK for general circuits. We treat the concrete MPC backend and coSNARK construction as an instantiation question outside our scope.

3 Collaborative RLN Signaling

We present coRLN, a protocol that permits a group $G = \{U_1, U_2 \ldots, U_n\}$ to register as a single RLN member and emit anonymous signals only by acting jointly. The group is treated as one identity throughout: it occupies a single leaf in the membership Merkle tree, locks a single aggregated stake, and is bound by a single per-epoch signaling limit. Rate-limit violations are attributed to the group as a whole. We fix $r = 1$ in this section; the extension to $r \geq 2$ is presented in Section 4.

We assume an active PPT adversary corrupting up to $n-1$ of the n members. Privacy of sk_G and soundness of every signal hold as long as one party is honest, so the construction is dishonest-majority secure. The MPC is secure with abort: a corrupted party can halt progress, but no strict subset of G can forge a signal or recover sk_G, since one honest share is always required. "Group authorization" here means *n-out-of-n*, in the sense that every member must contribute a share to produce a signal; threshold (t-out-of-n) variants are left to future work (§8). We further assume collision and preimage resistance of H (e.g. Poseidon Hash Function [8], matching circom-rln), soundness and zero-knowledge of the coSNARK, and correct enforcement of protocol logic by the contract. The construction rests on two primitives, recalled formally in Appendix B. For the MPC we use SPDZ, additively shared over $\mathbb{F}$ and secure with abort against $n-1$ active corruptions, writing $[x]$ for a sharing of $x \in \mathbb{F}$ with U_i holding $[x]_i$ and $\sum_i [x]_i = x$; Poseidon is evaluated on shared inputs by standard arithmetic-circuit MPC: we write $H([x])$ for such an evaluation, which yields a sharing $[H(x)]$ of the digest, and reserve the plain $H(x)$ for an opened value or a relation constraint. The collaborative zk-SNARK then produces a single proof from this distributed witness, no party learning the full witness, and π verifies under the standard SNARK verifier.

Definition 1 (Collaborative RLN scheme). *A* collaborative RLN scheme *for a group* $G = \{U_1, \ldots, U_n\}$ *is a tuple* $(\mathsf{Setup}, \mathsf{Register}, \mathsf{Signal}, \mathsf{Verify}, \mathsf{Extract}, \mathsf{Slash}, \mathsf{Withdraw})$*, defined relative to a contract holding a Merkle tree of depth* d *and the stake bound to each leaf. For the* n*-party protocols we write* $\langle U_i(\cdot)\rangle_{i\in[n]}$ *for the private input of each* U_i*, a following argument list for the inputs common to all, and a semicolon in the output for the boundary between the common output and* U_i*'s private one.*

- $\mathsf{Setup}(1^\lambda, R_{\mathsf{coRLN}}, d) \to (\mathsf{pk}, \mathsf{vk})$ *takes the security parameter, the relation and the tree depth.*
- $\mathsf{Register}\big\langle U_i(\mathsf{stake}_i, \mathsf{sk}_i)\big\rangle_{i\in[n]} \to (\mathsf{groupIdComm}, \ell, \mathsf{path}\,;\, \mathsf{st}_i)$*: the parties open* $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ *without reconstructing* sk_G *and the contract inserts it at the next free leaf* ℓ *against the pooled* stake_G*, each* U_i *keeping* $\mathsf{st}_i = ([\mathsf{sk}_G]_i, S_i)$ *with* S_i *initially empty.*
- $\mathsf{Signal}\big\langle U_i(\mathsf{st}_i)\big\rangle_{i\in[n]}(\mathsf{pk}, m, e, \mathsf{path}, \mathsf{root}) \to (\sigma\,;\, \mathsf{st}_i')$ *with* $\sigma = (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ *and* $\mathsf{st}_i' = ([\mathsf{sk}_G]_i, S_i \cup \{e\})$*; an honest* U_i *declines when* $e \in S_i$ *and records* e *on* entering *the session, since* y *is opened before the proof is produced.*
- $\mathsf{Verify}(\mathsf{vk}, \sigma, e) \to \{0, 1\}$ *recomputes* $x = H(m)$ *from the message* σ *carries and* $\mathsf{extNull} = H(e)$*, rejects on a mismatch with the* x *in* σ*, and otherwise returns the coSNARK verifier's bit on* $(x, \mathsf{extNull}, \mathsf{root}, y, \mathsf{nullifier})$*, so that the message is bound to the signal rather than transported alongside it.*
- $\mathsf{Extract}(\sigma_1, \sigma_2) \to \mathsf{sk}_G$ *takes two accepted signals sharing a nullifier with* $x_1 \neq x_2$ *and interpolates the line both lie on; acceptance is a precondition on the caller, discharged in step (D1.2) of Algorithm 3.*
- $\mathsf{Slash}(\sigma_1, \sigma_2, \mathsf{sk}_G, \ell) \to \mathsf{stake}_G$ *takes that pair with the extracted secret and the leaf it was signed under; the contract recomputes* $H(\mathsf{sk}_G)$*, checks it sits at* ℓ*, and on success forfeits* stake_G*.*
- $\mathsf{Withdraw}\big\langle U_i(\mathsf{st}_i)\big\rangle_{i\in[n]}(\mathsf{pk}, \mathsf{addr}) \to (\mathsf{groupIdComm}, \pi_w)$*, on which the contract releases* stake_G *to* addr *and removes the leaf.*

Register, Signal *and* Withdraw *are secure with abort, so any party may instead receive* $\perp$*, while* Verify, Extract *and* Slash *take no secret input and may be run by any observer. The rate limit here is one signal per epoch. In the general setting of §4 the syntax changes in five places:* Setup *takes* $R^{(r)}_{\mathsf{coRLN}}$*,* Register *also outputs* r_G*,* Signal *also takes* $\mathsf{messageId} \in [0, r_G)$ *as a common input,* Slash *also takes* r_G *and recomputes the leaf as* $H(H(\mathsf{sk}_G), r_G)$*, and* st_i *records slots* $(e, \mathsf{messageId})$ *rather than epochs. One* Setup *serves every budget, since* r_G *enters the relation as a witness pinned by the rate commitment rather than as a circuit parameter.*

3.1 Group Registration

Registration produces a single leaf in the RLN membership Merkle tree, bound to the group's aggregated stake. Each U_i samples $\mathsf{sk}_i \xleftarrow{\$} \mathbb{F}$ and submits it as its share

$[\mathsf{sk}_G]_i := \mathsf{sk}_i$, so by linearity the parties hold $[\mathsf{sk}_G] = \sum_{i=1}^{n} [\mathsf{sk}_i]$ with no further interaction, and $\mathsf{sk}_G \in \mathbb{F}$ is never reconstructed in an honest execution. They evaluate H on $[\mathsf{sk}_G]$ inside MPC and open the commitment $\mathsf{groupIdComm} := H(\mathsf{sk}_G)$ while sk_G stays shared. Finally each U_i deposits stake_i; the contract aggregates $\mathsf{stake}_G := \sum_i \mathsf{stake}_i$, inserts $\mathsf{groupIdComm}$ as a new leaf bound to stake_G, and returns the leaf index and authentication path, which form a *common witness* shared by every $U_i \in G$. Algorithm 1 gives the full procedure.

Algorithm 1: coRLN-Register (rate-limit-1 setting)

Input: Group $G = \{U_1, \ldots, U_n\}$, individual deposits $(\mathsf{stake}_1, \ldots, \mathsf{stake}_n)$
Output: Leaf index ℓ and authentication path path for G, or $\perp$

1 **(R1.1)** Each $U_i \in G$ samples $\mathsf{sk}_i \xleftarrow{\$} \mathbb{F}$ and submits $[\mathsf{sk}_G]_i := \mathsf{sk}_i$ to the SPDZ network;
2 **(R1.2)** The parties locally hold $[\mathsf{sk}_G] \leftarrow \sum_{i=1}^{n} [\mathsf{sk}_i]$;
3 **(R2.1)** The parties evaluate H on the shared input $[\mathsf{sk}_G]$ inside the MPC network, obtaining a sharing $[\mathsf{groupIdComm}]$ of the digest;
4 **(R2.2)** They open $[\mathsf{groupIdComm}]$ to reveal $\mathsf{groupIdComm} := H(\mathsf{sk}_G)$, while sk_G remains shared;
5 **(R3.1)** Each U_i deposits stake_i against the opened commitment $\mathsf{groupIdComm}$; the contract aggregates $\mathsf{stake}_G := \sum_i \mathsf{stake}_i$ under $\mathsf{groupIdComm}$;
6 **(R3.2)** The contract inserts $\mathsf{groupIdComm}$ at the next free leaf ℓ and binds the aggregated stake_G to ℓ, so the leaf, the stake, and the commitment $\mathsf{groupIdComm}$ are bound together;
7 **(R3.3)** Each U_i reads (ℓ, path) from contract events;
8 **return** (ℓ, path);

3.2 Collaborative Signaling

To send a message m in epoch e, the group jointly produces a coSNARK proving the signal is well-formed under the registered $\mathsf{groupIdComm}$. The relation R_{coRLN} has public instance $(x, \mathsf{extNull}, \mathsf{root})$, where $x = H(m)$, $\mathsf{extNull} = H(e)$ is the per-epoch external nullifier, and root the current Merkle root. Its witness has two parts: a *distributed witness* of secret shares $(\mathsf{sk}_1, \ldots, \mathsf{sk}_n)$, with U_i holding sk_i privately, and a *common witness*, the authentication path path for $\mathsf{groupIdComm}$ held identically by every member after registration. Inside MPC the parties evaluate $[a_1] := H([\mathsf{sk}_G], \mathsf{extNull})$ and $[y] := [\mathsf{sk}_G] + [a_1] \cdot x$ (multiplication by the public x is local), open y, compute $[\mathsf{nullifier}] := H([a_1])$ and open $\mathsf{nullifier}$, keeping sk_G and a_1 shared throughout, since revealing either would let a single party forge a second signal and trigger collective slashing. The parties then run the coSNARK prover for R_{coRLN}, defined by

$$R_{\mathsf{coRLN}}\Big((x, \mathsf{extNull}, \mathsf{root}, y, \mathsf{nullifier});\ (\mathsf{sk}_i)_{i=1}^{n},\ \mathsf{path}\Big) = 1$$

iff there exist sk_G, $\mathsf{groupIdComm}$, and a_1 such that

1. $\mathsf{sk}_G = \sum_{i=1}^{n} \mathsf{sk}_i$,
2. $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$,
3. $\mathsf{MerkleVerify}(\mathsf{root}, \mathsf{groupIdComm}, \mathsf{path}) = 1$,
4. $a_1 = H(\mathsf{sk}_G, \mathsf{extNull})$,
5. $y = \mathsf{sk}_G + a_1 \cdot x$,
6. $\mathsf{nullifier} = H(a_1)$.

The resulting π_{coSNARK} verifies under the standard verification key, and any member broadcasts $\sigma := (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$. Authenticity of m is inherited unchanged from single-party RLN, and it lives in the verification procedure rather than in the relation, which never sees m: Verify recomputes $x = H(m)$ from the message the signal carries and rejects on a mismatch with the x in σ (Definition 1), so the x entering the checked evaluation $y = \mathsf{sk}_G + a_1 x$ is the hash of the message actually delivered. Substituting m' for m therefore either changes the instance, making the result a forgery excluded by Theorem 3, or exhibits a collision in H. A relay that checked π_{coSNARK} against the x carried in σ without recomputing $H(m)$ would accept the honest proof under any message, so the recomputation is mandatory at every verifier. Since x maps arbitrary-length messages into $\mathbb{F}$, the assumption here is collision resistance of the sponge wrapping the permutation, which deployments should domain-separate from the fixed-arity instances used inside the relation. Algorithm 2 gives the full procedure.

Algorithm 2: coRLN-Signal (rate-limit-1 setting)

Input: Message m, epoch e, group $G = \{U_1, \ldots, U_n\}$ holding $[\mathsf{sk}_G]$, common witness path

Output: Signal $\sigma = (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ or $\perp$

(S1.1) Each U_i locally computes $x \leftarrow H(m)$ and $\mathsf{extNull} \leftarrow H(e)$;

(S1.2) The parties jointly compute $[a_1] \leftarrow H([\mathsf{sk}_G], \mathsf{extNull})$ and then $[y] \leftarrow [\mathsf{sk}_G] + [a_1] \cdot x$ inside the MPC;

(S1.3) The parties open $[y]$ to obtain y, then compute $[\mathsf{nullifier}] \leftarrow H([a_1])$ inside the MPC and open the result as $\mathsf{nullifier}$;

(S2) The parties run the coSNARK prover for R_{coRLN} with public instance $(x, \mathsf{extNull}, \mathsf{root}, y, \mathsf{nullifier})$, distributed witness $(\mathsf{sk}_i)_{i=1}^{n}$, and common witness path, producing π_{coSNARK};

(S3) Any U_i broadcasts $\sigma := (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ to the gossip network;

return σ;

3.3 Slashing on Rate Violation

Since the group is one RLN identity, the limit $r = 1$ binds it as a whole: a second same-epoch signal reveals the line on which sk_G lies, and any observer recovers

sk_G from the two points. Because $\mathsf{nullifier} = H(a_1)$ with $a_1 = H(\mathsf{sk}_G, \mathsf{extNull})$ depends only on (sk_G, e), every epoch-e signal carries the same $\mathsf{nullifier}$, so an observer can flag two signals sharing $\mathsf{nullifier}$ but differing in x, verify both proofs, and interpolate $a_1 = (y_2 - y_1)/(x_2 - x_1)$ and then $\mathsf{sk}_G = y_1 - a_1 x_1$ in $O(1)$ field operations. The slasher submits $(\sigma_1, \sigma_2, \mathsf{sk}_G, \ell)$ with ℓ the leaf index of $\mathsf{groupIdComm}$; the contract recomputes $H(\mathsf{sk}_G)$, checks it sits at leaf ℓ under root, and on success transfers stake_G and removes the leaf. Since the stake was registered as one aggregated amount, slashing forfeits the entire group stake in one action, consistent with the one-identity, one-stake treatment of G. Algorithm 3 gives the full procedure.

Algorithm 3: coRLN-Slash (rate-limit-1 setting)

Input: Two signals $\sigma_1 = (m_1, x_1, y_1, \mathsf{nullifier}, \mathsf{root}, \pi_1)$ and $\sigma_2 = (m_2, x_2, y_2, \mathsf{nullifier}, \mathsf{root}, \pi_2)$ with shared $\mathsf{nullifier}$ and $x_1 \neq x_2$; the leaf index ℓ of $\mathsf{groupIdComm}$ under root

Output: Recovered sk_G and slashed amount stake_G, or $\perp$

1 **(D1.1)** An observer flags σ_1, σ_2 as a duplicate-epoch pair sharing $\mathsf{nullifier}$ with $x_1 \neq x_2$;
2 **(D1.2)** The observer verifies both proofs π_1, π_2 under the coRLN verification key (aborting and returning $\perp$ on failure);
3 **(D2.1)** The observer interpolates the line $y = \mathsf{sk}_G + a_1 \cdot x$ from (x_1, y_1) and (x_2, y_2);
4 **(D2.2)** The observer recovers $a_1 \leftarrow (y_2 - y_1)/(x_2 - x_1)$ and $\mathsf{sk}_G \leftarrow y_1 - a_1 \cdot x_1$;
5 **(D3.1)** The slasher submits $(\sigma_1, \sigma_2, \mathsf{sk}_G, \ell)$ to the contract;
6 **(D3.2)** The contract recomputes $\mathsf{groupIdComm} \leftarrow H(\mathsf{sk}_G)$ and verifies that $\mathsf{groupIdComm}$ sits at leaf ℓ under root;
7 **(D3.3)** On success, the contract transfers stake_G to the slasher and removes the leaf;
8 **return** $(\mathsf{sk}_G, \mathsf{stake}_G)$;

4 Generalization to Rate Limit $r \geq 1$

We now lift coRLN to an arbitrary per-epoch budget $r_G \geq 1$, following the standard RLN-v2 trick of distinguishing messages within an epoch by an additional input $\mathsf{messageId}$ rather than raising the polynomial degree. Each message uses a distinct $\mathsf{messageId} \in \{0, 1, \ldots, r_G - 1\}$ folded into the coefficient a_1, and the slashing argument of §3.3 applies *per* $(\mathsf{epoch}, \mathsf{messageId})$ pair: a collision on this pair is the rate-limit violation, and any two such signals interpolate to recover sk_G. The three operations of §3 carry over with small, local changes, each presented as a delta over its $r = 1$ counterpart. Registration and signaling are given in full as Algorithms 4 and 5, with primed step labels distinguishing them from their $r = 1$ counterparts (Algorithms 1 and 2); slashing alters

only its on-chain step and so is described inline rather than restated as a separate algorithm. One new quantity drives every change: the *rate commitment* $\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$, which replaces the bare $\mathsf{groupIdComm}$ as the membership leaf and binds the group to its budget r_G.

4.1 Group Registration with Rate Commitment

Registration proceeds as in §3.1: each U_i samples sk_i, the parties hold $[\mathsf{sk}_G] = \sum_i [\mathsf{sk}_i]$, and the MPC opens $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ while sk_G stays shared. The per-epoch budget scales with the pooled stake, where $\mathsf{stake}_G = \sum_i \mathsf{stake}_i$ and $\mathsf{MIN_DEPOSIT}$ is the per-message stake unit: $r_G := \lfloor \mathsf{stake}_G / \mathsf{MIN_DEPOSIT} \rfloor$. Algorithm 4 gives the full procedure: its steps (R1′)–(R2′) repeat (R1)–(R2) of Algorithm 1 verbatim, and the two changes are confined to the on-chain step (R3′):

1. the budget r_G is derived from the pooled stake, so it scales with the group's collective deposit;
2. the inserted leaf is the rate commitment $\mathsf{groupRateComm}$ rather than the bare $\mathsf{groupIdComm}$.

The contract binds stake_G to the new leaf ℓ and emits $(\ell, \mathsf{path}, r_G)$, where (ℓ, path) is the common witness for membership proofs and r_G the budget now known to every member. Binding r_G into the leaf at registration is what later prevents a group from inflating its budget at signal time.

Algorithm 4: coRLN-Register (rate-limit $r \geq 1$ setting)

Input: Group $G = \{U_1, \ldots, U_n\}$, individual deposits $(\mathsf{stake}_1, \ldots, \mathsf{stake}_n)$
Output: Leaf index ℓ, authentication path path, and budget r_G for G, or $\perp$

(R1′.1) Each $U_i \in G$ samples $\mathsf{sk}_i \xleftarrow{\$} \mathbb{F}$ and submits $[\mathsf{sk}_G]_i := \mathsf{sk}_i$ to the SPDZ network;
(R1′.2) The parties locally hold $[\mathsf{sk}_G] \leftarrow \sum_{i=1}^{n} [\mathsf{sk}_i]$;
(R2′.1) The parties evaluate H on the shared input $[\mathsf{sk}_G]$ inside the MPC network, obtaining a sharing $[\mathsf{groupIdComm}]$ of the digest;
(R2′.2) They open $[\mathsf{groupIdComm}]$ to reveal $\mathsf{groupIdComm} := H(\mathsf{sk}_G)$, while sk_G remains shared;
(R3′.1) Each U_i deposits stake_i against the opened commitment $\mathsf{groupIdComm}$; the contract aggregates $\mathsf{stake}_G := \sum_i \mathsf{stake}_i$ under $\mathsf{groupIdComm}$;
(R3′.2) The contract computes $r_G := \lfloor \mathsf{stake}_G / \mathsf{MIN_DEPOSIT} \rfloor$ and the rate commitment $\mathsf{groupRateComm} := H(\mathsf{groupIdComm}, r_G)$;
(R3′.3) The contract inserts $\mathsf{groupRateComm}$ at the next free leaf ℓ, binding stake_G to that leaf and hence to the rate commitment $\mathsf{groupRateComm}$;
(R3′.4) Each U_i reads $(\ell, \mathsf{path}, r_G)$ from contract events;
return $(\ell, \mathsf{path}, r_G)$;

4.2 Collaborative Signaling with messageId

To send the j-th message of epoch e ($0 \le j < r_G$), the group fixes $\mathsf{messageId} := j$ and runs the signaling procedure of §3.2 with the coefficient now keyed on the counter. Inside MPC the parties evaluate $[a_1] := H([\mathsf{sk}_G], \mathsf{extNull}, \mathsf{messageId})$ and $[y] := [\mathsf{sk}_G] + [a_1] \cdot x$, open y, compute $[\mathsf{nullifier}] := H([a_1])$ and open $\mathsf{nullifier}$, with sk_G and a_1 kept shared as before. Because a_1 now depends on $\mathsf{messageId}$, distinct counters within one epoch evaluate distinct lines, and the underdetermined-system property is preserved up to r_G signals per epoch rather than just one. The parties then run the coSNARK prover for the extended relation

$$R^{(r)}_{\mathsf{coRLN}}\Big((x, \mathsf{extNull}, \mathsf{root}, y, \mathsf{nullifier});\ (\mathsf{sk}_i)_{i=1}^{n},\ \mathsf{path},\ r_G,\ \mathsf{messageId}\Big) = 1,$$

which is the relation R_{coRLN} of §3.2 with three modifications:

1. the coefficient folds in the counter, $a_1 = H(\mathsf{sk}_G, \mathsf{extNull}, \mathsf{messageId})$ in place of $H(\mathsf{sk}_G, \mathsf{extNull})$;
2. the membership check targets the rate commitment: the relation computes $\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$ and proves that $\mathsf{groupRateComm}$ sits under root along path, in place of the bare-$\mathsf{groupIdComm}$ Merkle check;
3. a range proof $0 \le \mathsf{messageId} < r_G$ is added (§4.3).

The remaining constraints ($\mathsf{sk}_G = \sum_i \mathsf{sk}_i$, $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$, $y = \mathsf{sk}_G + a_1 x$, $\mathsf{nullifier} = H(a_1)$) are exactly those of §3.2, and the common witness grows from path to $(\mathsf{path}, r_G, \mathsf{messageId})$. Neither $\mathsf{messageId}$ nor r_G is an input to the public instance: r_G is pinned by the rate-commitment constraint, so it cannot be inflated at signal time without breaking the Merkle proof, while $\mathsf{messageId}$ is left free but range-checked, so the prover may pick any $\mathsf{messageId} \in [0, r_G)$ yet cannot exceed the budget. The budget itself is public, derived from the pooled stake and emitted by the contract at registration (§4.1); what keeping it out of the instance buys is anonymity, a budget carried in the instance narrowing the anonymity set to the leaves holding that budget. The residual leak through the registration event is coarser but real, so deployments should quantize budgets into a few tiers. Operationally this is Algorithm 2 with the modified $[a_1]$ and each U_i supplying the range-check witness from $(\mathsf{messageId}, r_G)$; any member then broadcasts $\sigma := (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$. Algorithm 5 gives the full procedure.

4.3 Range Check on messageId

The range-check modification enforces $0 \le \mathsf{messageId} < r_G$; without it, a group could supply an out-of-range $\mathsf{messageId}$ and obtain a valid-looking proof for an unauthorized signal. We reuse the RangeCheck template of circom-rln, keeping coRLN consistent with the deployed circuit. Writing L for the maximum bit-size LIMIT_BIT_SIZE, both r_G and $\mathsf{messageId}$ are field elements fitting in L bits, and $\mathsf{RangeCheck}(L)$ asserts $0 \le \mathsf{messageId} < r_G$ via two circomlib gadgets:

Algorithm 5: coRLN-Signal (rate-limit $r \geq 1$ setting)

Input: Message m, epoch e, counter $\mathsf{messageId} \in [0, r_G)$ agreed among G, group $G = \{U_1, \ldots, U_n\}$ holding $[\mathsf{sk}_G]$, common witness (path, r_G)
Output: Signal $\sigma = (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ or $\perp$

(S1′.1) Each U_i locally computes $x \leftarrow H(m)$ and $\mathsf{extNull} \leftarrow H(e)$;
(S1′.2) The parties jointly compute $[a_1] \leftarrow H([\mathsf{sk}_G], \mathsf{extNull}, \mathsf{messageId})$ and then $[y] \leftarrow [\mathsf{sk}_G] + [a_1] \cdot x$ inside the MPC;
(S1′.3) The parties open $[y]$ to obtain y, then compute $[\mathsf{nullifier}] \leftarrow H([a_1])$ inside the MPC and open the result as $\mathsf{nullifier}$;
(S2′.1) Each U_i locally derives the range-check witness from $(\mathsf{messageId}, r_G)$ (§4.3), contributing it as part of the common witness;
(S2′.2) The parties run the coSNARK prover for $R^{(r)}_{\mathsf{coRLN}}$ with public instance $(x, \mathsf{extNull}, \mathsf{root}, y, \mathsf{nullifier})$, distributed witness $(\mathsf{sk}_i)_{i=1}^n$, and common witness $(\mathsf{path}, r_G, \mathsf{messageId})$, producing π_{coSNARK};
(S3′) Any U_i broadcasts $\sigma := (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ to the gossip network;
return σ;

Num2Bits, an L-bit decomposition placing each input in $[0, 2^L)$, and LessThan, which compares messageId against r_G.

This block has a direct MPC consequence. Both inputs $\mathsf{messageId}, r_G$ are *common-witness* values, with r_G fixed at registration and messageId agreed out of band (§4.5), so the auxiliary witnesses inside Num2Bits and LessThan are deterministic functions of common inputs with no dependence on the shares sk_i. Each party derives them locally before the SPDZ session, leaving MPC to handle only the secret-dependent constraints that evaluate sk_G, a_1, y, and nullifier. This matches single-party RLN's cost at the range-check block and confines SPDZ communication to the parts genuinely depending on sk_G.

4.4 Slashing on Rate Violation

Slashing proceeds as in §3.3: two signals sharing nullifier with $x_1 \neq x_2$ lie on the same line $y = \mathsf{sk}_G + a_1 x$, and interpolating it recovers sk_G. Two things shift in the $r \geq 1$ setting: what counts as the violation, and the leaf the contract matches (now the rate commitment groupRateComm, not the bare groupIdComm). Under budget r_G, honest behavior emits at most one signal per messageId per epoch; since $a_1 = H(\mathsf{sk}_G, \mathsf{extNull}, \mathsf{messageId})$ depends only on $(\mathsf{sk}_G, e, \mathsf{messageId})$, two signals sharing $(\mathsf{epoch}, \mathsf{messageId})$ share a_1 and hence nullifier, whereas two signals with *different* messageId in one epoch carry different nullifiers and lie on different lines. The rate-limit violation is therefore "two signals with the same $(\mathsf{epoch}, \mathsf{messageId})$" rather than "two signals in one epoch," so r_G legitimate messages per epoch are safe, which is exactly the budget the group paid for.

Detection and interpolation are unchanged from steps (D1)–(D2) of Algorithm 3: an observer flags a pair sharing nullifier with $x_1 \neq x_2$, verifies both

proofs π_1, π_2, and recovers $a_1 \leftarrow (y_2 - y_1)/(x_2 - x_1)$ and $\mathsf{sk}_G \leftarrow y_1 - a_1 x_1$. The two changes are confined to the on-chain step (D3):

1. the slasher additionally supplies r_G, submitting $(\sigma_1, \sigma_2, \mathsf{sk}_G, \ell, r_G)$, and the contract recomputes $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ and the rate commitment $\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$, checking it sits at leaf ℓ under root in place of the bare-$\mathsf{groupIdComm}$ check;
2. the contract confirms the supplied r_G matches the budget bound to ℓ at registration.

On success it transfers the entire stake_G to the slasher and removes the leaf, exactly as in the $r = 1$ case.

4.5 Counter Management

In single-party RLN-v2 counter management is trivial: one device tracks which $\mathsf{messageId}$ values it has consumed and resets at epoch boundaries. In coRLN no single party holds the canonical record; each U_i has its own view, and the parties must agree on the next $\mathsf{messageId}$ before each session. This ties counter handling to slashing: two signals under the same $(\mathsf{epoch}, \mathsf{messageId})$ carry identical nullifiers (§4.4) and forfeit stake_G, whereas distinct values are safe. A failure here, namely two sessions invoked with the same $\mathsf{messageId}$ (whether by two members claiming the same slot or by a deviating subset reusing one), is therefore a direct cause of full forfeiture rather than a soft coordination issue, so robust scheduling is mandatory.

The protocol prescribes no specific mechanism; any out-of-band agreement suffices. Two natural designs are (a) a coordinator proposes the next $\mathsf{messageId}$ and members confirm before the SPDZ session, and (b) a deterministic schedule fixed at registration, e.g. round-robin $\mathsf{messageId}$ slots keyed by member index modulo r_G. Design (b) removes race conditions but cedes liveness when a scheduled member is unavailable, while (a) keeps flexibility at the cost of a coordinator that is a liveness single point of failure. For soundness all that matters is that the parties hold a consistent $\mathsf{messageId}$ as common input when entering the session; the coSNARK enforces the rest.

4.6 Stake Withdrawal

Classical RLN provides a withdrawal circuit [13] that releases a member's bond on exit by taking the member's $\mathsf{identitySecret}$ privately and outputting its hash, matched against the registered leaf (in v2, against the stored rate commitment and message limit). This single-party circuit does not apply, since no party holds sk_G. Instead the group runs a coSNARK over a withdrawal relation proving jointly, without reconstructing sk_G, that (1) $\mathsf{sk}_G = \sum_{i=1}^{n} \mathsf{sk}_i$ and (2) $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$, with $\mathsf{groupIdComm}$ a public output and r_G furnished to the contract. Since $\mathsf{groupIdComm}$ has been on-chain since registration (§4.1), opening it here reveals nothing new. The contract checks

$\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$ against the proof output and the stored r_G, then removes the leaf and releases the bond.

The proof alone must not authorize the release. A withdrawal transaction is public before it is mined, so a relation whose only public data is $\mathsf{groupIdComm}$ yields a transferable authorization: an observer copies π_w from the mempool, resubmits it naming its own account, and collects stake_G. Classical RLN forecloses this by taking the recipient address as a public input of the withdraw circuit [13], and we do the same. Since addr is a public input, the Groth16 verification equation binds π_w to it already; what has to be ensured is that the compiler not discard it as unused, for which we add the dummy constraint $\mathsf{addr} \cdot \mathsf{addr} = \mathsf{addr}^2$ in the circom idiom. The group fixes addr before the session under the same n-out-of-n agreement as every other common value. The same exposure applies to the slashing call, where sk_G is submitted in the clear and a front-runner may resubmit the claim as its own; this is inherited from single-party RLN, whose standard mitigation is a commit-reveal on that call.

The MPC layer is the same secret-dependent evaluation used in signaling (§4.2), only the relation differs. The disbursement policy is enforced by the contract, over addr and over any schedule registered alongside the leaf, and not by the relation, which binds the single recipient it names and nothing further. In the first of two designs, the relation names one addr, agreed before the session, and the split among the members is settled off-chain; the payout stays private, but exit needs fresh n-out-of-n agreement, so one member who refuses to enter, or who enters and aborts (Remark 2), freezes stake_G indefinitely. In the second, a disbursement schedule is registered with the leaf at registration and addr need not enter the relation at all, a copied π_w then still paying the registered members. We take the first as the default, paired with a contract-level timeout that falls back to the second, which removes the hostage risk at the cost of publishing the payout structure.

5 Security Properties

We state the security properties under the threat model of §3: an active adversary corrupting up to $n-1$ members, a collision- and preimage-resistant H (Poseidon), and a correctly-enforcing contract. Each argument is a reduction invoking the collaborative-SNARK composition theorem of Ozdemir and Boneh (Proposition 1, Appendix B.2), instantiated with $R = R^{(r)}_{\mathsf{coRLN}}$ and Π a SPDZ-based prover secure with abort against $t = n-1$ corruptions, combined with the structural properties of $R^{(r)}_{\mathsf{coRLN}}$. Definitions are in Appendix B; proofs are at the reduction level, with full UC-style proofs left to future work (§8.1).

Corruptions are *static*: the adversary fixes $C \subset G$ with $|C| \leq n-1$ before the protocol begins, which is the regime in which the SPDZ and coSNARK simulators invoked below are proven secure. We claim no adaptive security, which would require an adaptively secure MPC backend, or erasures, together with an adaptively simulatable coSNARK. The claims are not *selective*, however: beyond C the adversary commits to nothing in advance, choosing every message, epoch

and counter after observing all earlier sessions. Each property below is stated as an experiment over the oracles of Appendix B.3, which admit polynomially many registrations, so the target group is attacked inside a tree where other groups register, signal and withdraw throughout.

Definition 2 (Security notions for coRLN). *Let Σ be a collaborative RLN scheme (Definition 1) and fix a static $C \subset G$ with $|C| \leq n-1$. The adversary $\mathcal{A}$ receives $(\mathsf{pk}, \mathsf{vk})$ and the shares $\{\mathsf{sk}_i\}_{i \in C}$, and queries the oracles $\mathcal{O} = (\mathcal{O}_{\mathsf{Reg}}, \mathcal{O}_{\mathsf{Sig}}, \mathcal{O}_{\mathsf{Wdr}})$ of Appendix B.3, playing the parties of C and fixing the message, epoch and counter of every session. Those oracles admit polynomially many registrations; G, C, ℓ, S and Q refer throughout to the first, the target, with S its entered slots and Q the signals it output. Each experiment draws $(\mathsf{pk}, \mathsf{vk}) \leftarrow \mathsf{Setup}(1^\lambda, R^{(r)}_{\mathsf{coRLN}}, d)$ and runs $\mathcal{A}^{\mathcal{O}}(\mathsf{pk}, \mathsf{vk}, \{\mathsf{sk}_i\}_{i \in C})$; $\mathsf{inst}(\sigma)$ is a signal's instance, $\mathcal{E}$ the extractor of Proposition 1, and $\mathsf{leaf}(j)$ the value stored at index j.*

- $\mathbf{Exp}^{\mathsf{priv}}_{\Sigma,\mathcal{A}}(\lambda, C)$*: $\mathcal{A}$ outputs $\mathsf{sk}^* \in \mathbb{F}$; return 1 iff $H(\mathsf{sk}^*) = \mathsf{groupIdComm}$.*
- $\mathbf{Exp}^{\mathsf{slash}}_{\Sigma,\mathcal{A}}(\lambda, C)$*: $\mathcal{A}$ outputs $(\sigma_1, e_1, \sigma_2, e_2)$; return 0 unless $\mathsf{Verify}(\mathsf{vk}, \sigma_j, e_j) = 1$ for $j \in \{1, 2\}$, the two signals carry a common root and a common $\mathsf{nullifier}$, and $x_1 \neq x_2$. Otherwise set $\mathsf{sk}^* \leftarrow \mathsf{Extract}(\sigma_1, \sigma_2)$, let ℓ^* be the leaf index opened by the witness $\mathcal{E}$ returns on π_1 and r^* the budget bound to ℓ^* at registration; return 1 iff $\mathsf{leaf}(\ell^*) \neq H(H(\mathsf{sk}^*), r^*)$.*
- $\mathbf{Exp}^{\mathsf{auth}}_{\Sigma,\mathcal{A}}(\lambda, C)$*: $\mathcal{A}$ outputs (σ, e); return 1 iff $\mathsf{Verify}(\mathsf{vk}, \sigma, e) = 1$, $\mathsf{inst}(\sigma) \notin \{\mathsf{inst}(\sigma') : \sigma' \in Q\}$, and the witness $\mathcal{E}$ returns on σ opens the leaf ℓ.*
- $\mathbf{Exp}^{\mathsf{wdr}}_{\Sigma,\mathcal{A}}(\lambda, C)$*: $\mathcal{A}$ outputs $(\mathsf{groupIdComm}^*, \pi^*, \mathsf{addr}^*)$; return 1 iff the contract accepts the triple for the leaf ℓ and $\mathcal{A}$ made no query $\mathcal{O}_{\mathsf{Wdr}}(1, \mathsf{addr}^*)$.*
- $\mathbf{Exp}^{\mathsf{anon}\text{-}b}_{\Sigma,\mathcal{A}}(\lambda)$*, for $b \in \{0, 1\}$: $\mathcal{A}$ names two registrations $k_0 \neq k_1$ with $C^{(k_0)} = C^{(k_1)} = \emptyset$ and $r_G^{(k_0)} = r_G^{(k_1)}$, then outputs $(m, e, \mathsf{messageId})$ with $(e, \mathsf{messageId})$ unentered for both; the challenger runs Signal for $G^{(k_b)}$ on that input and returns the broadcast σ alone, withholding the coordination transcript; $\mathcal{A}$ outputs b'; return b'.*

Set $\mathsf{Adv}^{\mathsf{X}}_{\Sigma,\mathcal{A}}(\lambda) := \Pr[\mathbf{Exp}^{\mathsf{X}}_{\Sigma,\mathcal{A}}(\lambda, C) = 1]$ *for* $\mathsf{X} \in \{\mathsf{priv}, \mathsf{slash}, \mathsf{auth}, \mathsf{wdr}\}$*, and*

$$\mathsf{Adv}^{\mathsf{anon}}_{\Sigma,\mathcal{A}}(\lambda) := \left|\Pr[\mathbf{Exp}^{\mathsf{anon}\text{-}0}_{\Sigma,\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{Exp}^{\mathsf{anon}\text{-}1}_{\Sigma,\mathcal{A}}(\lambda) = 1]\right|.$$

Then Σ satisfies group-secret privacy*,* slashing soundness*,* joint authorization*,* withdrawal authorization *or* signal anonymity *if the corresponding advantage is negligible in λ for every PPT $\mathcal{A}$ and every admissible C.*

Group-secret privacy is an unpredictability rather than an indistinguishability notion: $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ is public from registration, so sk_G is determined information-theoretically by the transcript, and one-wayness under the published commitment is the strongest shape available as well as the one the slashing semantics need. Freshness in $\mathbf{Exp}^{\mathsf{auth}}$ is a condition on the instance and not on the proof string, since Groth16 proofs are publicly re-randomizable and

a forgery defined by $\pi \notin \{\pi_\sigma : \sigma \in Q\}$ would be achievable with no secret at all; message authenticity at group granularity then follows from this notion rather than forming a separate one, since Verify binds m to the instance it checks (§3.2). Signal anonymity is inherited rather than established, being the anonymity of the underlying RLN circuit, which is why its experiment requires equal budgets, no corruption in either group, and withholds the coordination channel of Remark 1. Theorems 1, 2 and 3 bound $\mathsf{Adv}^{\mathsf{priv}}$, $\mathsf{Adv}^{\mathsf{slash}}$ and $\mathsf{Adv}^{\mathsf{auth}}$ in turn, and Corollary 1 bounds $\mathsf{Adv}^{\mathsf{wdr}}$.

Theorem 1 (Group secret privacy). *Suppose at least one party $U_i \in G$ is honest, the underlying SPDZ protocol is secure with abort against $n-1$ active corruptions, the coSNARK is zero-knowledge, and the group emits at most r_G signals per epoch. Then no PPT adversary corrupting any subset $C \subset G$ with $|C| \leq n-1$ can recover sk_G except with negligible probability if we model H as a random oracle.*

Proof. The proof is given in Appendix A.1.

Theorem 1 captures coRLN's motivating guarantee: an adversary compromising up to $n-1$ members cannot recover sk_G, even after many honest signals up to the budget. The budget hypothesis is a property of the interface rather than an assumption on the group's behaviour, since $\mathcal{O}_{\mathsf{Sig}}$ refuses a slot already entered and no scheduling drives the honest parties past r_G entered slots in an epoch; a deployment enforces it through its members. Only *some* honest share need stay hidden; its holder's identity need not be known. This concerns secrecy, not availability: a corrupt subset never recovers sk_G or disturbs the honest share, so $\mathsf{groupRateComm}$ stays usable. Liveness, by contrast, is only n-out-of-n. So any member can stall a session, but none can speak as G. The proof bounds the adversary's view by SPDZ secrecy, coSNARK zero-knowledge with the underdetermined-line property, and preimage resistance of H.

Remark 1. Theorem 1 concerns the broadcast σ and is silent about the SPDZ coordination channel. An adversary observing both networks may correlate MPC-session timing with subsequent broadcasts and conclude a broadcast came from G, collapsing its anonymity set from "any RLN member" to "the group G," though still without revealing sk_G or any share. Deployments seeking strong external anonymity must route SPDZ traffic over an anonymizing network or decouple session timing from broadcasts.

Theorem 2 (Slashing soundness). *Suppose the coSNARK is knowledge-sound per Proposition 1 and H is collision-resistant. Let σ_1, σ_2 be two signals that verify under the coRLN verification key against a common root root, carry the same $\mathsf{nullifier}$, and satisfy $x_1 \neq x_2$. Then, except with probability negligible in λ over collisions in H, the two signals reuse a single $(\mathsf{epoch}, \mathsf{messageId})$ slot of one registered member, and the public algorithm $\mathsf{Extract}$ of Definition 1 outputs, in $O(1)$ field operations, that member's group secret sk_G: the value satisfying $H(H(\mathsf{sk}_G), r_G) = \mathsf{groupRateComm}$ at the leaf ℓ its proofs open under root,*

where r_G is the budget bound to ℓ at registration. The slasher reads ℓ off the matching leaf and r_G off the registration event, both public, and on submission of $(\sigma_1, \sigma_2, \mathsf{sk}_G, \ell, r_G)$ the contract's checks pass and it forfeits the entire stake_G. In the rate-limit-1 setting of §3.3 the same statement holds with the bare commitment as the leaf, the extracted value then satisfying $H(\mathsf{sk}_G) = \mathsf{groupIdComm}$.

Proof. The proof is given in Appendix A.2.

Theorem 3 (Joint authorization). *Suppose at least one party $U_i \in G$ is honest and the coSNARK is knowledge-sound, and let H be as in Theorem 1. Then coRLN satisfies joint authorization in the sense of Definition 2: a PPT adversary corrupting any $C \subset G$ with $|C| \leq n-1$ and querying $\mathcal{O}_{\mathsf{Reg}}$, $\mathcal{O}_{\mathsf{Sig}}$ and $\mathcal{O}_{\mathsf{Wdr}}$ produces, except with negligible probability, no signal σ that verifies under the coRLN verification key, carries an instance returned by none of its $\mathcal{O}_{\mathsf{Sig}}$ calls, and whose witness opens the leaf ℓ registered to G. Nothing is claimed about leaves other than ℓ, under which the adversary may register and signal at will.*

Proof. Knowledge soundness turns a fresh verifying σ that opens ℓ into a witness for the relation, hence into a preimage of the commitment stored at ℓ, so it suffices that sk_G stay hidden from C. The shares alone are $(n-1)$-private over $\mathbb{F}$, but that does not make sk_G statistically hidden during the protocol: registration publishes $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$, which determines sk_G information-theoretically, and every $\mathcal{O}_{\mathsf{Sig}}$ answer publishes $y = \mathsf{sk}_G + a_1 x$ with x public. Secrecy of sk_G against C is therefore computational from registration onward and is supplied by Theorem 1, whose loss the bound here inherits. The freshness condition rules out the trivial attack, a corrupted member rebroadcasting a σ from a session it joined.

Corollary 1 (Withdrawal authorization). *Under the hypotheses of Theorem 3, coRLN satisfies withdrawal authorization in the sense of Definition 2.*

Proof. The withdrawal relation proves $\mathsf{sk}_G = \sum_i \mathsf{sk}_i$ and $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ with $\mathsf{groupIdComm}$ public, so a π_w accepted for ℓ yields, by knowledge soundness, a preimage of the commitment stored at ℓ, and Theorem 3 applies verbatim. The clause on addr^* follows from the binding instead: since addr is a public input (§4.6), a pair (π'_w, addr') accepted for ℓ is a proof on a different instance, so knowledge soundness again yields a preimage at ℓ, while the honest π_w copied from the mempool verifies only against the addr it names.

Remark 2. Theorem 3 implies a dual property: any single party can prevent signal production by refusing to join the SPDZ session, so liveness in coRLN is n-out-of-n. A corrupted minority cannot forge signals, but a corrupted singleton can deny service; threshold variants (t-out-of-n) are discussed in §8.1.

Theorem 1 models H as a random oracle, following the deployed RLN analyses, while the instantiation is Poseidon; the idealization is heuristic, and doubly so here, since what makes H cheap in MPC and in R1CS is algebraic structure

that a random oracle lacks. Only the privacy argument invokes it, to program a_1 at an unqueried point and to extract a preimage of groupIdComm. Theorem 2 rests on collision resistance alone and survives under a standard-model hash; Theorem 3 does not, drawing secrecy of sk_G from Theorem 1. A concrete one-wayness assumption on Poseidon would remove the idealization; we leave this open.

Two aspects of the soundness argument deserve comment. First, Proposition 1 establishes only *distributed* knowledge: an extractor recovers the joint witness, not that any individual U_i knows their share. As Ozdemir and Boneh [12] note, this is intrinsic to the non-interactive syntax and is partly circumvented by having R enforce structural properties of the witness, which constraint (1) does by forcing $\mathsf{sk}_G = \sum_i \mathsf{sk}_i$ and so requiring n shares. Second, Garg et al. [7] show that invalid witnesses can leak honest provers' inputs under the "semi-honest compiled to malicious" template; this concerns the MPC compiler for Π, not the verifier or R, so it motivates a leakage-resilient backend without undermining our theorems.

6 Use Cases

coRLN's defining feature is that a group G shares one externally-visible identity, one rate budget, and one slashable stake, with signaling gated behind joint MPC participation. We present four application classes, each highlighting a distinct property: *joint authorization* (§6.1), *collective slashing* (§6.2), *stake pooling under Sybil-resistance* (§6.3), and *group-controlled subscriptions* (§6.4). In each of them the naive workarounds of §1 fail where the group-as-one-identity model succeeds.

6.1 Organizational and Coalition Anonymous Speech

Many anonymous speakers are organizations or coalitions: a newsroom publishing leaks, an NGO issuing statements under threat, a DAO working group, or a whistleblower coalition corroborating a story. The natural unit of authorship is the group, and an unauthorized statement in its name is a serious harm, reputational for organizations and potentially physical for whistleblowers. Since sk_G exists only as shares, a leaked laptop, compromised individual, or subpoenaed member cannot speak unilaterally, and the budget is fixed at registration regardless of n, so adding members never amplifies the group's voice. The cost, a loss of liveness when a member is unavailable, is exactly what an authorization layer for high-stakes speech should impose.

6.2 Multi-Operator Infrastructure with Collective Bonds

Decentralized infrastructure is increasingly run by multi-operator committees: bridge councils signing attestations, oracle committees publishing price feeds, or sequencer sets issuing status updates. Such a committee needs a public channel

whose messages reflect joint authorization, appear as one sender, and are backed by a collective bond against spam or conflicting messages. coRLN folds into one what would otherwise be a threshold signature bolted onto a separate spam-prevention mechanism: SPDZ participation *is* the authorization layer and RLN slashing *is* the spam-prevention layer, both bound by the shared sk_G. Because any violation forfeits stake_G regardless of cause, operator hygiene becomes a shared financial concern.

6.3 Stake-Pooled Memberships under Sybil-Resistance

In stake-gated RLN, MIN_DEPOSIT forces a tension: a low value makes the budget cheap and weakens spam protection, while a high value excludes participants who cannot afford a full membership. coRLN offers a third option. Small participants register as one group, pooling deposits into one stake_G and one leaf, so the deployment sees a single high-stake member with one identity, budget $r_G = \lfloor \mathsf{stake}_G / \mathsf{MIN_DEPOSIT} \rfloor$, and one bond. This case is purely economic, since members share no joint statement, so only registration and slashing semantics matter, not joint authorization. Sybil-resistance is preserved, as a pool gets the same r_G as a single member of equal stake; grouping yields no budget multiplication, and pooling lowers the entry barrier without weakening the security argument.

6.4 Group-Controlled Subscriptions across Service Providers

Anonymous subscriptions let a user authenticate repeated access without revealing identity across sessions. Lee et al. [10] formalize the single-service primitive and Davidson et al. [4] extend unlinkable token redemption across origins via Privacy Pass, but neither addresses one subscription redeemable across *independent* providers with protocol-level privacy and decentralized credential control. coRLN supplies all three: n providers (e.g. metered AI inference endpoints) honor one shared subscription whose holder may issue up to r_G queries per epoch against any provider, each carrying a membership proof checked against the shared budget without revealing identity. Making the subscriber a group means no member queries alone, the allowance stays bounded by r_G independent of $|G|$, and overspending interpolates sk_G and forfeits stake_G for all.

7 Cost Analysis

coRLN reuses single-party RLN unchanged on the verifier and chain side: $R^{(r)}_{\mathsf{coRLN}}$ is the RLN-v2 circuit augmented only by the linear constraint $\mathsf{sk}_G = \sum_i \mathsf{sk}_i$, so proof size, signal format, verification, and the slashing call are exactly RLN-v2's (with the verification key updated), all $O(1)$ and independent of n. Slashing needs no MPC, since any observer recovers sk_G by interpolation in $O(1)$ (§4.4). The only new cost is producing each signal jointly, analyzed below; we leave end-to-end benchmarks to an extended version.

Per signal the parties evaluate the secret-dependent part under MPC and prove it with a collaborative SNARK. Writing m_H for the MPC multiplications per Poseidon (driven by its x^5 S-boxes), the secret-dependent work is three Poseidon evaluations, the in-circuit re-derivation $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ together with a_1 and $\mathsf{nullifier}$, hence $O(m_H)$ multiplications, independent of d and of r_G. The depth-d path contributes nothing, because its accumulator starts at $\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$, $\mathsf{groupIdComm}$ was opened at registration (R2′.2), and r_G and the siblings are read from contract events (R3′.4), so every wire along it is a function of values all parties know and is assigned locally, exactly as for the range check (§4.3). The commitment is still recomputed in-circuit, since revealing the leaf would break anonymity, and that recomputation is the one hash on the path MPC must carry, its input being the shared sk_G; from its output, the public $\mathsf{groupIdComm}$, the rest of the path is local. The remaining per-signal arithmetic is local: $y = \mathsf{sk}_G + a_1 x$ is multiplication-free as x is public, and r_G adds no secret-dependent cost at all. The collaborative prover then communicates $O(n\,|C|)$ for $|C| = O(d)$ [12], the bottleneck whose dependence on n Liu et al. [11] reduce. The number of communication *rounds*, by contrast, is set by the multiplicative depth of that secret-dependent part, so it too is constant in d and n: only the genuine secret×secret multiplications inside the Poseidon S-boxes interact, so latency is the same for any group size. The price of the group setting is thus paid once, off-chain at proving time, growing with n only through MPC communication and preprocessing, never through round complexity.

Table 1 states the resulting comparison against single-party RLN-v2 in the circuit and protocol parameters: c_H and m_H are the R1CS constraints and MPC multiplications per Poseidon call, D_H its multiplicative depth, L the range-check width, d the tree depth, n the group size, and $|C|$ the constraint count of the circuit. The secret-dependent row is constant in tree depth, budget and group

Table 1. Cost of coRLN against single-party RLN-v2 in circuit and protocol parameters. The added constraint $\mathsf{sk}_G = \sum_i \mathsf{sk}_i$ is linear, so it contributes no multiplicative gate, only $n-1$ witness variables.

	RLN-v2	**coRLN**
Constraints $\|C\|$	$(d{+}4)c_H + O(L)$	same, $+\,n{-}1$ witness vars
Proof size, verifier cost	$O(1)$	identical
Signal format, slashing call	RLN-v2	identical
Poseidon calls under MPC	—	3
MPC rounds	—	$O(D_H)$
MPC communication	—	$O(n\,\|C\|)$

size. The communication row is not, and it is the quantity an implementation must measure, since the constant hidden in $O(n\,|C|)$ decides whether joint signaling is interactive-latency or batch-latency work. The measurement is taken over the circom/Groth16 instantiation of $R^{(r)}_{\mathsf{coRLN}}$ with a SPDZ backend, and reports wall-clock proving time and bytes sent per party at $n \in \{2, 4, 8, 16\}$ and

$d \in \{16, 20, 24\}$, preprocessing separated from the online phase, each against a single-party RLN-v2 prover on the same circuit and machine; verification time and proof size are reported once, being identical by construction.

8 Conclusions

We presented coRLN, which lifts RLN from a single-party to a group-authorized setting while preserving the externally-visible identity model, the membership tree, the slashing argument, and the public signal interface. Composing SPDZ additive sharing with collaborative zk-SNARKs, it holds sk_G only as shares, computes the identity commitment in MPC and registers it as one leaf, produces every signal jointly, and on any violation interpolates to recover sk_G and forfeit the collective bond. We developed the rate-limit-1 and general $r \geq 1$ settings and a withdrawal procedure that releases the bond without reconstructing sk_G, and reduced three properties to the collaborative-SNARK composition: privacy of sk_G against $n-1$ corruptions, an extractor on duplicate-$(\mathsf{epoch}, \mathsf{messageId})$ signals, and unforgeability by any strict subset, together giving n-out-of-n authorization with collective accountability.

The construction has explicit limitations: no individual blame attribution, since a violation forfeits the whole pooled stake; n-out-of-n liveness requiring every member; counter management whose failure mode is full slashing (§4.5); and broadcast privacy that does not extend to the SPDZ channel (Remark 1). The collective-slashing semantics are not incidental but the property that makes the protocol useful for §6: by tying authorization and rate-limiting to one shared sk_G rather than composing them at the application layer, coRLN gives a clean primitive for group-authorized anonymous signaling that interoperates with existing RLN infrastructure with only the verification key updated.

8.1 Future Work and Open Directions

Several extensions are left for future work.

- Threshold authorization. Replacing additive sharing with a t-out-of-n scheme would restore liveness under partial unavailability at the cost of relaxing the strict joint-authorization guarantee.
- Membership maintenance. A proactive secret-sharing-style refresh that adds, removes, or rotates members without disturbing the on-chain leaf would avoid full re-registration on group changes.
- Formal security proofs. Full UC-style proofs that integrate SPDZ's abort behavior with the coSNARK security model and the on-chain contract semantics would put the construction on firmer footing.
- Stake-split policies. A general framework for parameterizing the disbursement policy at withdrawal time would let deployments choose between pro-rata, single-coordinator, and off-chain settlement.

A Proofs of Theorems

This appendix collects the proofs of Theorems 1 and 2 of Section 5. The proofs are reductions to the collaborative-SNARK composition theorem (Proposition 1, recalled in Appendix B.2) and to the standard primitives underneath: SPDZ secrecy, additive secret sharing over $\mathbb{F}$, and collision-/preimage-resistance of the hash function H. Both are stated against the oracles $\mathcal{O}_{\mathsf{Reg}}$, $\mathcal{O}_{\mathsf{Sig}}$ and $\mathcal{O}_{\mathsf{Wdr}}$ of Appendix B.3. The proof of Theorem 3 is given in-line in Section 5 and combines knowledge soundness with secrecy of sk_G against C, which is the computational guarantee of Theorem 1 rather than the $(n-1)$-privacy of additive sharing, the latter covering the shares alone and not the published $\mathsf{groupIdComm}$.

A.1 Proof of Theorem 1

Proof. We prove the statement in the rate-limit-1 setting, where the group emits at most one signal per epoch; the general case $r_G > 1$ follows by the remark at the end. Suppose that there exists a PPT adversary $\mathcal{A}$ that, while corrupting some subset $C \subset G$ with $|C| \leq n-1$, outputs sk_G with non-negligible probability $\varepsilon(\lambda)$ after polynomially many queries to $\mathcal{O}_{\mathsf{Reg}}$, $\mathcal{O}_{\mathsf{Sig}}$ and $\mathcal{O}_{\mathsf{Wdr}}$ (Definition 2), which in the rate-limit-1 setting admit at most one entered slot per epoch. We show that the existence of $\mathcal{A}$ contradicts at least one of the assumptions of the theorem, namely zero-knowledge of the coSNARK (Proposition 1), security with abort of SPDZ against $n-1$ corruptions, or the hardness of inverting H modeled as a random oracle. We proceed via a sequence of games and a final reduction. Throughout, we model H as a random oracle, in line with the deployed RLN proofs based on Poseidon. We let $q = q(\lambda)$ denote a polynomial bound on the total number of random-oracle queries that $\mathcal{A}$ makes.

Game G_0 (real interaction). $\mathcal{A}$ queries $\mathcal{O}_{\mathsf{Reg}}$, $\mathcal{O}_{\mathsf{Sig}}$ and $\mathcal{O}_{\mathsf{Wdr}}$, each answered by running the real protocol: real SPDZ sessions, real coSNARK proofs, real broadcasts. Let

$$\varepsilon_0 := \Pr[\mathcal{A} \text{ outputs } \mathsf{sk}_G \text{ in } G_0] = \varepsilon(\lambda).$$

Game G_1 (simulated proofs). Identical to G_0, except every π_{coSNARK} returned by $\mathcal{O}_{\mathsf{Sig}}$ or $\mathcal{O}_{\mathsf{Wdr}}$ is replaced by an output of the coSNARK simulator $\mathcal{S}_{\mathsf{coSNARK}}(\mathsf{vk}, \mathsf{x})$ on the public instance only. Let $\varepsilon_1 := \Pr[\mathcal{A} \text{ outputs } \mathsf{sk}_G \text{ in } G_1]$. If $|\varepsilon_0 - \varepsilon_1|$ were non-negligible, then $\mathcal{A}$ would yield a distinguisher between real and simulated coSNARK proofs, contradicting zero-knowledge of the coSNARK (Proposition 1). Hence $|\varepsilon_0 - \varepsilon_1| = \mathsf{negl}(\lambda)$.

Game G_2 (simulated SPDZ transcript). Identical to G_1, except the SPDZ transcript each oracle returns to C is replaced by the output of the SPDZ simulator $\mathcal{S}_{\mathsf{SPDZ}}$ guaranteed by security with abort, taking as input only $\{\mathsf{sk}_i : i \in C\}$ and that call's public output ($\mathsf{groupIdComm}$ from $\mathcal{O}_{\mathsf{Reg}}$ and $\mathcal{O}_{\mathsf{Wdr}}$, the pair $(y, \mathsf{nullifier})$ from $\mathcal{O}_{\mathsf{Sig}}$). Let $\varepsilon_2 := \Pr[\mathcal{A} \text{ outputs } \mathsf{sk}_G \text{ in } G_2]$. If $|\varepsilon_1 - \varepsilon_2|$ were non-negligible, then $\mathcal{A}$ would yield a distinguisher between real and simulated SPDZ transcripts, contradicting SPDZ security with abort against $n-1$ active corruptions. Hence $|\varepsilon_1 - \varepsilon_2| = \mathsf{negl}(\lambda)$.

In G_2, $\mathcal{A}$'s view depends on the honest share sk_{i^*} (for $i^* \in G \setminus C$) only through the public outputs of its oracle calls: $\mathsf{groupIdComm}$ on-chain and, per $\mathcal{O}_{\mathsf{Sig}}$ call, the pair $(y, \mathsf{nullifier})$ with $y = \mathsf{sk}_G + a_1 \cdot x$, $a_1 = H(\mathsf{sk}_G, \mathsf{extNull})$, and $\mathsf{nullifier} = H(a_1)$. All other elements of $\mathcal{A}$'s view are simulator outputs that are independent of sk_{i^*} by construction.

We construct from $\mathcal{A}$ an adversary $\mathcal{B}$ that inverts H on a fresh challenge with probability negligibly close to ε_2. $\mathcal{B}$ receives a preimage challenge $c = H(z)$ for $z \xleftarrow{\$} \mathbb{F}$ and simulates G_2 to $\mathcal{A}$ as follows:

- On the first $\mathcal{O}_{\mathsf{Reg}}$ call, $\mathcal{B}$ lets $\mathcal{A}$ sample the corrupted-party shares $\{\mathsf{sk}_i\}_{i \in C}$ at step (R1) as usual and, at (R2)/(R3), sets $\mathsf{groupIdComm} := c$, returning $(\mathsf{groupIdComm}, \ell, \mathsf{path}, r_G)$. This is consistent with $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ for the unknown $\mathsf{sk}_G = z$, on which $\mathcal{B}$ holds no share. On every later $\mathcal{O}_{\mathsf{Reg}}$ call, and on any $\mathcal{O}_{\mathsf{Sig}}$ or $\mathcal{O}_{\mathsf{Wdr}}$ call naming another group, $\mathcal{B}$ runs the real protocol, sampling those groups' honest shares itself; those sessions are perfectly simulated and cost nothing in the bound, the challenge being embedded in the first registration alone.
- On a call $\mathcal{O}_{\mathsf{Sig}}(1, m, e, \mathsf{messageId})$, $\mathcal{B}$ first applies the oracle's own bookkeeping, returning $\perp$ if the slot $(e, \mathsf{messageId})$ already lies in S and entering it otherwise. It then sets $x := H(m)$ and $\mathsf{extNull} := H(e)$, samples $a_1, y \xleftarrow{\$} \mathbb{F}$ uniformly, sets $\mathsf{nullifier} := H(a_1)$, and returns $\sigma = (m, x, y, \mathsf{nullifier}, \mathsf{root}, \pi_{\mathsf{coSNARK}})$ with the proof supplied by $\mathcal{S}_{\mathsf{coSNARK}}$; if $\mathcal{A}$ aborts the session, $\mathcal{B}$ returns what was opened before the abort and the slot stays spent. In the real protocol this coefficient equals $H(\mathsf{sk}_G, \mathsf{extNull})$, whose first argument is the unknown preimage $\mathsf{sk}_G = z$; $\mathcal{B}$ therefore cannot evaluate or assign H at this point a priori. Instead it simulates the random oracle by on-demand sampling, maintaining a table that records the value returned on each queried input, so that a fresh input receives an independent uniform value and a repeated input is answered consistently. The sampled a_1 is taken to define $H(z, \mathsf{extNull})$ implicitly; this assignment is compatible with the sampling discipline and is committed to the table only if that point is subsequently queried.
- On the target group's call $\mathcal{O}_{\mathsf{Wdr}}(1, \mathsf{addr})$, $\mathcal{B}$ returns $(\mathsf{groupIdComm}, \pi_w) = (c, \mathcal{S}_{\mathsf{coSNARK}}(\mathsf{vk}, \mathsf{x}_w))$, exposing the same c as at registration and nothing further. Every oracle answer carries the SPDZ transcript produced by $\mathcal{S}_{\mathsf{SPDZ}}$ on $\{\mathsf{sk}_i\}_{i \in C}$ and that call's public output, as in G_2.
- $\mathcal{B}$ monitors every random-oracle query of $\mathcal{A}$: whenever a queried input contains a field element w with $H(w) = c$, $\mathcal{B}$ halts and outputs w as a preimage of c. This both extracts the preimage and keeps the implicit assignment above consistent, since the only way $\mathcal{A}$ can reach the point $H(z, \mathsf{extNull})$ is to query the unknown z, which this rule detects.

The view $\mathcal{B}$ presents to $\mathcal{A}$ is statistically indistinguishable from G_2: in G_2 the value $\mathsf{sk}_G \in \mathbb{F}$ is uniform conditioned on $\mathcal{A}$'s view, which contains the honest share only through the published $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ and the per-signal

pair $(y, \mathsf{nullifier})$; in the random-oracle model $\mathsf{groupIdComm}$ leaves sk_G uniform on the inputs $\mathcal{A}$ has not queried, and the remaining shares are independent of it. Hence, conditional on (a_1, x), the value $y = \mathsf{sk}_G + a_1 \cdot x$ is uniform on $\mathbb{F}$, matching $\mathcal{B}$'s sampling. The values $a_1 = H(\mathsf{sk}_G, \mathsf{extNull})$ are uniform conditional on $\mathcal{A}$'s view because sk_G is unknown to $\mathcal{A}$ and the random-oracle input has never been queried by $\mathcal{A}$ for any other purpose; this last property may fail only if $\mathcal{A}$ guesses sk_G in one of its q random-oracle queries, an event of probability at most $q/|\mathbb{F}|$. Since $\mathcal{O}_{\mathsf{Sig}}$ refuses a slot already in S and enters the slot before the session opens y, each line $y = \mathsf{sk}_G + a_1 \cdot x$ is observed at exactly one (x, y) point whether that session completed or aborted, so $\mathcal{A}$ cannot interpolate any line.

If $\mathcal{A}$ outputs sk'_G with $H(\mathsf{sk}'_G) = \mathsf{groupIdComm} = c$, then $\mathcal{B}$ outputs sk'_G as a preimage of c. By the indistinguishability of $\mathcal{B}$'s simulation from G_2, $\mathcal{B}$ succeeds with probability at least $\varepsilon_2 - q/|\mathbb{F}| - \mathsf{negl}(\lambda)$.

Let $Q = Q(\lambda)$ be a polynomial bound on the total number of random-oracle queries $\mathcal{B}$ makes, counting those it answers on $\mathcal{A}$'s behalf together with its own evaluations and monitoring checks. In the random-oracle model the challenge image $c = H(z)$ is uniform, so $\mathcal{B}$ inverts c with probability at most $Q/|\mathbb{F}|$. Together with the lower bound $\varepsilon_2 - q/|\mathbb{F}| - \mathsf{negl}(\lambda)$ on $\mathcal{B}$'s success, this gives $\varepsilon_2 \leq (Q+q)/|\mathbb{F}| + \mathsf{negl}(\lambda)$. Combined with $|\varepsilon_0 - \varepsilon_1| = \mathsf{negl}(\lambda)$ and $|\varepsilon_1 - \varepsilon_2| = \mathsf{negl}(\lambda)$, we obtain $\varepsilon_0 = \varepsilon(\lambda) \leq (Q+q)/|\mathbb{F}| + \mathsf{negl}(\lambda)$, which is negligible in λ since Q and q are polynomial and $|\mathbb{F}|$ is exponential. This contradicts the assumption that ε is non-negligible. Therefore no such PPT adversary $\mathcal{A}$ exists.

For $r_G > 1$, the same argument applies with only notational changes. The coefficient becomes $a_1 = H(\mathsf{sk}_G, \mathsf{extNull}, \mathsf{messageId})$, and each epoch contributes up to r_G signals carrying distinct counters $\mathsf{messageId} \in \{0, \ldots, r_G - 1\}$, which evaluate r_G distinct lines.

A.2 Proof of Theorem 2

Proof. Let σ_1, σ_2 be two signals verifying against a common root root and carrying the same nullifier $\mathsf{nullifier}$, with $x_1 \neq x_2$; neither a shared epoch nor a shared counter is assumed, both being forced below. Write $\mathsf{extNull}_j$ for the epoch nullifier in the public instance of π_j, so that $\mathsf{extNull}_j = H(e_j)$ for the epoch e_j against which σ_j was verified. By knowledge soundness of π_1, π_2 (Proposition 1), each signal has a witness satisfying the relation, giving values $\mathsf{sk}_G^{(j)}, a_1^{(j)}, \mathsf{messageId}_j$ with $y_j = \mathsf{sk}_G^{(j)} + a_1^{(j)} x_j$ and $a_1^{(j)} = H(\mathsf{sk}_G^{(j)}, \mathsf{extNull}_j, \mathsf{messageId}_j)$ for $j \in \{1, 2\}$.

The Poseidon instances of arity one, two and three, all three of which the argument uses, are domain-separated as $\mathsf{circom\text{-}rln}$ [13] instantiates them, so no collision across arities is available to the adversary. The shared nullifier gives $H(a_1^{(1)}) = H(a_1^{(2)})$, so $a_1^{(1)} = a_1^{(2)} =: a_1$ by collision resistance. A second application, to $a_1 = H(\mathsf{sk}_G^{(j)}, \mathsf{extNull}_j, \mathsf{messageId}_j)$, equates the preimage tuples, yielding $\mathsf{sk}_G^{(1)} = \mathsf{sk}_G^{(2)} =: \mathsf{sk}_G$ together with $\mathsf{extNull}_1 = \mathsf{extNull}_2$ and $\mathsf{messageId}_1 = \mathsf{messageId}_2$, and a third, to $\mathsf{extNull}_j = H(e_j)$, gives $e_1 = e_2$. The two signals therefore belong to one member and reuse a single $(\mathsf{epoch}, \mathsf{messageId})$ slot, the violation the budget r_G forbids.

Knowledge soundness returns, alongside $\mathsf{sk}_G^{(j)}$, a path path_j with $\mathsf{MerkleVerify}(\mathsf{root}, \mathsf{groupRateComm}^{(j)}, \mathsf{path}_j) = 1$ for $\mathsf{groupRateComm}^{(j)} = H(H(\mathsf{sk}_G^{(j)}), r_G^{(j)})$. Since $\mathsf{sk}_G^{(1)} = \mathsf{sk}_G^{(2)}$, a fourth application of collision resistance, now to the arity-two compression at each of the d levels, gives that $\mathsf{groupRateComm}^{(1)}$ is the value stored at the leaf path_1 addresses and $r_G^{(1)}$ the budget bound there at registration, so the r_G the slasher supplies is checked against a pinned value. The two paths address different leaves only if the same $H(\mathsf{sk}_G)$ was registered twice, which the contract rejects at insertion.

The evaluations $y_1 = \mathsf{sk}_G + a_1 x_1$ and $y_2 = \mathsf{sk}_G + a_1 x_2$ form a non-singular system since $x_1 \neq x_2$, with solution

$$a_1 = \frac{y_2 - y_1}{x_2 - x_1}, \qquad \mathsf{sk}_G = y_1 - a_1 x_1,$$

computable in $O(1)$ field operations from the public parts of σ_1, σ_2. The extractor returns this sk_G. By the relation it satisfies $\mathsf{groupIdComm} = H(\mathsf{sk}_G)$ and $\mathsf{groupRateComm} = H(\mathsf{groupIdComm}, r_G)$, where $\mathsf{groupRateComm}$ sits at the leaf ℓ under root fixed above; the contract recomputes both, checks r_G against the budget bound to ℓ at registration, and forfeits the entire stake_G. For $r_G = 1$ the leaf is the bare $\mathsf{groupIdComm}$ and the second hash is omitted.

B Cryptographic Preliminaries

For completeness, this appendix recalls the formal definitions of zk-SNARKs and collaborative zk-SNARKs underlying the security argument of §5. We follow the formulations of [12].

B.1 zk-SNARKs

Let R be an NP relation: a set of pairs (x, w) where x is the public instance and w the witness, and where membership of (x, w) in R is decidable in polynomial time. We write $\mathcal{L}_R := \{\mathsf{x} : \exists w,\ (\mathsf{x}, w) \in R\}$ for the language of R.

Definition 3 (zk-SNARK). *A* zero-knowledge succinct non-interactive argument of knowledge *(zk-SNARK) for R is a triple of probabilistic polynomial-time algorithms* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *where:*

- $\mathsf{Setup}(1^\lambda, R) \to (\mathsf{pk}, \mathsf{vk})$ *takes a security parameter λ and the relation R and outputs a proving key* pk *and a verification key* vk*;*
- $\mathsf{Prove}(\mathsf{pk}, \mathsf{x}, w) \to \pi$ *takes the proving key, an instance, and a witness, and outputs a proof π;*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \pi) \to \{0, 1\}$ *takes the verification key, an instance, and a proof, and outputs an accept/reject bit.*

The triple satisfies the following properties:

1. *Completeness. For all* $(\mathsf{x}, w) \in R$, $\Pr\big[\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \mathsf{Prove}(\mathsf{pk}, \mathsf{x}, w)) = 1\big] = 1$ *over the randomness of* Setup *and* Prove.
2. *Knowledge soundness. For every PPT adversary* $\mathcal{A}$*, there exists a PPT extractor* $\mathcal{E}$ *such that, for all* x,$\Pr\big[\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \pi) = 1 \ \wedge \ (\mathsf{x}, w) \notin R\big]$ *is negligible in* λ*, where* $\pi \leftarrow \mathcal{A}(\mathsf{pk}, \mathsf{x})$ *and* $w \leftarrow \mathcal{E}^{\mathcal{A}}(\mathsf{pk}, \mathsf{x})$.
3. *Zero-knowledge. There exists a PPT simulator* $\mathcal{S}$ *such that, for every* $(\mathsf{x}, w) \in R$*, the distributions* $\mathsf{Prove}(\mathsf{pk}, \mathsf{x}, w)$ *and* $\mathcal{S}(\mathsf{vk}, \mathsf{x})$ *are computationally indistinguishable.*
4. *Succinctness. The proof size* $|\pi|$ *and the verifier runtime are bounded by a polynomial in* λ *and a polylogarithmic factor in* $|w|$.

B.2 Collaborative zk-SNARKs

In a collaborative zk-SNARK, the witness $w = (w_1, \ldots, w_n)$ is split across n parties: party P_i holds w_i and no party holds the full w. Proof generation is realized by replacing the single-party prover Prove with an MPC protocol Π that the n parties execute jointly. The verifier interface remains unchanged.

Definition 4 (Collaborative zk-SNARK). *A* collaborative zk-SNARK *for an* n*-party-witness relation* $R(\mathsf{x}; w_1, \ldots, w_n)$ *is a triple* $(\mathsf{Setup}, \Pi, \mathsf{Verify})$ *where:*

- $\mathsf{Setup}(1^\lambda, R) \to (\mathsf{pk}, \mathsf{vk})$ *is the same as for the underlying zk-SNARK;*
- Π *is an* n*-party MPC protocol; party* P_i *inputs* w_i*, all parties input the public* $(\mathsf{pk}, \mathsf{x})$*, and the protocol outputs a proof* π *to all parties;*
- $\mathsf{Verify}(\mathsf{vk}, \mathsf{x}, \pi) \to \{0, 1\}$ *is the same as for the underlying zk-SNARK.*

The triple satisfies completeness, knowledge soundness, and zero-knowledge as in Definition 3, with security holding against PPT adversaries that may corrupt up to $t < n$ *parties during the execution of* Π.

The following proposition, due to Ozdemir and Boneh [12], shows that a zk-SNARK can be lifted to a collaborative zk-SNARK by replacing its prover with any sufficiently secure MPC protocol that computes the same function. This is the result we invoke in §5.

Proposition 1 (Ozdemir–Boneh, [12, Prop. 8]). *Let* $(\mathsf{Setup}, \mathsf{Prove}, \mathsf{Verify})$ *be a zk-SNARK satisfying Definition 3 for an NP relation* R *with split witness, and let* Π *be an* n*-party MPC protocol that securely computes* Prove *with abort against an active adversary corrupting up to* $t < n$ *parties. Then* $(\mathsf{Setup}, \Pi, \mathsf{Verify})$ *is a collaborative zk-SNARK for* R *in the sense of Definition 4, satisfying completeness, knowledge soundness, and zero-knowledge.*

For the purposes of §5, we instantiate Proposition 1 with $R = R^{(r)}_{\mathsf{coRLN}}$ (or R_{coRLN} in the rate-limit-1 case), and Π a SPDZ-based MPC for the corresponding prover, secure with abort against $t = n - 1$ active corruptions.

B.3 The coRLN Experiments

The notions of Definition 2 are stated over the oracles below, in the general setting $r_G \geq 1$; the rate-limit-1 case is the specialization $r_G = 1$, $\mathsf{messageId} = 0$. Fix a static $C^{(k)} \subset G^{(k)}$ with $|C^{(k)}| \leq n-1$ for each registration k. In every oracle the honest parties run their protocol code while $\mathcal{A}$ supplies the messages of each corrupted member and may abort at any point, and every oracle returns the transcript seen by the corrupted parties together with the public output, so the SPDZ coordination transcript is part of $\mathcal{A}$'s view. Write S_k for the slots $G^{(k)}$ has entered and Q_k for the signals it has output.

- $\mathcal{O}_{\mathsf{Reg}}(G^{(k)})$, callable polynomially many times, runs $\mathsf{Register}$ for $G^{(k)}$ and returns the public $(\mathsf{groupIdComm}^{(k)}, \ell^{(k)}, \mathsf{path}^{(k)}, r_G^{(k)})$.
- $\mathcal{O}_{\mathsf{Sig}}(k, m, e, \mathsf{messageId})$ returns $\perp$ if $G^{(k)}$ is unregistered or has withdrawn, if $\mathsf{messageId} \notin [0, r_G^{(k)})$, or if $(e, \mathsf{messageId}) \in S_k$; otherwise it adds $(e, \mathsf{messageId})$ to S_k, runs Signal for $G^{(k)}$ on $(m, e, \mathsf{messageId})$, and on completion returns σ and adds it to Q_k.
- $\mathcal{O}_{\mathsf{Wdr}}(k, \mathsf{addr})$, callable once per group, runs $\mathsf{Withdraw}$ and returns $(\mathsf{groupIdComm}^{(k)}, \pi_w)$; later $\mathcal{O}_{\mathsf{Sig}}$ calls naming k return $\perp$.

The first call fixes the target: $G = G^{(1)}$, $C = C^{(1)}$, $\ell = \ell^{(1)}$, $S = S_1$, $Q = Q_1$. Each $C^{(k)}$ is fixed statically and may be empty, so the tree carries honest, partly corrupted and target groups at once, and the guarantees of §5 are obtained in a tree where other groups act throughout the attack. These oracles cover exactly the sessions an honest party takes part in; $\mathcal{A}$ may also register, signal and withdraw under leaves of its own by transacting with the contract directly, and the tree the oracles operate over is the one resulting from all such activity. This is what Theorem 3 means in pinning the forgery to ℓ and claiming nothing about other leaves.

A slot is recorded in S_k when a session is *entered* and not when it completes, and the protocol forces this: Signal opens y before the coSNARK is produced (§3.2), so an adversary aborting after the opening already holds one point of the line for $(e, \mathsf{messageId})$, and honest parties re-entering that slot would publish a second. An aborted session therefore consumes its slot exactly as a completed one does, which is what the proof of Theorem 1 assumes in counting one point per line. Only privacy, joint authorization and withdrawal authorization are stated against these oracles; slashing soundness quantifies over any two signals verifying against a common root, whoever produced them, so it needs no honest-execution interface.

References

1. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: CRYPTO 2004, LNCS 3152, pp. 41–55. Springer (2004)
2. Chaum, D., van Heyst, E.: Group signatures. In: EUROCRYPT 1991, LNCS 547, pp. 257–265. Springer (1991)

3. Damgård, I., Pastro, V., Smart, N.P., Zakarias, S.: Multiparty computation from somewhat homomorphic encryption. In: CRYPTO 2012, LNCS 7417, pp. 643–662. Springer (2012)
4. Davidson, A., Goldberg, I., Sullivan, N., Tankersley, G., Valsorda, F.: Privacy pass: Bypassing internet challenges anonymously. PoPETs (2018)
5. Doerner, J., Kondi, Y., Lee, E., shelat, a., Tyner, L.: Threshold BBS+ signatures for distributed anonymous credential issuance. In: IEEE S&P 2023, pp. 773–789. IEEE (2023)
6. Flamini, A., Lee, E., Lysyanskaya, A.: Multi-holder anonymous credentials from BBS signatures. In: CRYPTO 2025. IACR ePrint 2024/1874
7. Garg, S., Goel, A., Jain, A., Roberts, B., Sekar, S.: Malicious security in collaborative zk-SNARKs. In: CRYPTO 2025. IACR ePrint 2025/1026
8. Grassi, L., Khovratovich, D., Rechberger, C., Roy, A., Schofnegger, M.: Poseidon: A new hash function for zero-knowledge proofs. In: USENIX Security 2021
9. Gurkan, K., Jie, K.W., Whitehat, B.: Semaphore: Zero-knowledge signaling on Ethereum. ZKProof Workshop 3 community proposal (2020).
10. Lee, M.Z., Dunn, A.M., Waters, B., Witchel, E., Katz, J.: Anon-pass: Practical anonymous subscriptions. In: IEEE S&P 2013, pp. 319–333. IEEE (2013)
11. Liu, X., Zhou, Z., Wang, Y., Pang, Y., He, J., Zhang, B., Yang, X., Zhang, J.: Scalable collaborative zk-SNARK and fully distributed proof delegation. In: 34th USENIX Security (2025)
12. Ozdemir, A., Boneh, D.: Experimenting with collaborative zk-SNARKs: Zero-knowledge proofs for distributed secrets. In: 31st USENIX Security (2022)
13. Rate-Limiting Nullifier: circom-rln: Reference circom implementation of RLN. https://github.com/Rate-Limiting-Nullifier/circom-rln
14. Rate-Limiting Nullifier Documentation. https://rate-limiting-nullifier.github.io/rln-docs/
15. Revuelta, A., Tikhomirov, S., Challani, A., Cornelius, H., Vivier, S.P.: Message latency in Waku Relay with Rate Limiting Nullifiers. In: DLT 2024. IACR ePrint 2024/1073
16. Sonnino, A., Al-Bassam, M., Bano, S., Meiklejohn, S., Danezis, G.: Coconut: Threshold issuance selective disclosure credentials. In: NDSS 2019. Internet Society (2019)
17. Taheri-Boshrooyeh, S., Thorén, O., Whitehat, B., Koh, W.J., Kilic, O., Gurkan, K.: WAKU-RLN-RELAY. In: IEEE ICDCS Workshops 2022, pp. 73–78. IEEE (2022)
18. Vac: 32/RLN-V1: Rate Limiting Nullifier specification. Vac RFC (2022)