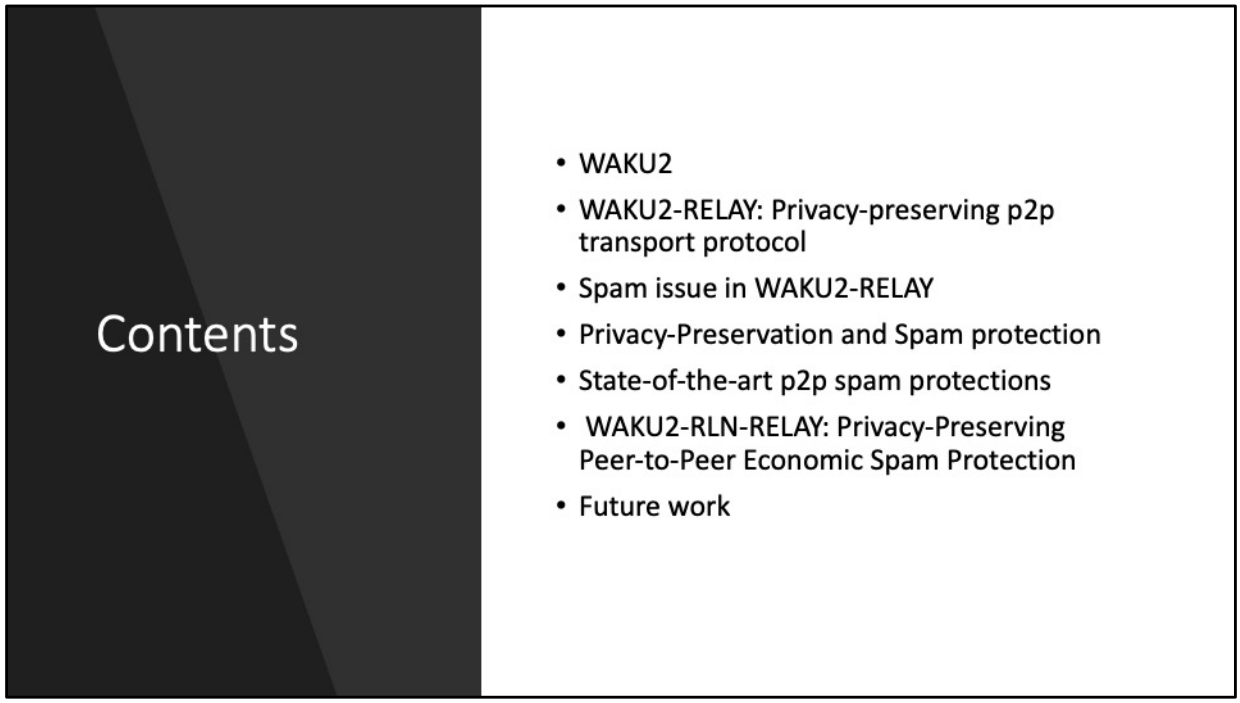
Hi,
Today, I am going to present waku-rln-relay which is a privacy preserving economic spam protection that suits private and anonymous messaging systems
This is a joint work by me, Oskar Thoren, Barry Whitehat, Wei Jie Koh, Onur Kilic, and Kobi Gurkan
The paper is also available in the link below



The agenda is as illustrated
I'll start by discussing WAKU2 and WAKU-RELAY, I will review the spam issue and the related studies, then I'll explain waku-rln-relay architecture and how it excels its counterparts. In the end, I will shed light on the future work.

WAKU2 [1]

- A family of modular, privacy-preserving peer-to-peer (p2p) protocols for private, secure, censorship resistant communication
- Suitable for resource restricted devices e.g., mobile phones
- WAKU2 protocols include:
  - **WAKU2-RELAY: privacy-preserving transport**
  - WAKU2-STORE: historical message storage
  - WAKU2-FILTER: light version of WAKU2-RELAY for bandwidth limited devices
  - **WAKU2-RLN-RELAY: spam-protected version of WAKU2-RELAY**
  - And many more …
- For the full list of RFCs is available in rfc.vac.dev

[1] https://rfc.vac.dev/spec/10/

Waku2 is a stack of peer-to-peer (p2p) protocols that enable anonymous and privacy-preserving communication.
It is designed to be able to run in resource-restricted environments.

It contains multiple protocols as you see, but
The focus of today's talk are WAKU-RELAY and WAKU-RLN-RELAY which are the transport layers of Waku

WAKU2-RELAY [1]

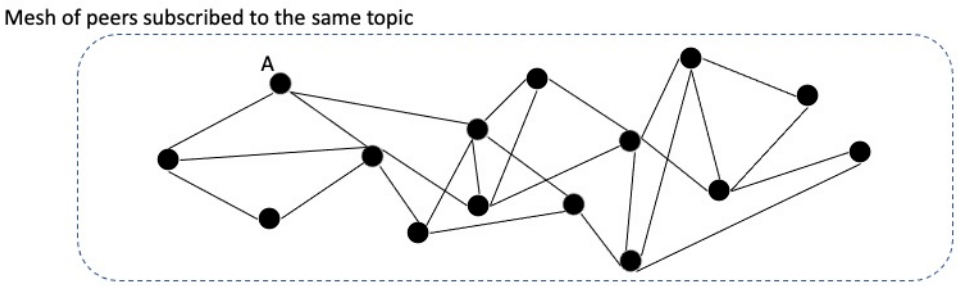
- Publisher-Subscriber Model
- Gossip-based Routing (extension of libp2p GossipSub-v1.1 [2])
- Anonymous and Privacy-Preserving

[1] https://rfc.vac.dev/spec/11/
[2] https://github.com/libp2p/specs/tree/master/pubsub/gossipsub

Let me start by WAKU2-RELAY transport protocol
It follows publisher-subscriber messaging model
Implements gossip-based routing protocol
Is a minor extension of libp2p GossipSub protocol
Its end goal of this protocol is to provide an anonymous and privacy preserving p2p transport layer

WAKU2-RELAY

- Peers subscribed to the same topic form a mesh

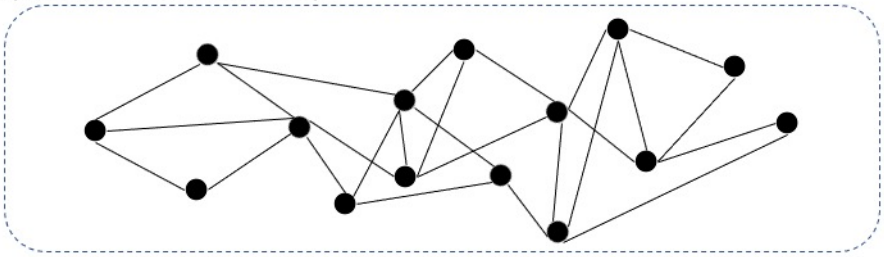


Peers in wakurelay congregate around topics they are interested in and can send messages to that topic or receive messages within that topic
here is the illustration of a pubsub mesh of peers subscribed to the same topic

WAKU2-RELAY [1]

- Publisher-Subscriber Model
  - Peers subscribed to the same topic form a mesh

Mesh of peers subscribed to the same topic

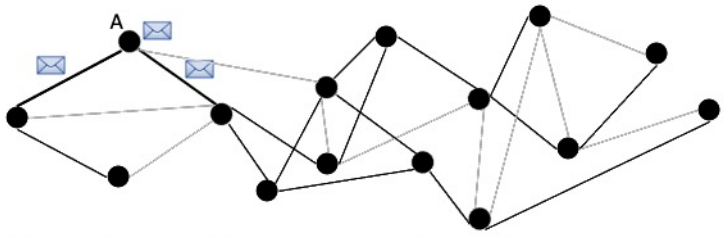


[1] https://rfc.vac.dev/spec/11/

Let me start by WAKU2-RELAY transport protocol
It follows publisher-subscriber messaging model that is
congregate around topics they are interested in and can send messages to that topic or receive messages within that topic
here is the illustration of a pubsub mesh of peers subscribed to the same topic

WAKU2-RELAY

- Publisher-Subscriber Model
- Gossip-based Routing (extension of libp2p GossipSub-v1.1 [2])



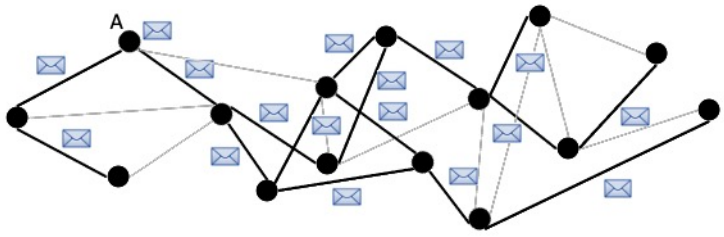
[2] https://github.com/libp2p/specs/tree/master/pubsub/gossipsub

It Implements gossip-based routing protocol
Is a minor extension of libp2p GossipSub protocol
Peers route messages by sending them to a subset of their connections

In this example, A as the message owner, forwards its message to a subset of neighbors.

WAKU2-RELAY

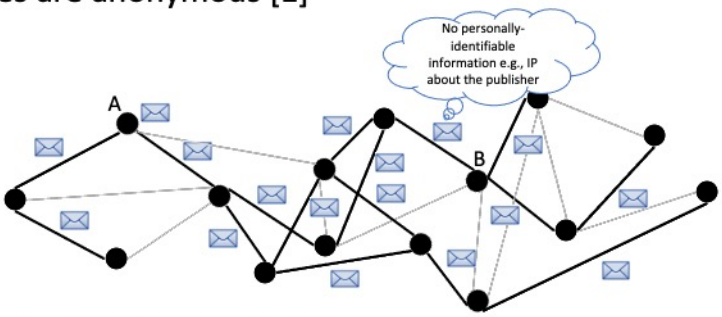
- Publisher-Subscriber Model
- Gossip-based Routing (extension of libp2p GossipSub-v1.1 [2])



The neighbors proceed similarly till the message reaches the entire mesh.

WAKU2-RELAY

- Publisher-Subscriber Model
- Gossip-based Routing (extension of libp2p GossipSub-v1.1 [2])
- Messages are anonymous [1]

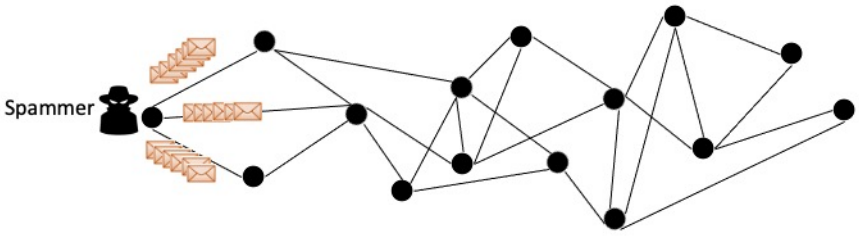


[1] https://rfc.vac.dev/spec/11/#security-analysis

Waku-relay is also an anonymous transport where there is no personally identifiable information like IP address attached to the protocol messages, therefore it is not feasible to identify the origin of a message.

Spam issue in WAKU2-RELAY

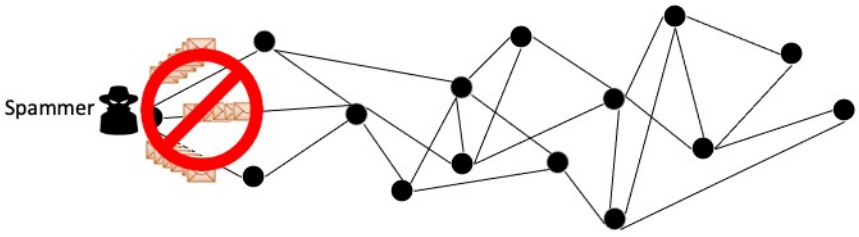
- We define spammers as entities that publish a large number of messages in a short amount of time, and cause denial-of-service



Waku Relay, as an open p2p transport protocol can be exploited by spammers; we define spammers as entities that publish a large number of messages in a short amount of time, and cause Denial of Service attack.

Spam issue in WAKU2-RELAY

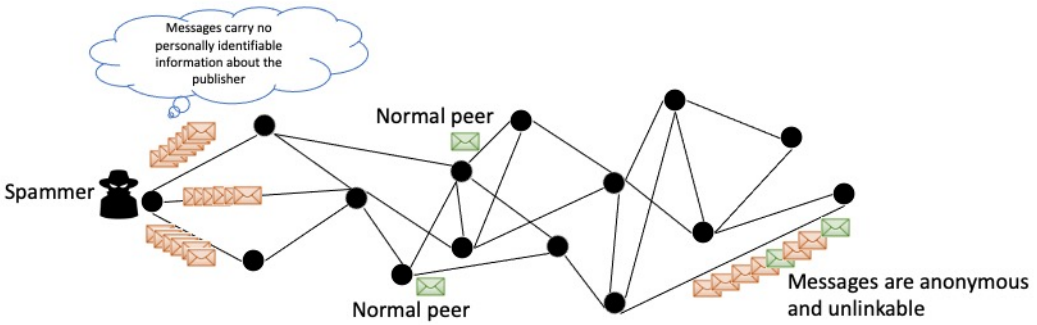
- We define spammers as entities that publish a large number of messages in a short amount of time, and cause denial-of-service
- Spam Protection = Controlled Messaging Rate



With that definition, Spammers can be controlled if we can control their messaging rate

Privacy-Preservation and Spam protection

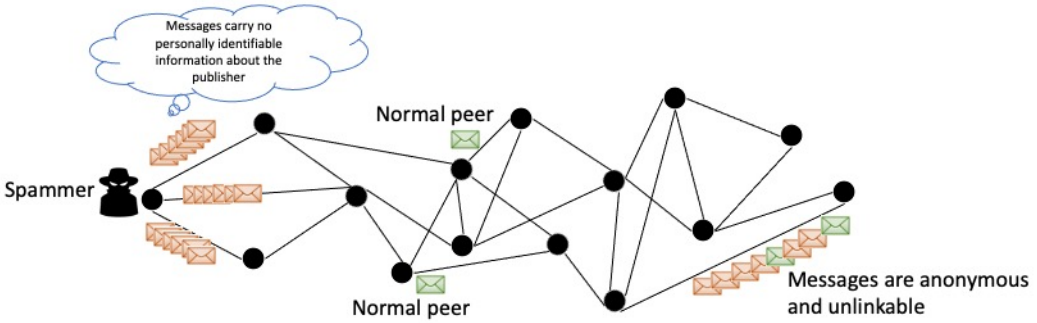
- Messages are anonymous: No Personally Identifiable information is available



But there is a big issue that is, messages are anonymous
Therefore routing peers just observe a surge of messages coming in without knowing who has published them
Thus spam messages are indistinguishable from non-spam

Privacy-Preservation and Spam protection

- Messages are anonymous: No Personally Identifiable information is available
- Solutions like IP blocking are not effective



as such solutions like IP blocking are not effective

State-of-the-art p2p spam protections

- Proof-of-work [1] deployed by Whisper [2]
  - Computationally expensive
  - Not suitable for network of heterogeneous peers with limited resources
- Peer Scoring [3] in libp2p
  - Local to each peer
  - No global identification of spammer
  - Subject to inexpensive attacks using bots
  - Prone to censorship

[1] Cynthia Dwork and Moni Naor. Pricing via processing or combatting junk mail. In Annual 456 international cryptology conference. Springer, 1992.
[2] https://eips.ethereum.org/eips/eip-627.
[3] https://github.com/libp2p/specs/blob/master/pubsub/gossipsub/gossipsub-v1.1.mdpeerscoring.

The state-of-the-art p2p spam protection techniques are Proof of Work (POW) deployed by Whisper
and Peer scoring method adopted by libp2p

The PoW is not computationally efficient and does not fit resource limited devices (limited resources won't be able to participate and benefit from the messaging system)

On the other side, peer scoring is a local solution since each peer monitors and scores its direct connections and drops the connections with low scores.
However, a spammer would be still able to continue its activity by switching its connection from one peer to another as soon its score drops a threshold.
Furthermore, there are inexpensive attacks where the spammer can deploy millions of bots to send bulk messages.
~It is also prone to censorship~

WAKU2-RLN-RELAY [1]

WAKU2-RLN-RELAY = WAKU2-RELAY + Rate Limiting Nullifiers (RLN)

- P2p solution
- Global spam protection
- Privacy preserving
- Efficient
- Economic incentives
  - Financial punishment for the spammers and a financial reward for those who catch spammers.

[1] https://rfc.vac.dev/spec/17/

The good news is that in WAKU-RLN-RELAY we cope with the aforementioned issues
We take the waku-relay protocol as an anonymous transport protocol and combine it with the rate limiting nullifier construct to control the messaging rate
The end result has a p2p structure, with no central entity involved
it **allows global identificatio**n and removal of spammers
it is **privacy-preserving, user anonymity is preserved** since there is no need to personally identifiable information e.g., email address, IP, etc. about peers to be able to identify and block spammers
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.
It has **economic-incentives, i.e.,** there is a financial punishment for the spammers and a financial reward for those who catch spammers.

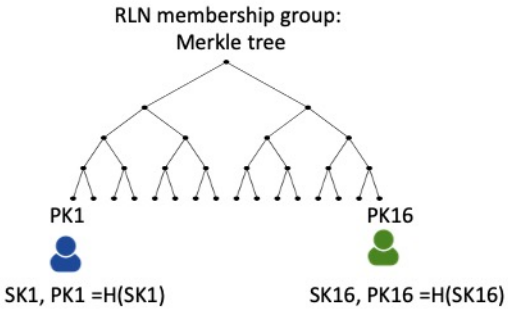
RLN Primitive [1]

- RLN is a zero-knowledge and rate-limited signaling framework
- Each user can only send M messages for each External Nullifier
- External nullifier can be seen as a voting booth where each user can only cast one vote
- M and external nullifier are application dependent
- M=1 for this presentation

[1] https://ethresear.ch/t/semaphore-rln-rate-limiting-nullifier-for-spam-prevention-in-anonymous-p2p-setting/5009

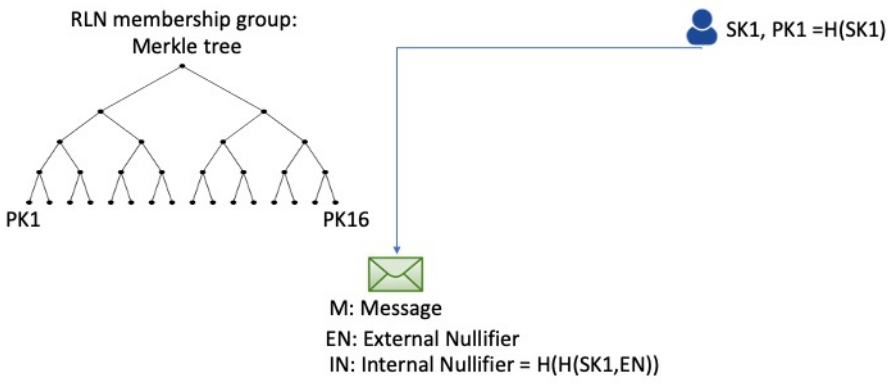
- Lets begin with the RLN construct
- it is a zero-knowledge and rate-limited signaling framework
- It allows a set of users to broadcast arbitrary signals (where signal is any value like a string, vote, etc.) while proving they are among a group of authorized users without disclosing their identities
- The idea is that each user can only send M messages for a specific external nullifier. External nullifier can be seen as a voting booth where each user can only cast one vote
- For the rest of this presentation we consider the messaging rate to be 1

RLN Primitive: Membership Tree



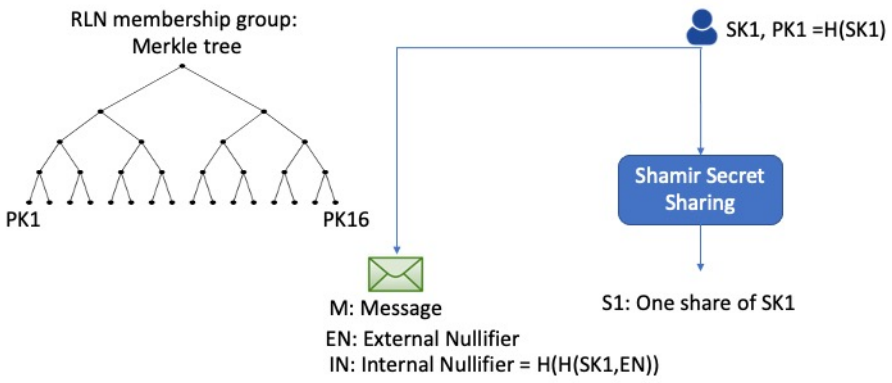
- RLN primitive consists of a merkle tree that represents a group of authorized users,
- each user has a pk registered in this tree,
- the corresponding SK is only known to the user

RLN Primitive: Signaling

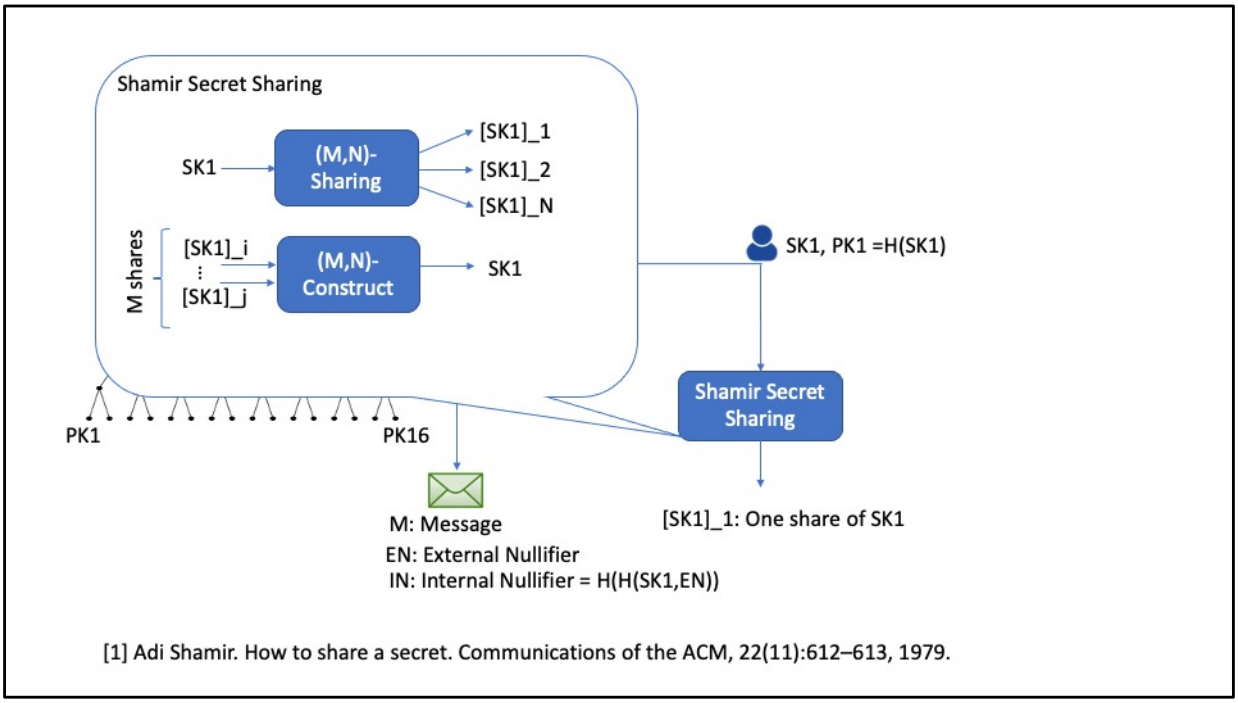


- For the signaling/publishing:
- The user specifies an external nullifier
- as well as an internal nullifier which is derived from the SK and the external nullifier (as you can see in the slide)

RLN Primitive: Signaling

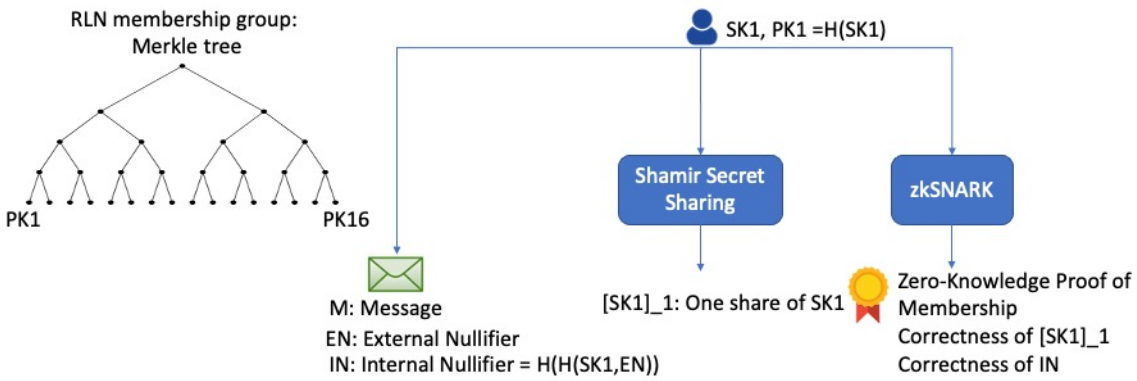


- The user also discloses a share of its secret key using shamir secret sharing scheme
- This share will be used to remove the user from the group in case of double signaling

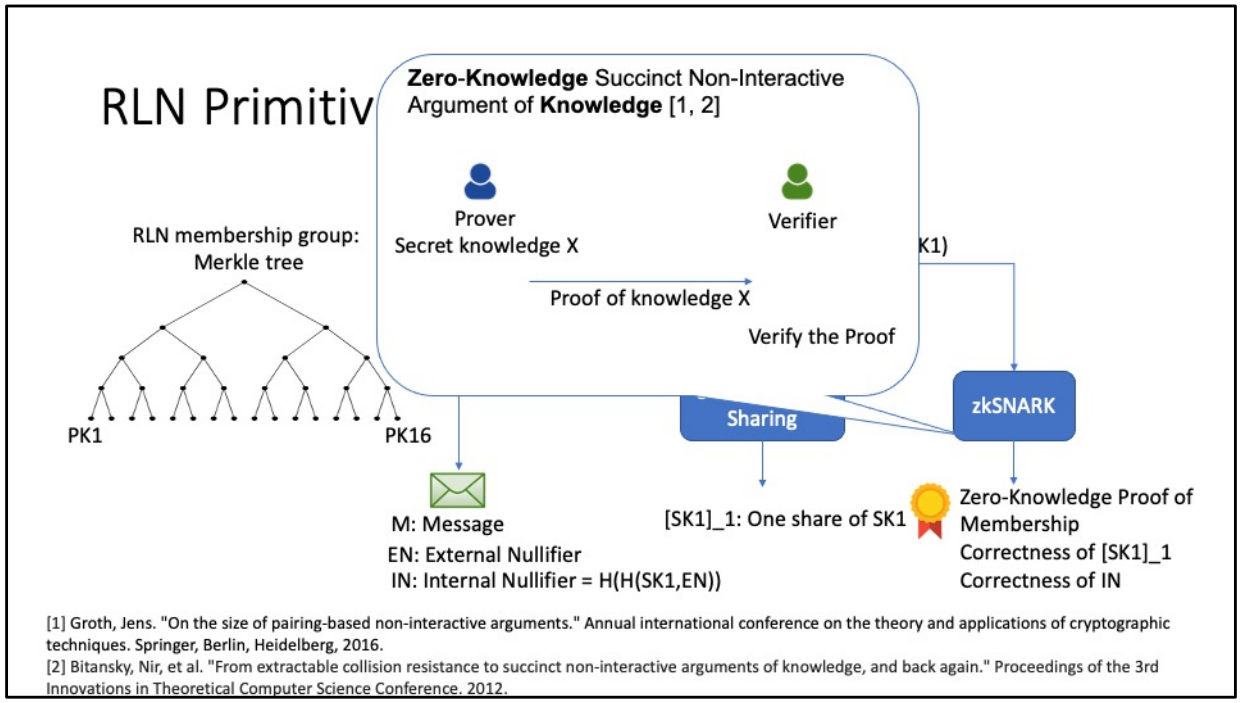


For those who dont know about Shamir secret sharing, the high level idea is that it is a technique that allows to split a secret data sk into N pieces
It is possible construct the sk back by having a subset of M shares

RLN Primitive: Signaling



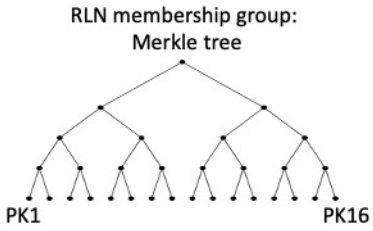
- Finally, the user proves in zero knowledge manner that:
- It is part of the group
- And that has computed the secret share and the Internal nullifier correctly

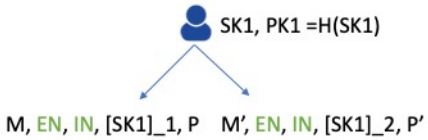


[1] Groth, Jens. "On the size of pairing-based non-interactive arguments." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2016.
[2] Bitansky, Nir, et al. "From extractable collision resistance to succinct non-interactive arguments of knowledge, and back again." Proceedings of the 3rd Innovations in Theoretical Computer Science Conference. 2012.

- For those not familiar with zk-SNARK, it stands for “Zero-Knowledge Succinct Non-Interactive Argument of Knowledge.”
- A zk-SNARK is a cryptographic proof that allows one party to prove the possession of certain information to a verifier without revealing that information.

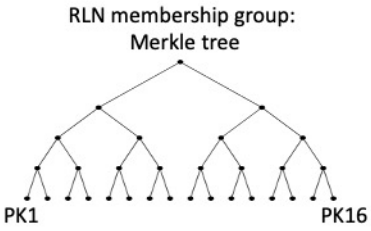
RLN Primitive: Detecting double signaling

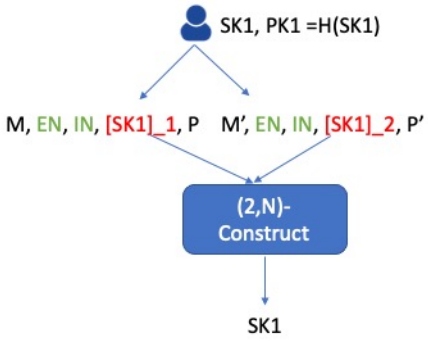




- Users cannot violate the messaging rate, why? Because If a user publishes more than 1 message for the same external nullifier, it will end up having two messages with the same external and internal nullifiers (remember those are deterministic values computed from SK and external nullifier)
- So the double signaling attempt can be detected

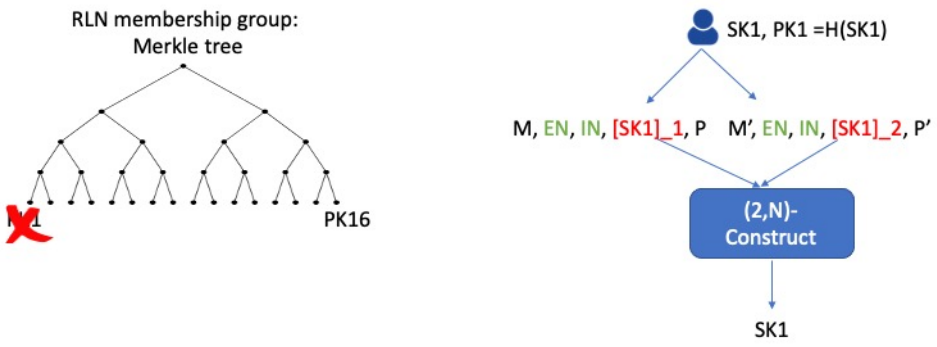
RLN Primitive: Slashing





- Moreover, by sending two messages, the usre will disclose 2 shares of its SK (one per each message)
- using which the corresponding SK can be reconstructed

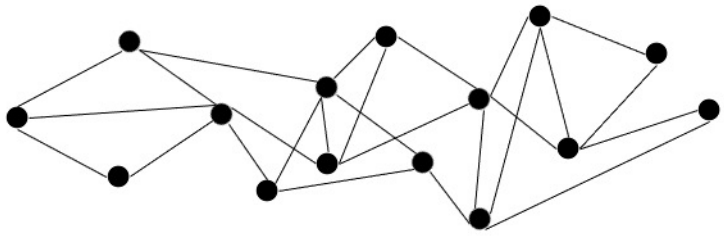
RLN Primitive: Slashing



- And the user gets removed from the tree. Thus, it can no longer use that sk for messaging.

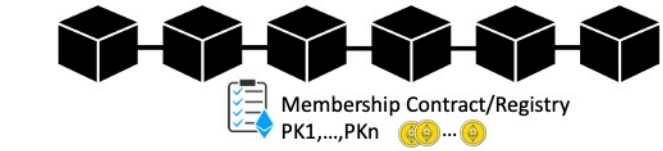
WAKU2-RLN-RELAY

RLN group = Peers subscribed to the same topic e.g., waku-rln-relay

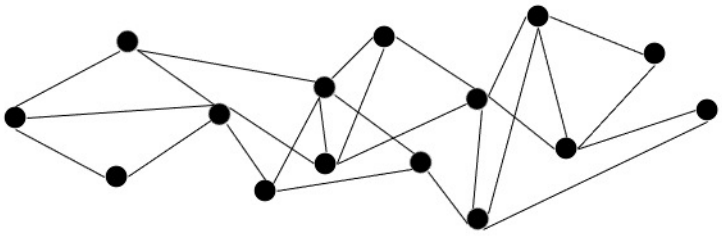


Now lets see the end to end integration of RLN into Waku-rln-relay
Here, the rln group consists of peers that belong to the same GossipSub layer (subscribed to the same topic)

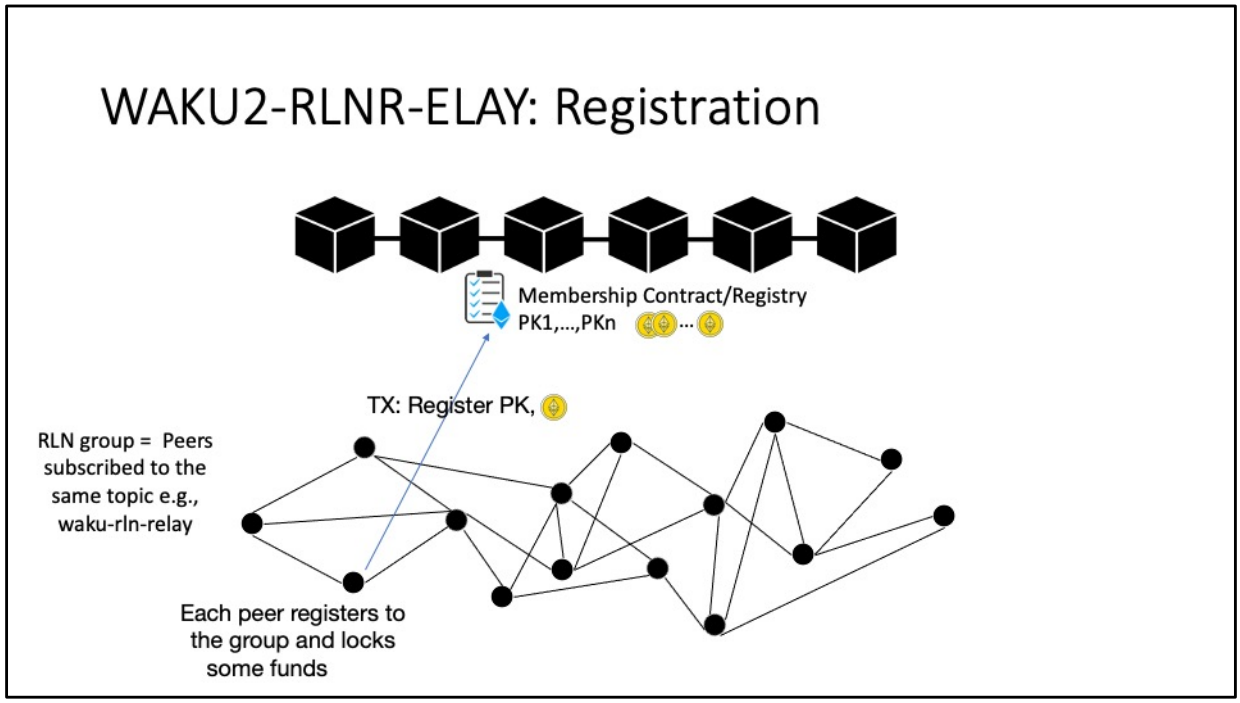
WAKU2-RLN-RELAY: Registration



RLN group = Peers subscribed to the same topic e.g., waku-rln-relay

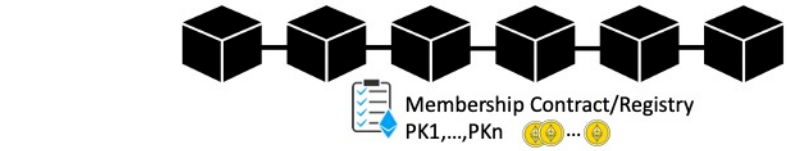


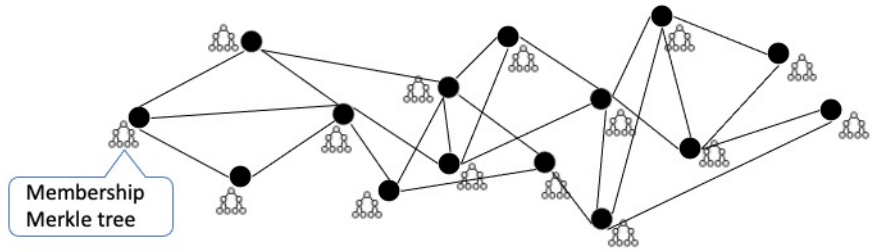
Each peer has a rln pk, and the list of pks is stored in a contract deployed on the Eth blockchain.



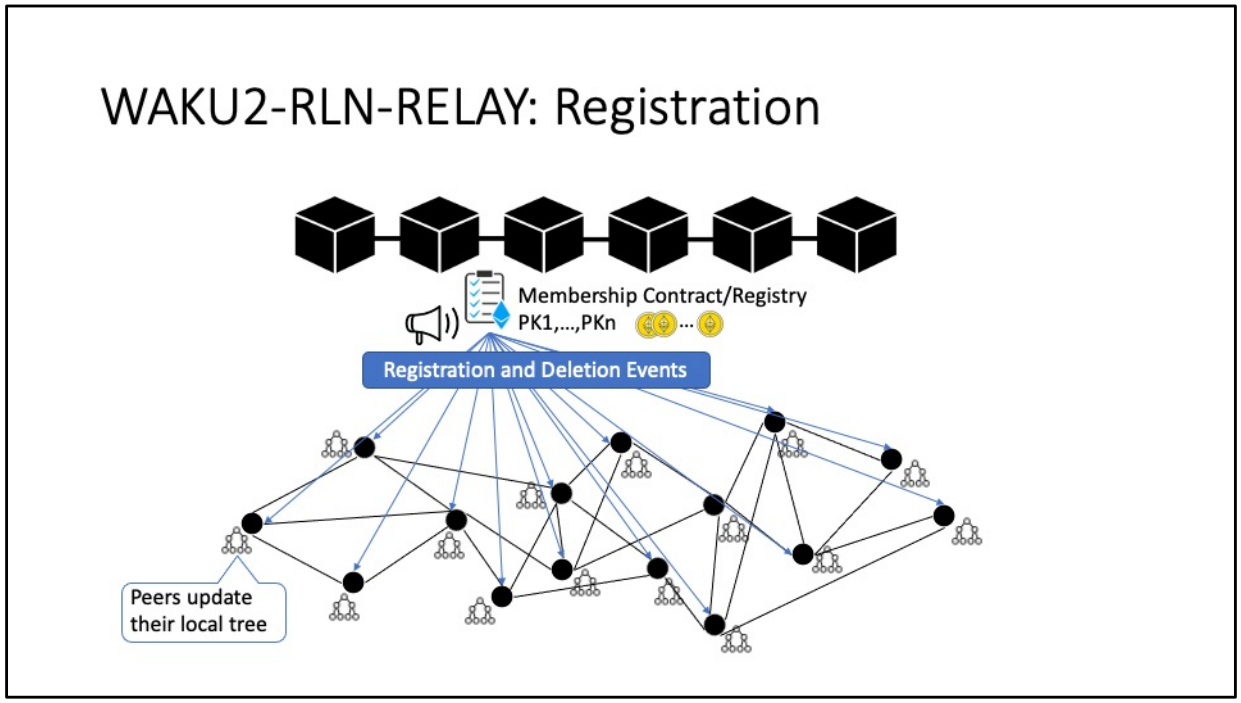
A peer willing to publish a message should register by sending a tx to the contract that contains its rln pk and some amount of Ether. This amount is deposited on the contract to prevent spam activity. Registration is a one time action.

WAKU2-RLN-RELAY: Registration





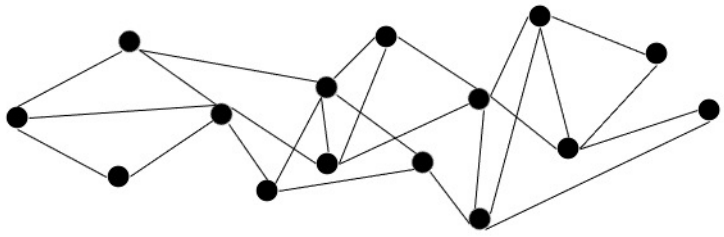
Peers construct the rln membership Merkle tree locally



And they listen to the registration and deletion events emitted from the contract in order to update their trees.

WAKU2-RLN-RELAY: External Nullifier

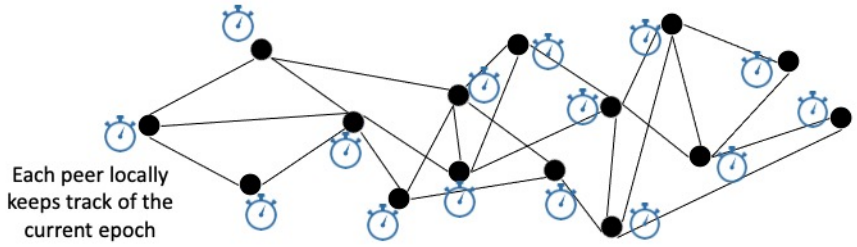
External Nullifier = Epoch = the number of T seconds that elapsed since the Unix epoch.
Messaging rate is limited to 1 per epoch.



For the external nullifier we denote it by Epoch which is the number of T seconds (where T is a system design parameter) that elapsed since the Unix epoch.
Peers are allowed to publish one message per epoch without being slashed

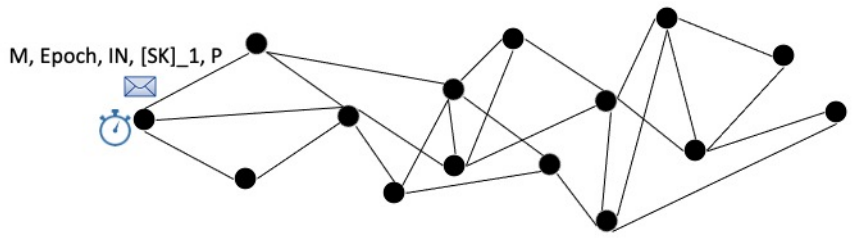
WAKU2-RLN-RELAY: External Nullifier

External Nullifier = Epoch = the number of T seconds that elapsed since the Unix epoch.
Messaging rate is limited to 1 per epoch.



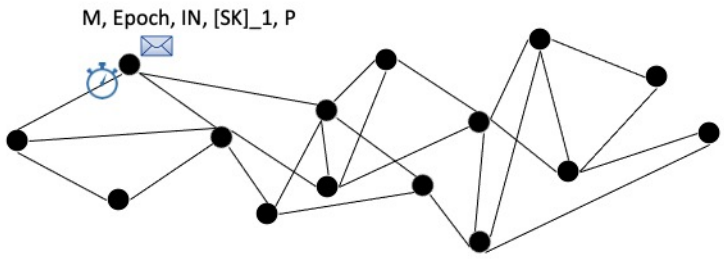
Each peer locally keeps track of the current epoch.

WAKU2-RLN-RELAY: Publishing

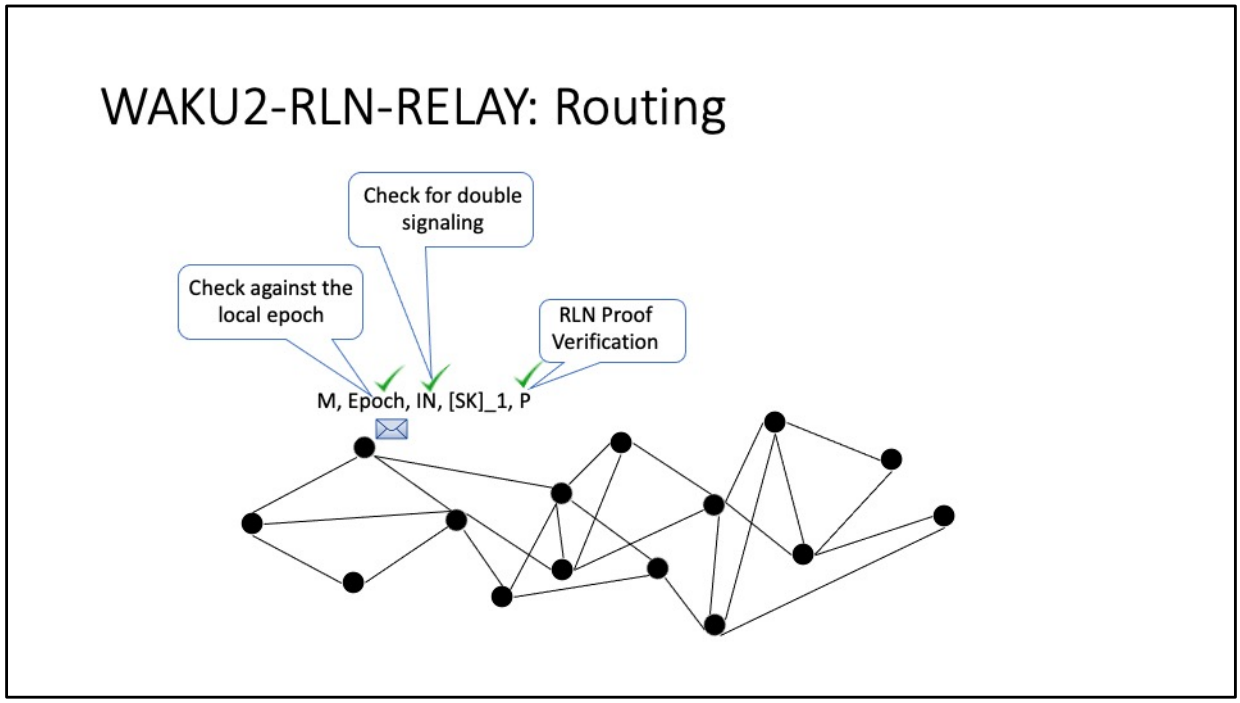


Message publishing in the network is the same as RLN framework
The message owner, attaches the nullifiers, together with its sk share, and the zero knowledge proof part to the message

WAKU2-RLN-RELAY: Routing



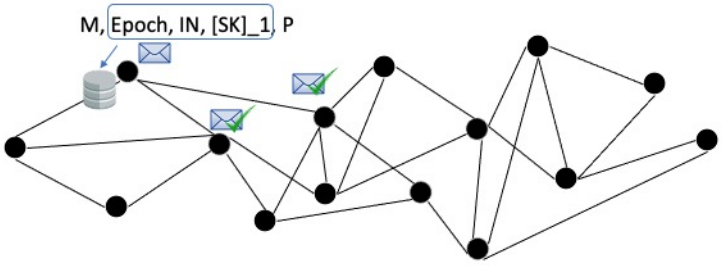
A routing peer follows the regular routing protocol of wakurelay (gossipSub protocol) and in addition
does the verification steps of RLN construct



It verifies the proof
Also validates the Epoch of the incoming message against its local Epoch to see if there is a huge gap or not

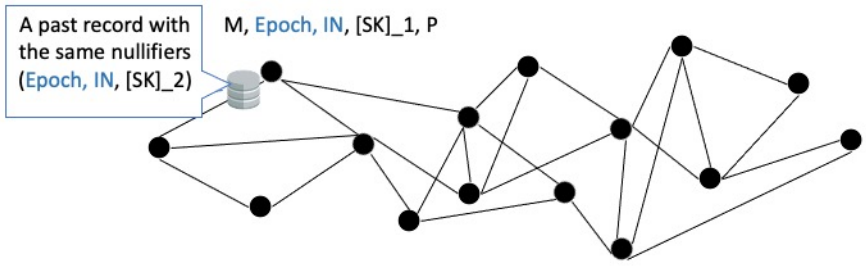
and checks the nullifiers to see if double signaling has happened
If all the checks pass, relays the message

WAKU2-RLN-RELAY: Routing



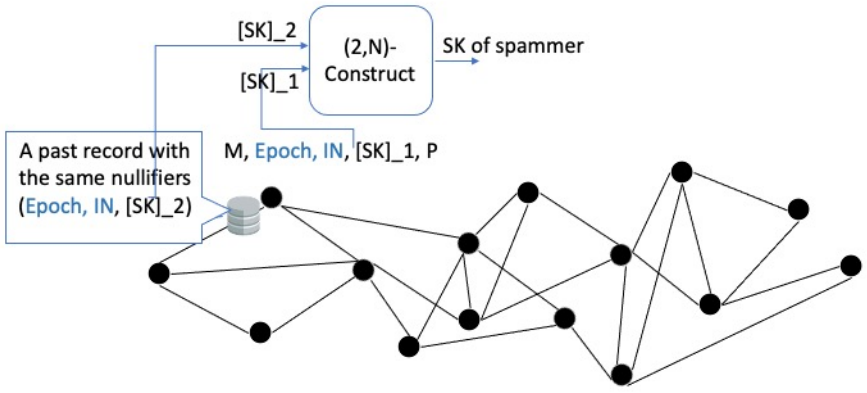
keeps record of of the nullifiers of the messages, it is needed to catch double signaling for the future incoming messages

WAKU2-RLN-RELAY: Slashing



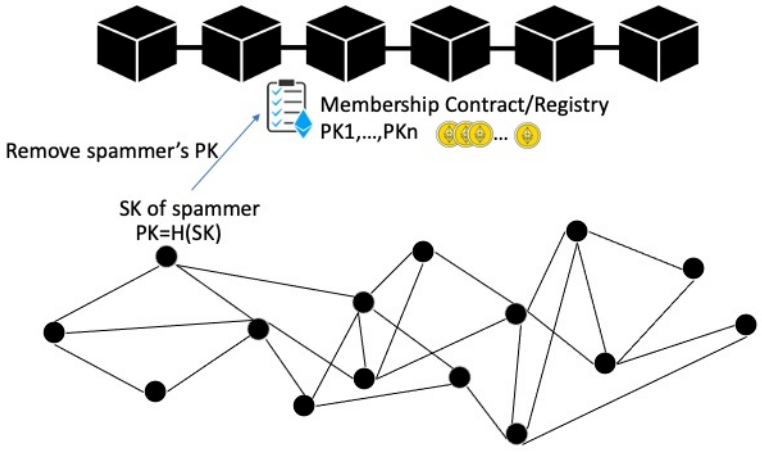
But what if the routing peer finds out that the messaging rate is violated, i.e., there has been an old message whose nullifiers match the new message

WAKU2-RLN-RELAY: Slashing



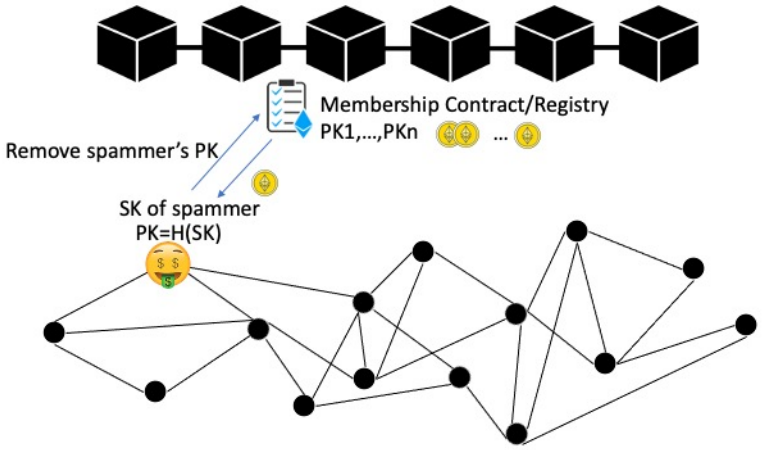
In that case, it reconstructs the sk of the spammer

WAKU2-RLN-RELAY: Slashing



Sends a transaction to the contract and removes the spammer pk from the group

WAKU2-RLN-RELAY: Slashing



and withdraws the spammer's fund

WAKU2-RLN-RELAY [1]

WAKU2-RLN-RELAY = WAKU2-RELAY + Rate Limiting Nullifiers (RLN)

- P2p solution
- Global spam protection
- Privacy preserving
- Efficient
- Economic incentives
  - Financial punishment for the spammers and a financial reward for those who catch spammers.

[1] https://rfc.vac.dev/spec/17/

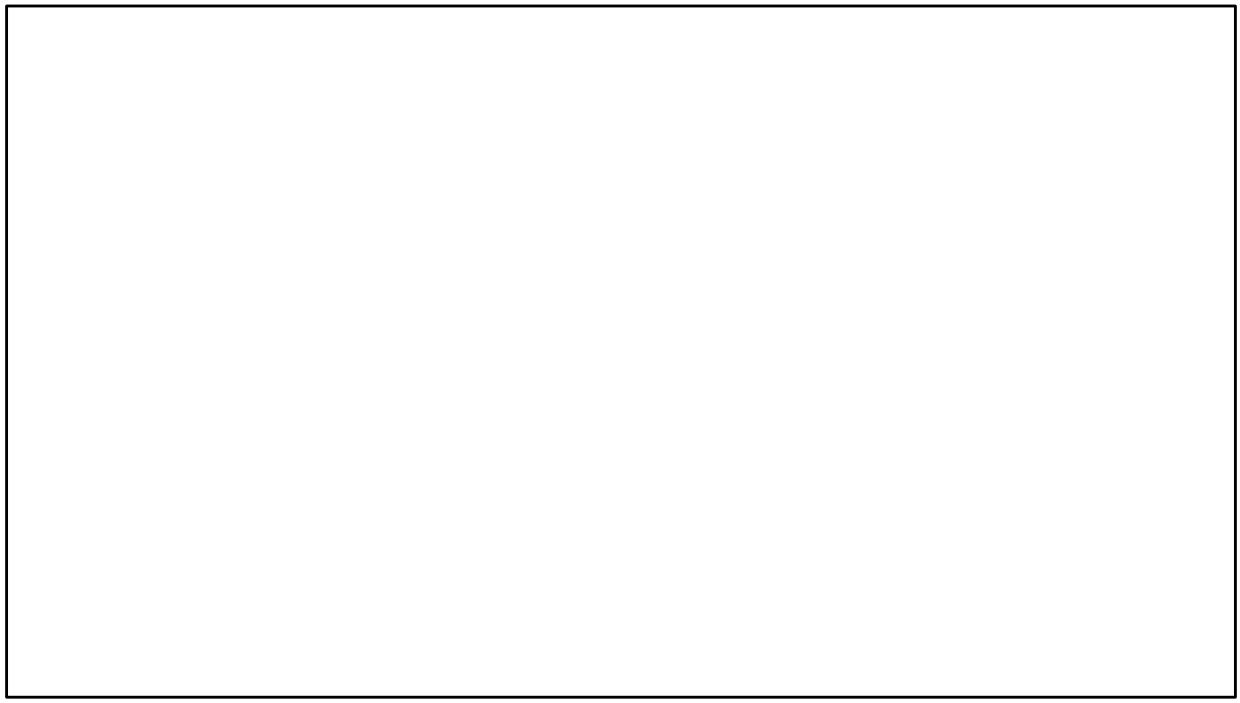
This brings us to the end of the presentation,
We talked about waku-rln-relay, and how the end to end interaction works, and how it enables global spam removal using the rln primitive on top of waku-relay
Also how it brings together anonymity and incentivized spam-resilience in a p2p messaging system

References

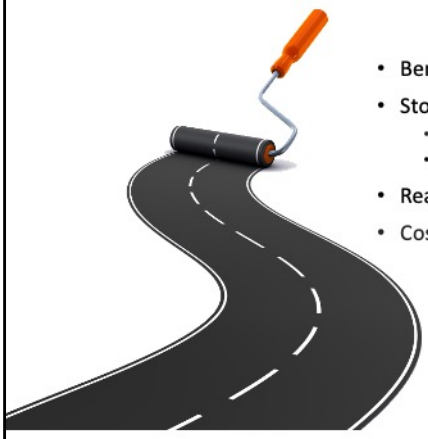
- Waku-rln-relay specs: https://rfc.vac.dev/spec/17/
- Waku-rln-relay paper: https://github.com/vacp2p/research/blob/master/rln-research/Waku_RLN_Relay.pdf
- Vac post on Waku-rln-relay: https://vac.dev/rln-relay
- Nim-Waku implementation: https://github.com/status-im/nim-waku
- js-Waku implementation: https://github.com/status-im/js-waku
- RLN Ethereum research post: https://ethresear.ch/t/semaphore-rln-rate-limiting-nullifier-for-spam-prevention-in-anonymous-p2p-setting/5009
- RLN medium post: https://medium.com/privacy-scaling-explorations/rate-limiting-nullifier-a-spam-protection-mechanism-for-anonymous-environments-bbe4006a57d
- RLN circuits: https://github.com/appliedzkp/rln
- RLN circuits spec: https://hackmd.io/7GR5Vi28Rz2EpEmLK0E0Aw
- RLN in Rust: https://github.com/kilic/rln

References for those interested to read further

Thank you



Future work



- Benchmarking
- Storage-efficient Merkle tree storage
  - P2p network of full-nodes and light-nodes
  - Partial view of Merkle tree
- Real-time removal of spammers using off-chain/p2p solutions
- Cost-effective way of member insertion and deletion using layer 2 solutions

In the end, I would like to wrap up with our future plan
Benchmarking is the first on our future work list
The next is to address storage overhead regarding the maintenance of the full Merkle tree.
Currently, peers maintain the entire tree locally which takes up to 67 MB for tree with depth 20 and almost 274 GB for d=32. This overhead might not fit resource limited devices, so a more optimized solution is desirable.

One solution is to use the light-node and full-node architecture where resource-full nodes retain the entire tree and serve it to the nodes with limited storage.
Another possible solution is to have a partial view of the tree and yet being able to construct and update the tree root and the authentication path when group state changes

We are also looking into an off-chain slashing solution because currently the On-chain slashing is subject to delay (the tx has to be mined), so is the removal of spammers. With an off-chain method, peers can communicate the slashed pks in a p2p manner, hence enjoy a real-time spam-protection

The other direction to pursue is to provide a cost-effective way of member insertion

and deletion using layer 2 solutions. The reason is that currently these operations cost almost 40 k gas, which translates to 15 USD which might be not affordable by the users, so an alternative solution is worth investigation

Asymptotic Performance

- Setting
  - 17/WAKU2-RLNRELAY utilizes the **RLN library** [1] for identity key generation and commitment, Shamir secret sharing, zkSNARK circuits, proof generation, and verification.
  - The underlying **Elliptic Curve** is **BN254** [2].
  - The instantiated hash function is **Poseidon** with the security level of **128 bits** [2].
  - Proof system is **Groth16** [2].
- Computation
  - Proof generation: According to the benchmarking report [3] for a Merkle tree **depth** of **24**, the **proof generation** on an **iPhone 8** takes almost **~0.5 seconds**.
  - User computation per **group update** is **O(d) hashing** operations (where **d=20**) to calculate the tree root and the authentication path.
  - **Bootstrapping** takes **O(2^d) hashing** operations to construct the entire tree.

[1] https://github.com/kilic/rln
[2] https://hackmd.io/tMTLMYmTR5eynw2lwK9n1w?view
[3] Groth, Jens. "On the size of pairing-based non-interactive arguments." Annual international conference on the theory and applications of cryptographic techniques. Springer, Berlin, Heidelberg, 2016.

Use this for more https://hackmd.io/U8nFzfLpQVGu5Zu0dVfJ_Q
0.5 proof generation This is sufficiently fast for many messaging applications, but may not be low enough for e.g. real-time communications.

d is the tree depth which is considered as 20
N is the number of Merkle tree leaves which is 2^d
H is the size of the hash output
A Batch consists of B=128 keys

Asymptotic Performance

- Gas costs*
  - **PK Registration**: The estimated gas cost is **40k**. This is for performing the insertion without locking any ether for the sake of slashing.
  - **PK Slashing**: The estimated gas cost is **40k**.
  - **Batch registration/slashing**: The estimated gas cost is **20k**. A Batch consists of B=128 keys
- Storage
  - The **Merkle tree** with **depth 20** takes up **~67MB** storage.
  - **Identity keys and identity commitment** keys are of size **32 bytes**.
  - **Prover key** size is approximately **~3.24 MB**.

* Derived from https://hackmd.io/JoxnlDq3RT6WhtA-KBxtYg?view

Use this for more https://hackmd.io/U8nFzfLpQVGu5Zu0dVfJ_Q
d is the tree depth which is considered as 20
N is the number of Merkle tree leaves which is 2^d
H is the size of the hash output
A Batch consists of B=128 keys

zkSNARK Setup

- Parameters generation (for Groth16) done in two phases:
  - Phase 1: The powers of tau ceremony
  - Phase 2: MPC for circuit specific parameters