17/WAKU2-RLNRELAY: Privacy-Preserving Peer-to-Peer Economic Spam Protection

Sanaz Taheri Boshrooyeh
Vac Research
Status Research and Development, Singapore

Oskar Thorén
Vac Research
Status Research and Development, Singapore

Barry Whitehat
Unaffiliated

Wei Jie Koh
Independent

Onur Kilic
Unaffiliated

Kobi Gurkan
cLabs

Abstract

This presentation gives an overview of 17/WAKU2-RLNRELAY, a privacy-preserving peer-to-peer economic spam protection mechanism. 17/WAKU2-RLNRELAY is an extension of 11/WAKU2-RELAY i.e., the transport layer of 10/WAKU2 messaging protocol stack. 11/WAKU2-RELAY is, a gossip-based pubsub protocol and is an extension of libp2p GossipSub. In 17/WAKU2-RLNRELAY, we utilize Rate Limiting Nullifiers (RLN) and zkSNARKs to enable this p2p privacy-preserving and economic spam protection mechanism on top of 11/WAKU2-RELAY. 17/WAKU2-RLNRELAY addresses the performance and privacy issues of the state-of-the-art p2p spam prevention techniques including peer scoring (utilized by libp2p), and proof-of-work (used by e.g. Whisper). The spam protection works by limiting the messaging rate of each network participant through rate-limiting nullifiers. To enforce the rate limit, we adopt the suggested framework of Semaphore, however, we modify that framework to properly address the unique requirements of a network of p2p resource-restricted users. The current work dives into the end-to-end integration of Semaphore into 17/WAKU2-RLNRELAY, the modifications required to make it suitable for resource-limited users, and the open problems and future research directions. We also provide a proof-of-concept implementation of 17/WAKU2-RLNRELAY (available in the nim-waku codebase [8]), specifications [22] together with a rough performance evaluation.

2012 ACM Subject Classification Information systems → Spam detection; Networks → Peer-to-peer protocols; Networks → Peer-to-peer networks; Security and privacy → Pseudonymity, anonymity and untraceability; Information systems → Chat; Networks → Routing protocols; Security and privacy → Privacy-preserving protocols

Keywords and phrases P2P, Spam Protection, Messaging, zkSNARKs, Zero-Knowledge, Anonymity, Routing, Pub/Sub, Gossipsub

Digital Object Identifier 10.4230/OASIcs.Tokenomics.2021.1

1 Introduction

10/WAKU2 [18] is a family of peer-to-peer (p2p) protocols for anonymous and privacy-preserving communication. It is designed to be able to run in resource-restricted environments. Being p2p means that 10/WAKU2 relies on no central server. Instead, peers collaboratively

deliver messages in the network. 11/WAKU2-RELAY constitutes the transport layer of 10/WAKU2 and aims at being privacy-preserving in which no one knows the owner and the receiver of a message except the two ends of the communication. Many of the design choices in this layer are centered around the anonymity requirement. 11/WAKU2-RELAY is a gossip-based pubsub protocol and a thin layer over the libp2p GossipSub [1] routing protocol. Its gossip-based structure enables a great level of receiver anonymity [32]. Peers in 11/WAKU2-RELAY congregate around topics they are interested in and can send messages to topics. Each message gets delivered to all peers subscribed to the topic. Each peer has a constant number of direct connections/neighbors. To publish a message, the author forwards its message to a subset of neighbors. The neighbors proceed similarly till the message gets propagated in the network of the subscribed peers.

In addition to 11/WAKU2-RELAY, 10/WAKU2 features other types of protocols for running in resource-restricted environments. Among which are the request/response protocols of 12/WAKU2-FILTER [20] which is a light-weight version of 11/WAKU2-RELAY for devices with limited bandwidth, and 13/WAKU2-STORE [21] by which resourceful peers can persist and offer historical messages to the querying nodes. Details of these protocols are out of the scope of this paper and can be found in the Vac RFCs [17].

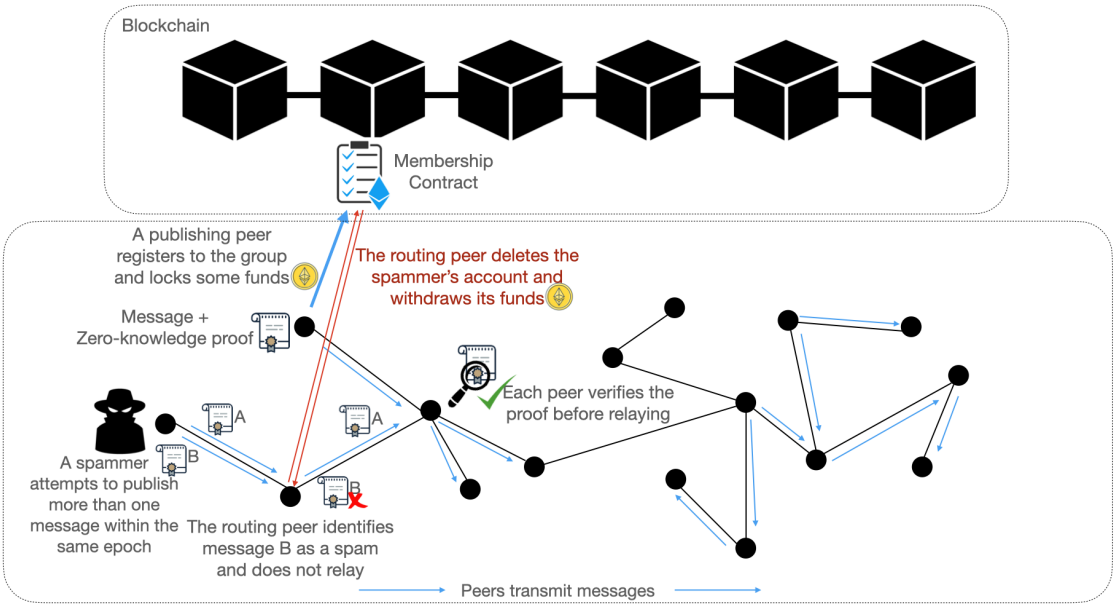

Figure 1 An overview of privacy-preserving p2p economic spam protection in 17/WAKU2-RLNRELAY protocol.

As an open messaging network, 11/WAKU2-RELAY is prone to spam messages. A spammer usually indicates an entity that uses the messaging system to send an *unsolicited* message (spam) to *large numbers of recipients*. However, in 11/WAKU2-RELAY with an open and gossip-based structure, spam messages not only affect the recipients but also all the other peers involved in the routing process where they have to spend their resources e.g., computational power, bandwidth, and storage capacity on processing spam messages. As such, we define a spammer as an entity that uses the messaging system to publish a large number of messages in a short amount of time, in other words, has a high messaging rate.

We define messages issued in this way as spam. Furthermore, this definition disregards the intention of the spammer as well as the content of the message and the number of intended recipients.

The state-of-the-art p2p spam protection techniques for messaging systems i.e., *Proof of Work* (POW) [27] deployed by e.g, Whisper [2] and *Peer scoring* [7] method adopted by libp2p are not suitable for resource-constrained environments and also demonstrate privacy issues. The PoW technique imposes a high computational cost for messaging hence devices with limited resources won't be able to participate and benefit from the messaging system. The peer scoring method is prone to censorship and is also subject to inexpensive attacks where the spammer can send bulk messages by deploying millions of bots. The centralized spam protection methods exhibit privacy issues where they usually ask users to disclose and commit to some piece of personally identifiable information e.g., phone number and email address at the registration time. In addition to this, the central provider is aware of messages owned and received by a particular user which is against privacy.

The economic-incentive spam protection mechanism of 17/WAKU2-RLNRELAY aims at coping with the aforementioned issues where

1. It suits **p2p** systems and does not rely on any central entity.
2. It is **efficient** i.e., with no unreasonable computational, storage, memory, and bandwidth requirement, as such, it fits the network of *heterogeneous* peers with limited resources.
3. It respects users' **privacy** unlike reputation-based and centralized methods. It also allows identification and removal of spammers while not requiring personally identifiable information from the participants.
4. It deploys **economic-incentives** to contain spammers' activity. Namely, there is a financial sacrifice for those who want to spam the system. This disincentives spam activity. Additionally, there is financial incentive to monitor the network for spam and remove the offending user.

At a high level, 17/WAKU2-RLNRELAY guarantees no one can publish more than a certain number of messages, namely, M messages per epoch, without being financially charged. In addition to financial punishment, the spammer gets removed from the network and will not be able to publish further messages. In 17/WAKU2-RLNRELAY, we set M to 1. The epoch can be every second, as defined by UTC date-time $\pm 20s$. The margin of ± 20 seconds is to account for the network transmission delay. Nevertheless, the values for M and epoch are not fixed and can be configured according to the application's needs.

2 Related Work

The studies around spam protection resulted in the development of various techniques and methods. However, each has its own trade-offs and use-case limitations. An overview of the state-of-the-art spam prevention methods is presented below. In this overview, we distinguish between techniques that are targeted for centralized messaging systems and those for p2p ecosystems.

2.1 Centralized Messaging Systems

In traditional centralized messaging systems, spam usually signifies unsolicited messages sent in bulk or messages with malicious content like malware. Protection mechanisms generally aim at making it expensive and difficult for the user to register or to send bulk messages. Such techniques include:

- Authentication through some piece of personally identifiable information e.g., phone number.
- Checksum-based filtering to protect against messages sent in bulk.
- Challenge-response systems.
- Content filtering on the server or via a proxy application.

These methods exploit the fact that the messaging system is centralized and a global view of the users' activities is available based on which spamming patterns can be extracted and defeated accordingly. Moreover, users are associated with an identifier e.g., a username which enables the server to profile each user e.g., to detect suspicious behavior like spamming. Such profiling possibility is against user anonymity and privacy and is easier to censor.

Among the techniques enumerated above, authentication through phone numbers is a somewhat economic incentive measure as providing multiple valid phone numbers will be expensive for the attacker. Notice that while using an expensive authentication method can reduce the number of accounts owned by a single spammer, cannot address the spam issue entirely. This is because the spammer can still send bulk messages through one single account. For this approach to be effective, a centralized mediator is essential. That is why such a solution would not fit decentralized and p2p environments where no central control exists.

2.2 P2P Systems

The state-of-the-art p2p spam prevention methods in messaging systems are *Proof of Work (POW)* [27] deployed by e.g., Whisper [2] and *Peer scoring* [7] method (namely reputation-based approach) adopted by libp2p. However, each of these solutions has its own shortcomings for real-life use-cases as explained below.

2.2.1 Proof of work

The idea behind the Proof Of Work (POW) [27] is to make messaging a computationally costly operation hence lowering the messaging rate of all the peers including the spammers. Specifically, the message publisher has to solve a puzzle and the puzzle is to find a nonce such that the hash of the message concatenated with the nonce has at least z leading zeros. z is known as the difficulty of the puzzle. Since the hash function is one-way, peers have to brute-force to find a nonce. Hashing is a computationally heavy operation so is brute force. While solving the puzzle is computationally expensive, it is comparatively cheap to verify the solution.

POW is also used as the underlying mining algorithm in Ethereum and Bitcoin blockchain. There, the goal is to contain the mining speed and allow the decentralized network to come to a consensus, or agree on things like account balances and the order of transactions.

While the use of POW makes perfect sense in public blockchains such as Ethereum and Bitcoin, it shows practical issues in heterogeneous p2p messaging systems with resource-restricted peers. Some peers won't be able to carry the designated computation and will be effectively excluded. Such exclusion showed to be practically an issue in applications like Status [24], which used to rely on POW for spam protection, to the extent that the difficulty level had to be set close to zero.

2.2.2 Peer Scoring

The peer scoring method [7] that is utilized by libp2p is to limit the number of messages

issued by a peer in connection to another peer. That is each peer monitors all the peers to which it is directly connected and adjusts their messaging quota i.e., to route or not route their messages depending on their past activities. For example, if a peer detects its neighbor is sending more than x messages per month, can drop its quota to $z \cdot x$ where z is less than one. The shortcoming of this solution is that scoring is based on peers' local observations and the concept of the score is defined in relation to one single peer. This leaves room for attacks where a spammer can make connections to k peers in the system and publishes $k \cdot (x-1)$ messages by exploiting all of its k connections. Another attack scenario is through botnets consisting of a large number of e.g., a million bots. The attacker rents a botnet and inserts each of them as a legitimate peer to the network and each can publish $x-1$ messages per month [10]. Another issue with the peer scoring method is that it is prone to censorship where malicious peers give arbitrary scores to their direct connections to make them look like spammers and prevent their messages from reaching the rest of the network.

3 Preliminaries

3.1 Semaphore

Semaphore [29] is a zero-knowledge signaling framework on Ethereum. It allows a set of users to broadcast arbitrary signals (where signal is any value like a string, vote, etc.) while proving they are among a group of authorized users without disclosing their identities. Use-cases of Semaphore are anonymous authentication and private voting. It also utilizes external nullifiers to prevent double-signaling. External nullifier can be seen as a voting booth where each user can only cast one vote [23]. Casting a second vote for the same booth will be rejected. Similarly, each signal is bound to an external nullifier, and each group member is allowed to signal only once for that external nullifier. While nullifiers can limit the signal rate per user, there is no way to identify and remove users who violate this rate limit. In an attempt to address this shortcoming, an extended version using Shamir Secret Sharing (SSS) [31] has been proposed. In the extended variant [4], if a user attempts more than one signal for the same external nullifier, it reveals its identity/private key by which it has registered to the group. As such, the identified identity key can be removed from the group and the user won't be able to signal any more. The removed user will be also financially punished. The user initially deposits some fund when joining the group, and the fund will be rewarded to anyone who identifies double signaling of that user.

3.2 Semaphore with Shamir Secret Sharing

In this section, we present an overview of the extended version of Semaphore that utilizes Shamir secret sharing [31] to enable identification of spammers.

Each member has a private key sk and a public key $pk = H(sk)$ where H is a Cryptographic hash function. These are also called identity key and the identity commitment, respectively. The list of group members are stored in an *Identity Commitment Tree* which is a Merkle tree whose leaves are members public keys. The root of tree is denoted by τ. The membership of a user in the identity commitment tree is proven by providing a branch of the tree that connects the root to that leaf corresponding to the user's pk. This branch is called *authentication path* and we denote it by $auth$. The tree is stored on a smart contract deployed on the Ethereum blockchain. Each user additionally needs to deposit some funds in the contract at the time of registration.

Signaling is bound to a publicly known *external nullifier* denoted by $\emptyset$. When publishing

signal m, a group member utilizes (2,n)-Shamir secret sharing to derive a share (x, y) of its private identity key sk where $x = H(m)$ and $y = sk + H(sk, \emptyset) * x$. (x, y) will be published alongside with the signal m.

The publishing user also calculates an *internal nullifier* ϕ as $\phi = H(H(sk, \emptyset))$ and publishes it together with the signal m. Finally, a user needs to prove in zero-knowledge manner that

1. its secret key sk belongs to the identity commitment tree namely i.e., provides proof of membership.
2. (x, y) is a valid share of its identity key
3. the internal nullifier ϕ is correctly calculated

The above items are proven through zkSNARKs where the circuit represents the aforementioned constraints. The public inputs to the zero-knowledge proof system are the external nullifier $\emptyset$, internal nullifier ϕ, the share of identity secret key (x, y) and the tree root τ whereas the private inputs provided by the message owner are the identity secret key sk and the authentication path $auth$. In 17/WAKU2-RLNRELAY, we utilize Groth16[28] for the proof system. The parameter generation is done through a multi-party setup [30, 16, 6, 14].

The user submits its signal m, identity key share (x, y) and internal nullifiers ϕ to the contract to be stored and accessible by the rest of group members. In fact the contract maintains the state of all the group signals together with all the metadata that can be used to identify double signaling. If a user attempts two different signals for the same external nullifier $\emptyset$, their internal nullifiers will collide, which signifies a double signaling attempt. Furthermore, the two shares of the user's identity key can be used to reconstruct its sk and remove the user from the group. The sk reconstruction stems in the fact that each line (polynomial of degree 1) can be uniquely reconstructed by using two distinct points on it. In the case of double signaling, the spammer reveals two distinct shares (x, y) and (x', y') on line $y = sk + H(sk, \emptyset) * x$, which enables the reconstruction of the line and its evaluation at $x = 0$ which is sk. The user who reconstructs the sk also gets rewarded by a portion of the slashed user's stake. This can be done by passing the recovered sk to a function of the contract.

4 Construction

In this section, we describe the flow of the economic-incentive spam detection mechanism in 17/WAKU2-RLNRELAY from the viewpoint of a single peer. An overview of this flow is provided in Figure 2.

4.1 Overview of 17/WAKU2-RLNRELAY vs Semaphore

17/WAKU2-RLNRELAY adopts the extended variant of Semaphore which provides the additional capability of identifying spammers. However, further adjustments are required to make it fit a p2p routing system. Following is the list of these adjustments that mainly affect the state of the contract as detailed next.

1. In 17/WAKU2-RLNRELAY the state of the contract keeps a simple ordered list of users' identity commitments (instead of Merkle tree) and the Merkle tree is kept off-chain by individual peers. This is in contrast to the original setting of Semaphore where the contract holds the entire commitment tree. The reason for this design shift is to mitigate the significant computational cost/ gas consumption associated with member insertion and deletion which is logarithmic in the number of registered members. While the cost

associated with insertion in Semaphore could be amortized by using batch insertion, this solution is not applicable on the deletion since deletion affects random leaves of the tree which cannot be necessarily batched together. 17/WAKU2-RLNRELAY mitigates this problem as insertion and deletion modify a single item of the list.

2. In 17/WAKU2-RLNRELAY, users' messages and their metadata are not stored in the contract which is in contrast to Semaphore. In 17/WAKU2-RLNRELAY messages are stored off-chain and are distributed through the 11/WAKU2-RELAY routing protocol. It has the benefits of 1) having higher massage propagation speed and 1) being more economic (messaging is for free) as opposed to the on-chain message store. In the on-chain message storage of Semaphore, published messages will not be visible until blocks containing those message transactions get mined. This results in an unnecessary and undesirable delay which is not acceptable for messaging systems with 1.1 million messages per second [26]. In an attempt to address this issue, 17/WAKU2-RLNRELAY decouples the message propagation and storage from the contract state and provides a p2p and off-chain medium for message transportation i.e., 11/WAKU2-RELAY [19] and storage i.e., 13/WAKU2-STORE [21]. The off-chain storage, adopted by 17/WAKU2-RLNRELAY, has another advantage of being more economic. That is it saves the financial cost (related to the gas consumption) associated with message insertion into the contract state. This cost for one-time messaging scenarios like voting systems can be tolerable, however, it is far from practical in a messaging application where there are millions of messages transmitted per second [26].

Due to these adjustments, sending messages in 17/WAKU2-RLNRELAY is for free i.e., does not need gas consumption. Furthermore, message transmission is not affected by the underlying blockchain and its consensus layer. Such separation allows utilization of various optimization techniques on the message transmission delay which would be otherwise impossible due to reliance on the blockchain.

4.2 Setup and Registration

A peer willing to publish a message is required to register. Registration is moderated through a smart contract deployed on the public Ethereum blockchain. The state of the contract contains the list of registered members' public keys. An overview of registration is illustrated in Figure 2.

For the registration, a peer creates its identity private key sk and its commitment $pk = H(sk)$ and sends a transaction to the contract to registers its identity commitment pk in the group. The transaction also transfers the v amount of Ether to the contract. This amount is deposited on the contract to prevent spam activity.

4.3 Maintaining the identity commitment tree

The construction and maintenance of the identity commitment tree, unlike the original proposal of Semaphore [29], is delegated to the peers. As we previously mentioned, the reason is that the cost associated with member deletion and insertion is high and unreasonable. As such, each peer needs to build the tree locally and listens to the contract's events i.e., peer insertion and deletion and updates its tree accordingly.

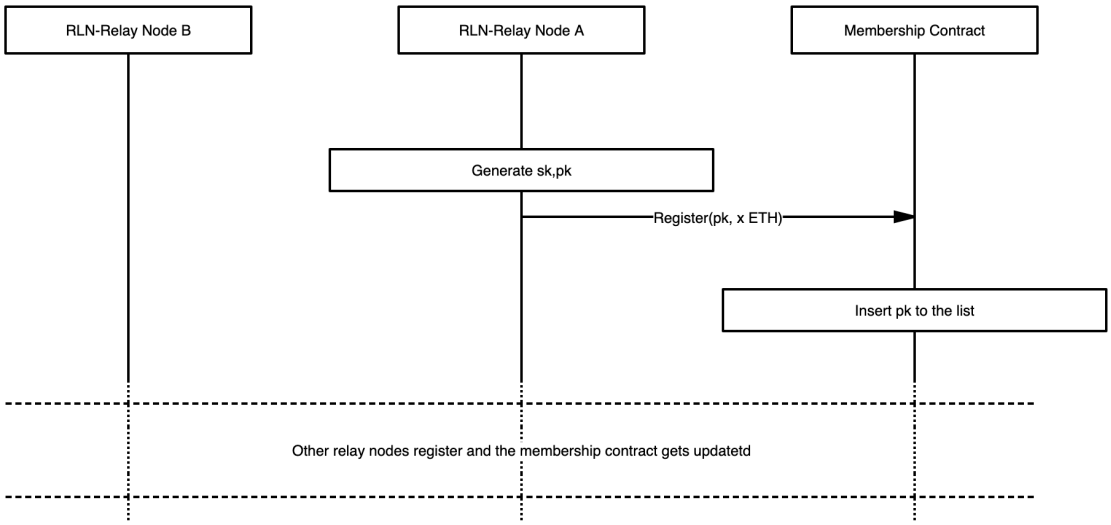

Figure 2 Overview of registration process.

4.4 External nullifier

We use *epoch* as the external nullifier which is defined as the number of T seconds that elapsed since the Unix epoch. The *epoch* is incremented every T seconds and is calculated as $epoch = UnixEpoch/T$ where $UnixEpoch$ is the number of seconds since the Unix epoch. Each peer locally keeps track of the current *epoch*. Peers are allowed to publish one message per *epoch*.

4.5 Publishing

Each peer is allowed to send one message m per *epoch* without being slashed/financially punished. The financial punishment is that the fund deposited by the spammer is rewarded to the peer reporting spammer's activity. In order to prove that the peer is not a spammer and has not violated the messaging rate for the current epoch, the peer is required to generate some metadata and send them alongside the message m. The metadata includes all the public inputs to the zkSNARKs as explained in 3.2 i.e., a share of the peer's identity secret key i.e., (x, y), the internal nullifier ϕ, and the external nullifier $\emptyset = epoch$ and the root of identity commitment tree τ and zkSNARKs proof π. The peer then sends the message bundle $(m, (x, y), \phi, epoch, \tau, \pi)$ to its direct connections as instructed by the routing algorithm i.e., 11/WAKU2-RELAY. An overview of the publishing procedure is provided in Figure 3.

Due to the privacy concerns, the publishing peers must always stay updated with the state of the group (the current members) and make sure they use the latest tree root as well as the authentication path w.r.t. that root for the proof generation. Using an old tree root allows inference about the index of the peer's pk in the tree hence its authentication path.

4.6 Routing and Slashing

Peers follow the regular routing protocol of 11/WAKU2-RELAY and in addition check the metadata of each relayed message to identify and spot spam messages and slash spammers. The routing procedure is depicted in Figure 3.

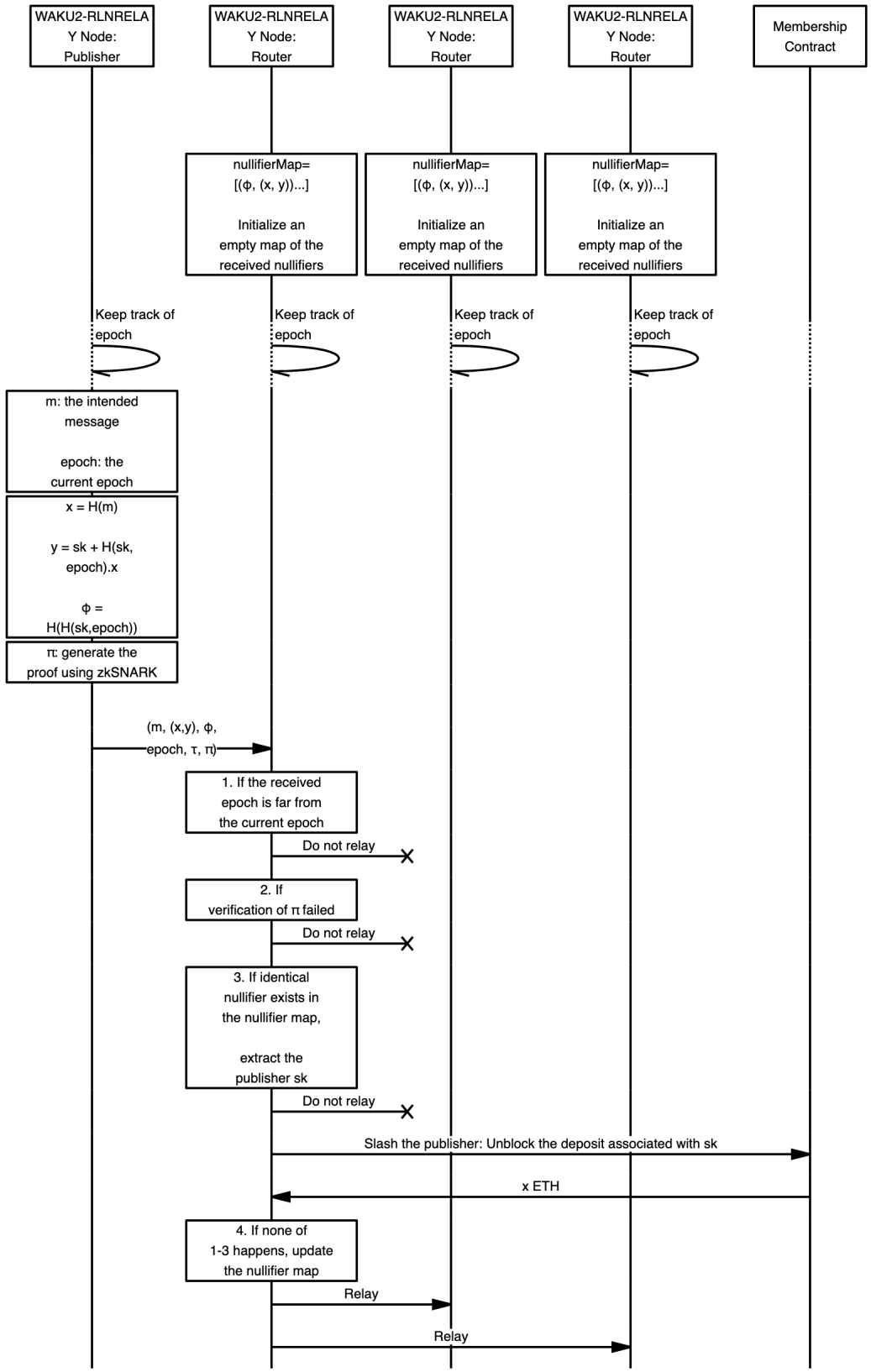

Figure 3 Overview of publishing, routing and slashing.

Upon the receipt of a message $(m, (x, y), \phi, epoch, \tau, \pi)$, the routing peer needs to decide whether to route it or not. The decision relies on the following factors:

1. If the *epoch* value attached to the message has more than Thr gap with the routing peer's current epoch, the message is considered invalid and must be dropped. This is to prevent a newly registered peer from spamming the system by messaging for all the past epochs.
2. The message must contain valid proof π that gets verified by the routing peer.
3. The messaging rate is no violated.

If the preceding checks are passed successfully, then the message is relayed. In case of invalid proof, the message is dropped. If spamming is detected, the publishing peer gets slashed.

In order to identify spam messages, each routing peer keeps a local record of the identity key share (x, y) and the internal nullifier ϕ of all of its valid incoming message bundles for the past Thr epochs. This list is called nullifier map. The routing peer checks every new message against this list to spot spam messages. This list does not have to capture the entire history because any incoming message whose epoch is older than the last Thr epochs is discarded by default (as explained in item 1 of section 4.6). The routing peer utilizes this list to locally identify spam messages and spammers as follows.

1. The routing peer first verifies the π and discards the message if not verified.
2. It checks for the presence of another past message with identical internal nullifier i.e., ϕ. If nothing is found, then the message gets relayed, otherwise:
   a. If the identity share i.e., (x', y') of the older message is different from the incoming message share i.e., $(x, y) \neq (x', y')$, then slashing takes place
   b. If $(x, y) = (x', y')$, then the message is a duplicate and should be discarded.

There is a possibility of race condition to slash an identified spammer. The race happens when a peer submits the identified identity key of the spammer to the contract in plain format and at the same time, another peer observes and copies the same information to reclaim the spammer's fund. This yields a race issue. To avoid this, the commit and reveal technique can be utilized, i.e., the routing peer sends a commitment to the spammer's identity key (instead of the plain identity key) to the contract so that no one else can replicate and reclaim it. Later it opens the commitment and reveals its knowledge of the spammer's identity key.

5 Performance Evaluation

5.1 Computation overhead

17/WAKU2-RLNRELAY utilizes the RLN library [5] for identity key generation and commitment, Shamir secret sharing, zkSNARKs circuit representation, proof generation, and verification. The underlying Elliptic Curve is BN254 and the instantiated hash function is Poseidon with the security level of 128 bits. According to the benchmarking report [15], for a group size of 2^{32} (Merkle tree depth of 32) the proof generation on an iPhone 8 takes almost 0.5 seconds. This is sufficiently fast for many messaging applications, but may not be low enough for e.g. real-time communications.

5.2 Storage overhead

A peer has to persist its identity key as well as the prover and verification keys. The former is of size 32 bytes and the latter is ≈ 3.89 MB [15]. Furthermore, for a group of size N, each peer has to dedicate $2N - 1.H$ (H is the hash size (Poseidon hash function)) storage space to maintain the Merkle tree. Let $N = 2^{32}$ and Hash output be $32 = 2^5$ bytes. Under this

setting, the size of the Merkle tree would be $(2 * 2^{32} - 1) * 32 \approx 2^{38}B \approx 2^8$ GB. For smaller size groups e.g., $N = 128$ the tree size drops to almost 8 Kilobytes. In section 6.1, we will discuss potential solutions that can reduce the storage requirement.

6 Conclusion, Future Work and Open Problems

17/WAKU2-RLNRELAY is a routing protocol that features privacy-preserving economic spam protection through rate-limiting nullifiers. The idea is to financially discourage peers from publishing more than one message per epoch. In specific, exceeding the messaging rate results in a financial charge. Those who violate this rule are called spammers and their messages are spam. The identification of spammers does not rely on any central entity. Also, the financial punishment of spammers is cryptographically guaranteed. In this solution, privacy is guaranteed since 1) Peers do not have to disclose any piece of personally identifiable information in any phase i.e., neither in the registration nor in the messaging phase 2) Peers can prove that they have not exceeded the messaging rate in a zero-knowledge manner and without leaving any trace to their identity public keys. Furthermore, it imposes light computational overhead to the routing peers which makes it suitable for resource-limited devices. The proof generation time, as the most recurring and expensive operation in this design, is almost half a second [15] for a group of $2^{32} \approx 4$ billion peers.

6.1 Future Work

17/WAKU2-RLNRELAY is currently a Proof of Concept (POC), and its development is in progress. The following are some of the future work that we would like to peruse further.

Evaluating Merkle tree computation overhead: We would like to evaluate the running time associated with the Merkle tree operations. Indeed, the need to locally store Merkle tree on each peer was one of the unknowns discovered during this POC and yet the concrete benchmarking result in this regard is not available.

Enhancing performance by off-chain solutions: Another possible improvement is to replace the membership contract with a distributed group management scheme e.g., through distributed hash tables. This is to address possible performance issues that the interaction with the public Ethereum blockchain may cause. For example, the registration transactions are subject to delay as they have to be mined before being visible in the state of the membership contract. This means peers have to wait for some time before being able to publish any message. The same issue exists for slashing and other smart contract related functions. The use of state-channels and optimistic Rollups are other ways to overcome transaction delays and achieve better performance.

Lowering the storage overhead per peer: Currently, peers are supposed to maintain the entire tree locally and it imposes storage overhead which is linear in the size of the group [12]. One way to cope with this is to use the light-node and full-node paradigm in which only a subset of peers who are more resourceful retain the tree whereas the light nodes obtain the necessary information by interacting with the full nodes. Another way to approach this problem is through a more storage efficient method [9] where peers store a partial view of the tree instead of the entire tree. Despite having a partial view, peers are able to construct and update the tree root and their authentication based on the dynamic state of the group. Keeping the partial view lowers the storage complexity to $O(log(N))$ where N is the size of the group. The use of Verkle tree [25]/ polynomial commitments [3] is also another path to follow. As our future work, we would like to investigate such solutions.

Cost-effective way of member insertion and deletion: Depending on Ethereum gas costs, the cost associated with 17/WAKU2-RLNRELAY membership can be more than 30 USD [11]. We aim at finding a more cost-effective approach. For example, using batch insertion and deletion can lower this cost to almost 15 USD. Another potential solution is to leverage layer two solutions or to migrate to alternative cost-efficient blockchains with support for smart contracts.

Evaluating user experience and epoch value: The usability of a messaging protocol is heavily influenced by the messaging speed. In the case of 17/WAKU2-RLNRELAY, this is impacted by the epoch value. While we recommend the epoch value to be 1 second $\pm 20s$ where $20s$ is the approximated network delay, an empirical analysis is required to measure network delay as well as to assess the impact of this recommended epoch value on the messaging speed and user experience.

6.2 Open Problems

Below is the list of open problems for which no immediate solution is known hence demand more long-term research.

Exceeding the messaging rate via multiple registrations: While the economic-incentive solution has an economic incentive to discourage spamming, we should note that there is still expensive attack(s) [13] that a spammer can launch to break the messaging rate limit. That is, the attacker can pay for multiple legit registrations e.g., k, hence being able to publish k messages per epoch. We believe that the higher the membership fee is, the less probable would be such an attack, hence a stronger level of spam protection can be achieved. Following this argument, the high fee associated with the membership, which is discussed in section 6.1, can indeed be beneficial for spam prevention.

Escaping punishment by early withdrawal: A spammer can escape from getting slashed by withdrawing its fund from the contract before its spam activity gets caught. While this means the attacker burns its initial membership fund (the fee paid to register its key to the group), it allows saving the other part of the fund that can be otherwise taken by other peers for slashing.

References

1 https://docs.libp2p.io/concepts/publish-subscribe/.
2 https://eips.ethereum.org/eips/eip-627.
3 https://ethresear.ch/t/open-problem-ideal-vector-commitment/7421/27.
4 https://ethresear.ch/t/semaphore-rln-rate-limiting-nullifier-for-spam-prevention-in-anonymous-p2p-setting/5009.
5 https://github.com/kilic/rln.
6 https://github.com/kobigurk/phase2-bn254/tree/master/powersoftau.
7 https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.1.mdpeer-scoring.
8 https://github.com/status-im/nim-waku.
9 https://github.com/vacp2p/research/blob/master/rln-research/merkle-tree-update.md.
10 https://github.com/vacp2p/research/issues/44.
11 https://github.com/vacp2p/research/issues/56.
12 https://github.com/vacp2p/research/issues/57.
13 https://github.com/vacp2p/rfc/issues/251.
14 https://github.com/weijiekoh/perpetualpowersoftau.
15 https://hackmd.io/tmtlmymtr5eynw2lwk9n1w?

16 https://medium.com/coinmonks/announcing-the-perpetual-powers-of-tau-ceremony-to-benefit-all-zk-snark-projects-c3da86af8377.
17 https://rfc.vac.dev/.
18 https://rfc.vac.dev/spec/10/.
19 https://rfc.vac.dev/spec/11/.
20 https://rfc.vac.dev/spec/12/.
21 https://rfc.vac.dev/spec/13/.
22 https://rfc.vac.dev/spec/17/.
23 https://semaphore.appliedzkp.org/.
24 https://status.im/.
25 https://vitalik.ca/general/2021/06/18/verkle.html.
26 https://www.oberlo.ca/blog/whatsapp-statistics.
27 Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In *Annual international cryptology conference*, pages 139–147. Springer, 1992.
28 Jens Groth. On the size of pairing-based non-interactive arguments. In *Annual international conference on the theory and applications of cryptographic techniques*, pages 305–326. Springer, 2016.
29 Kobi Gurkan, Koh Wei Jie, and Barry Whitehat. Community proposal: Semaphore: Zero-knowledge signaling on ethereum. 2020. URL: `https://github.com/appliedzkp/semaphore/blob/master/spec/\Semaphore%20Spec.pdf`.
30 Ian Miers. Scalable multi-party computation for zk-snark parameters in the random beacon model. 2019.
31 Adi Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.
32 Nik Unger, Sergej Dechand, Joseph Bonneau, Sascha Fahl, Henning Perl, Ian Goldberg, and Matthew Smith. Sok: secure messaging. In *2015 IEEE Symposium on Security and Privacy*, pages 232–249. IEEE, 2015.