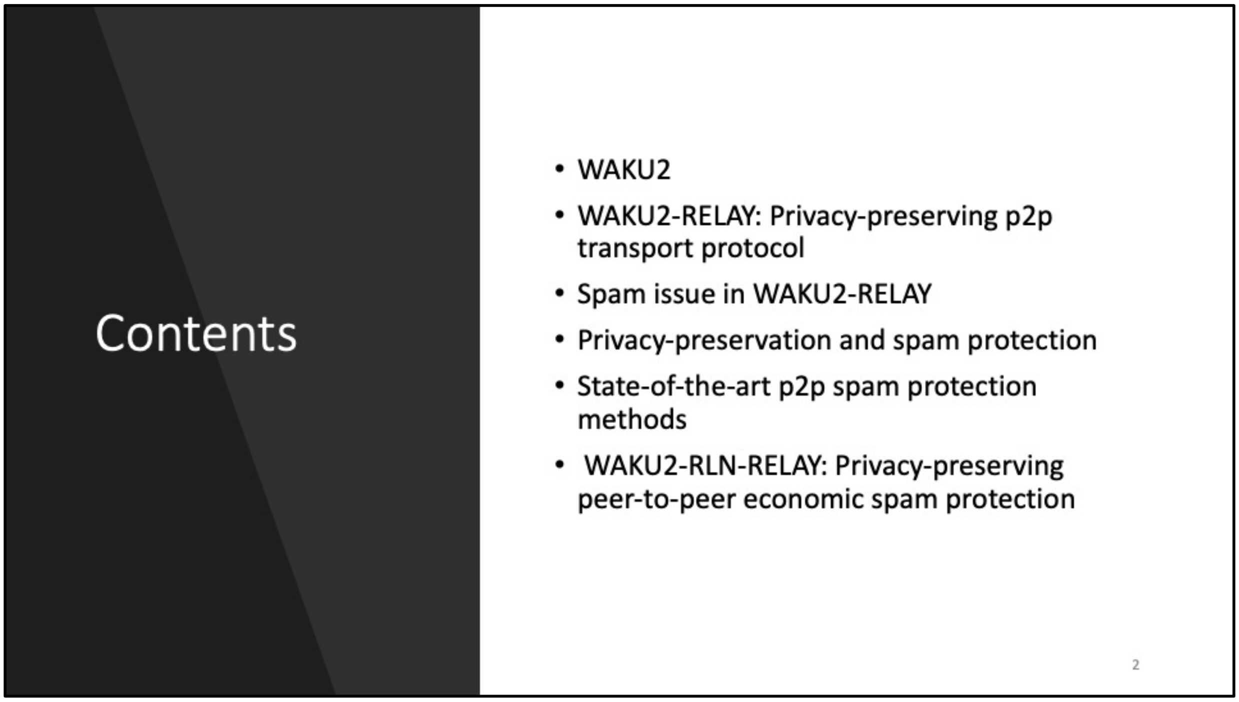
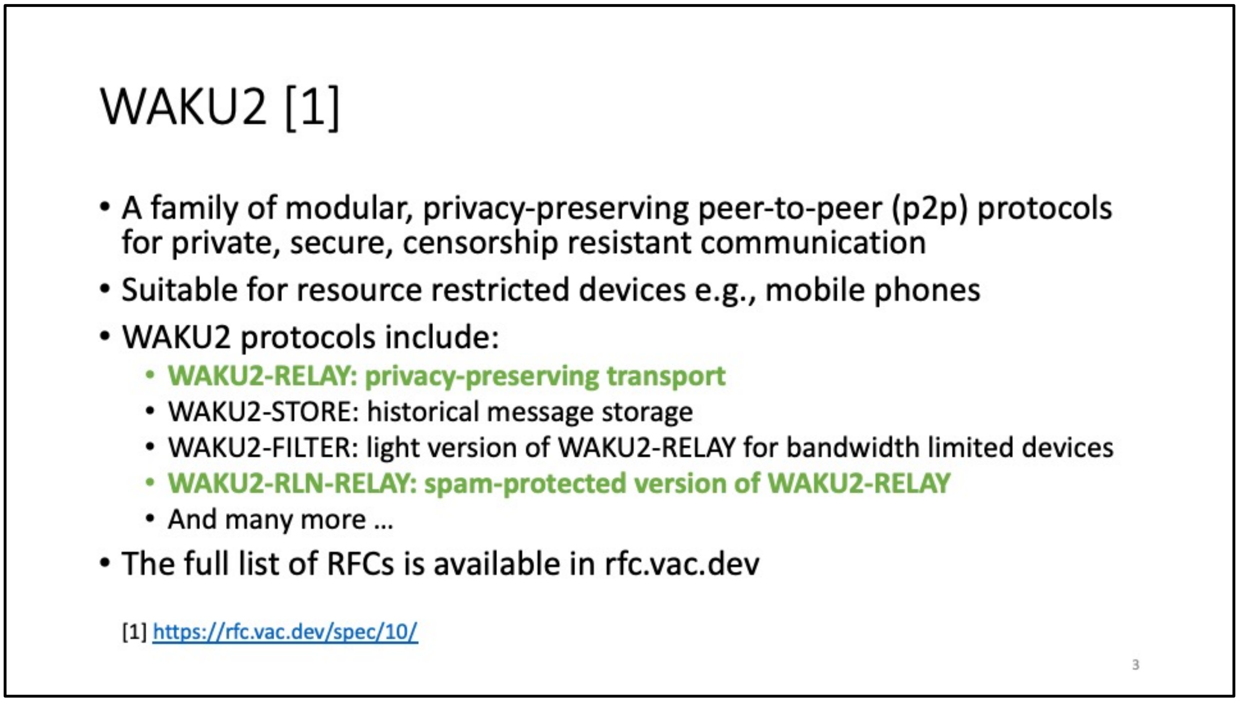
Hi,
Today, I am going to present waku-rln-relay which is a privacy preserving p2p economic spam protection that suits private and anonymous messaging systems
The paper is also available in the link below



I will start the presentation by explaning 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.



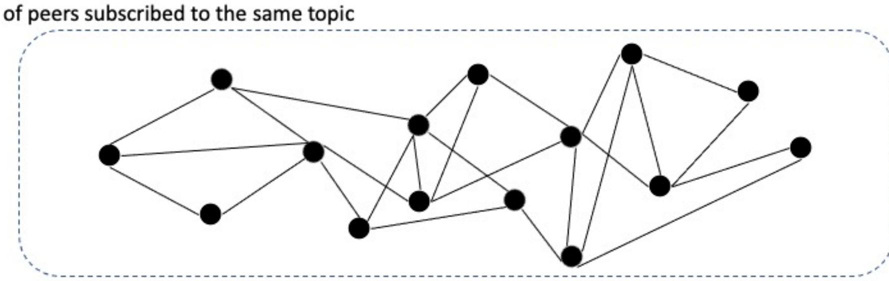
Waku2 is a stack of peer-to-peer (p2p) protocols that enable anonymous and privacy-preserving communication.
its protocols are designed to be able to run in resource-restricted environments.

Waku2 contains multiple protocols, but
The focus of today’s talk would be WAKU-RELAY and WAKU-RLN-RELAY which are the transport layers of Waku

WAKU2-RELAY [1]

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

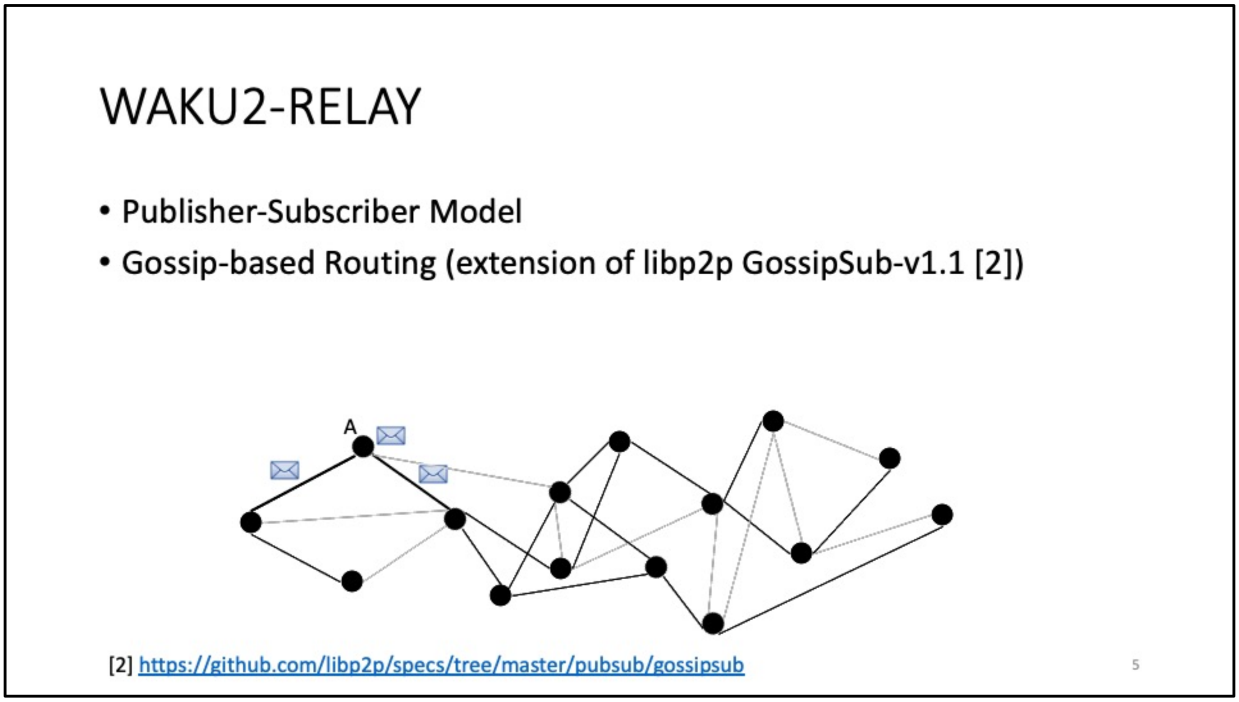
Mesh of peers subscribed to the same topic



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

4

Let me start by WAKU2-RELAY transport protocol
It follows publisher-subscriber messaging model
that is peers gather around topics they are interested in and can send messages to that topic or receive messages within that topic
here is an example illustration of a mesh of peers subscribed to the same topic (note that it does not match reality, it is just an example)

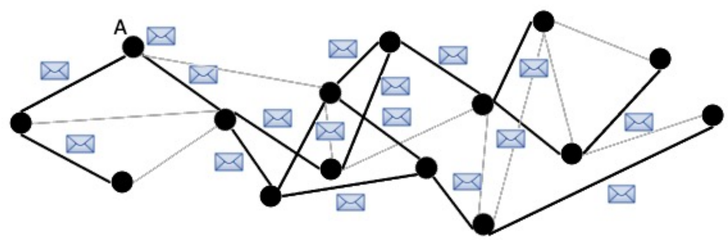


It Implements gossip-based routing protocol
Is a minor extension of libp2p GossipSub protocol
At a high level, Peers route messages by sending them to a subset of their connections

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

WAKU2-RELAY

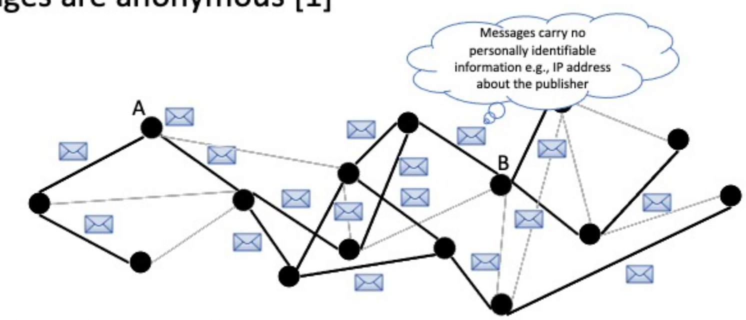
- Publisher-Subscriber Model
- Gossip-based Routing (extension of libp2p GossipSub-v1.1 [2])



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]



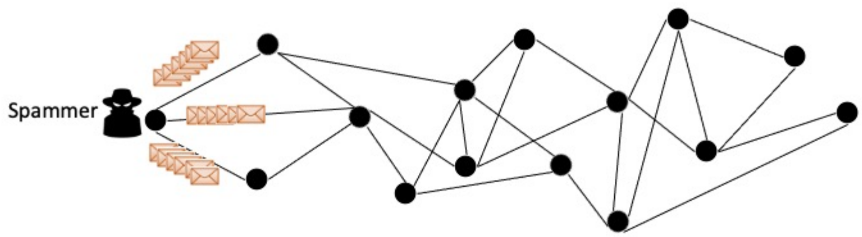
[1] https://rfc.vac.dev/spec/11/#security-analysis

7

Waku-relay is also an anonymous transport where there is no personally identifiable information like IP address or Peer ID 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

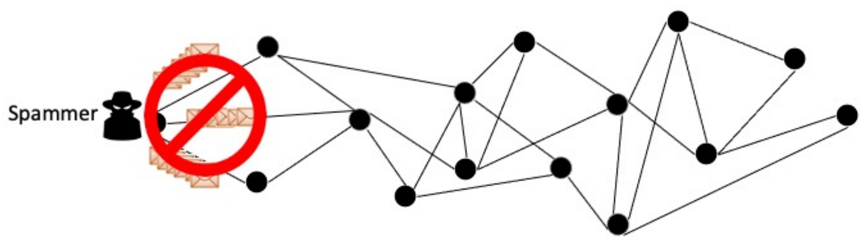


8

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

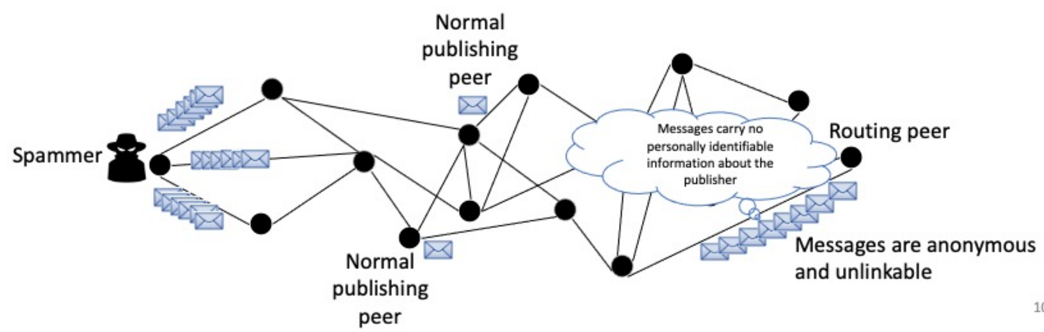
- We define spammers as entities that publish a large number of messages in a short amount of time, and cause denial-of-service
- Spam Protection = Controlled Messaging Rate



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**

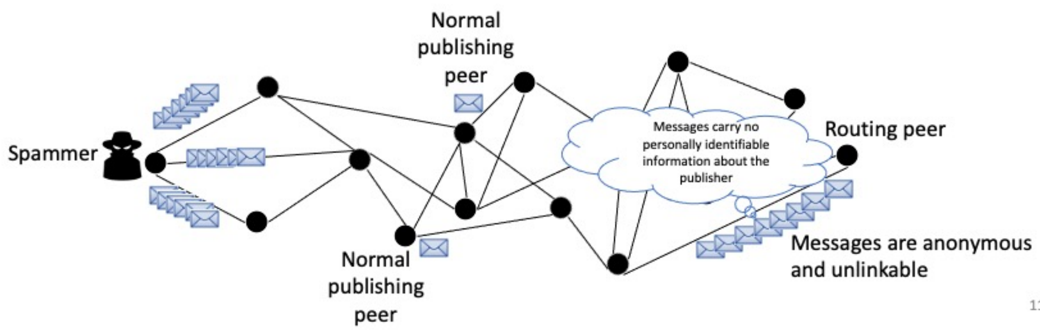


10

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



11

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.

12

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]

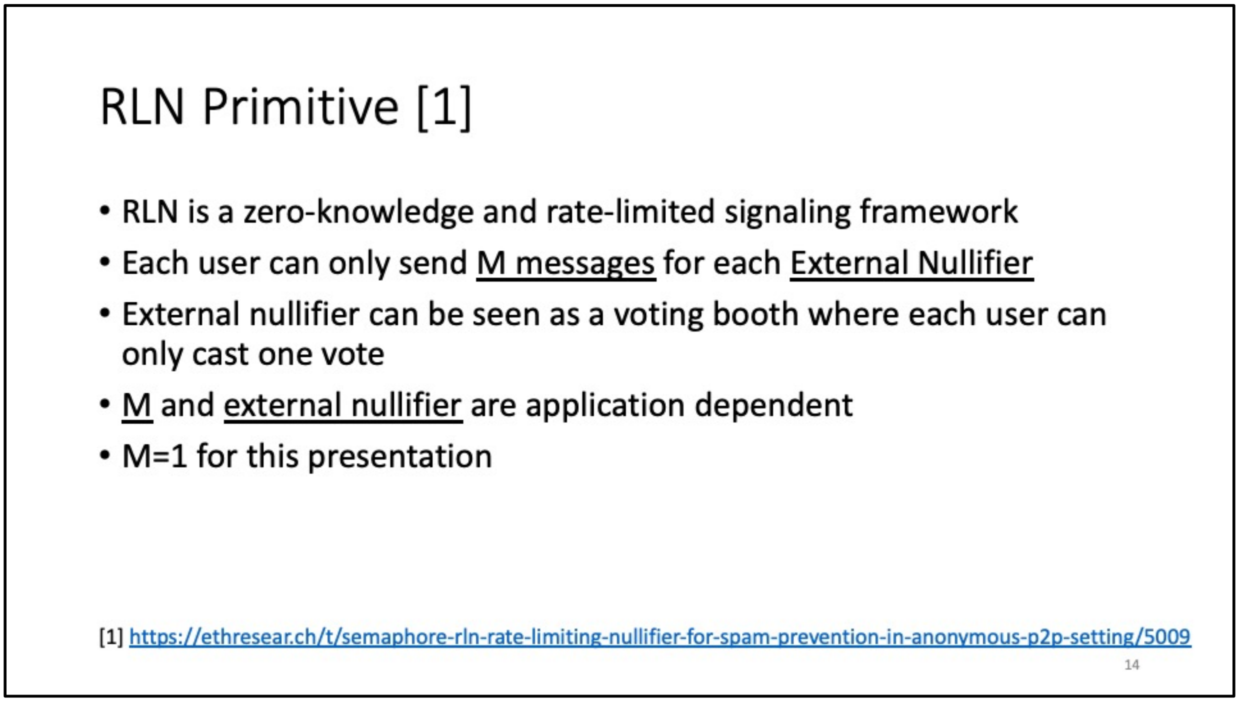
WAKU2-RLN-RELAY = WAKU2-RELAY + Rate Limiting Nullifiers (RLN)

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

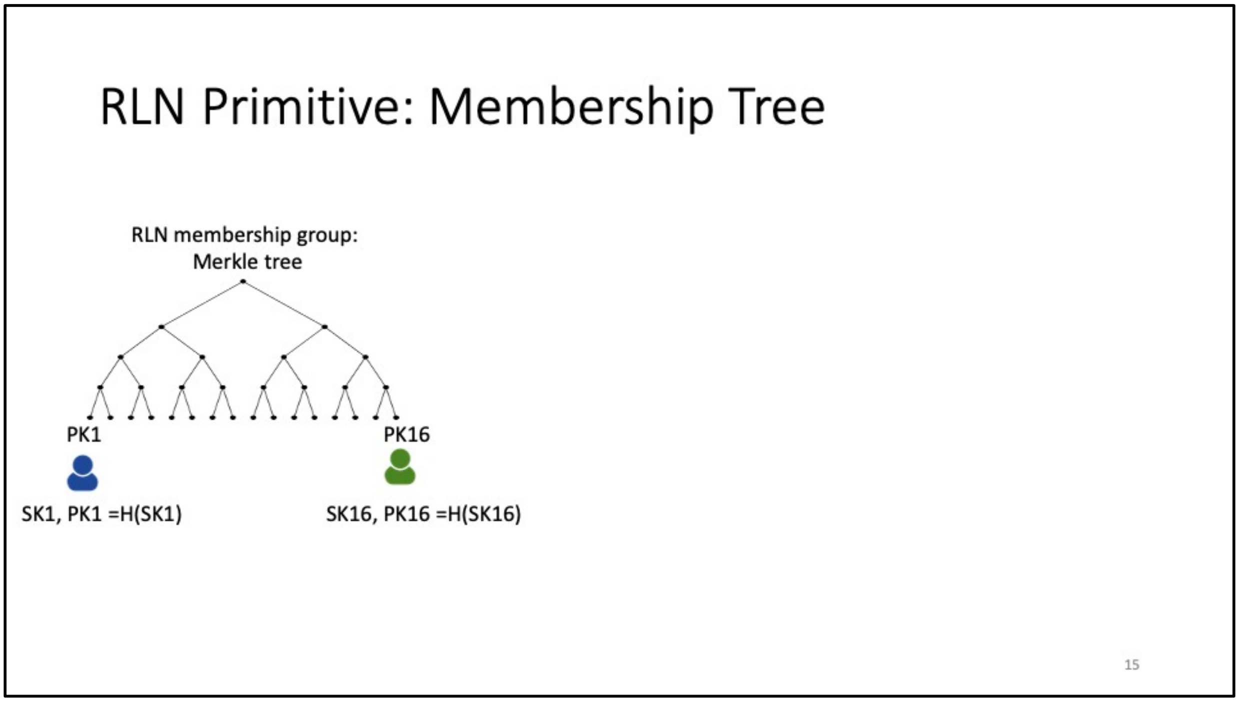
[1] https://rfc.vac.dev/spec/17/

13

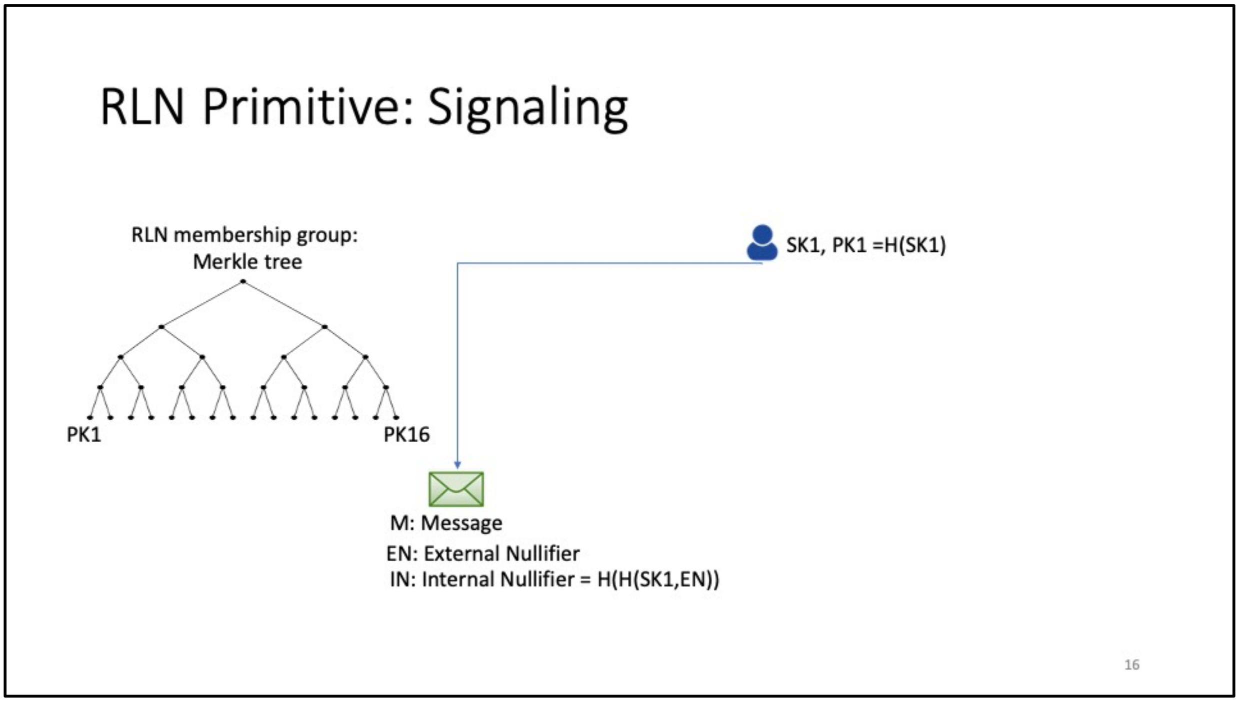
The good news is that in WAKU-RLN-RELAY we cope with the aforementioned issues
We take the waku-relay protocol as an anonymous transport protocol and combine it with the rate limiting nullifier construct to control the messaging rate
The end result has a p2p structure, with no central entity involved
it **allows global identificatio**n and removal of spammers
it is **privacy-preserving, because user anonymity is respected** 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.



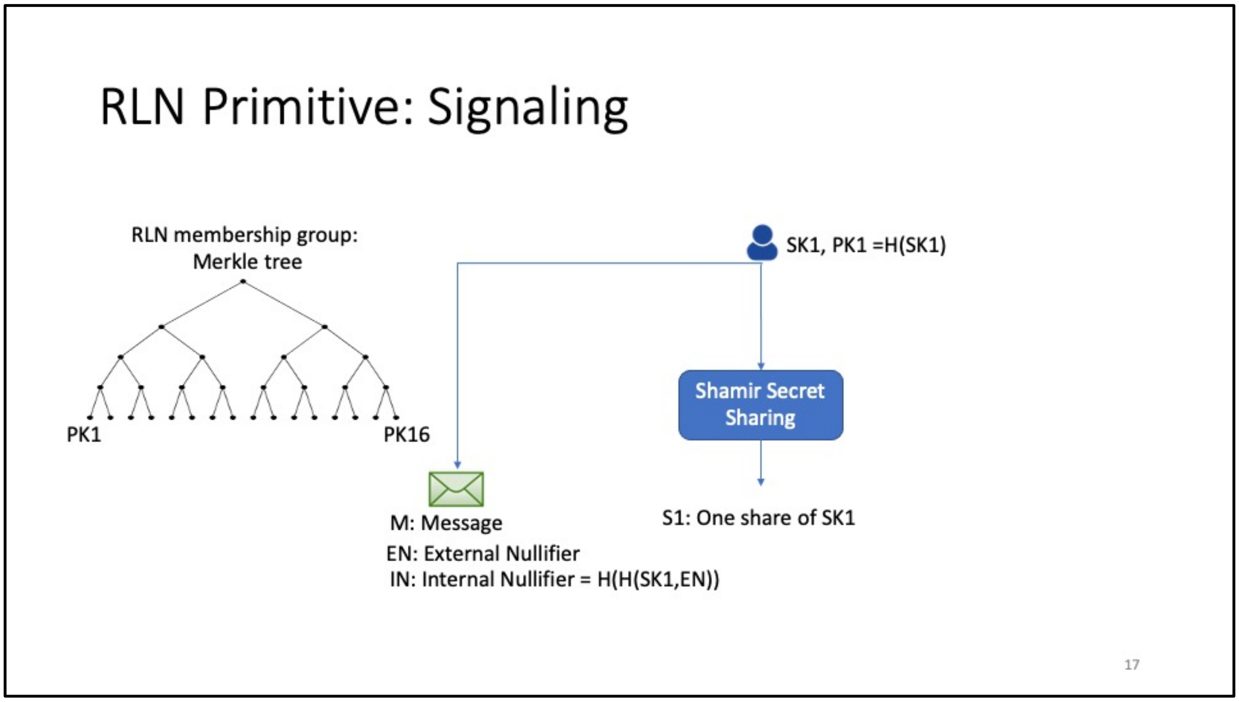
- Lets begin with the RLN construct
- it is a zero-knowledge and rate-limited signaling framework
- It allows a set of users to broadcast arbitrary signals (where signal is any value like a string, vote, etc.) while proving they are among a group of authorized users without disclosing their identities
- 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



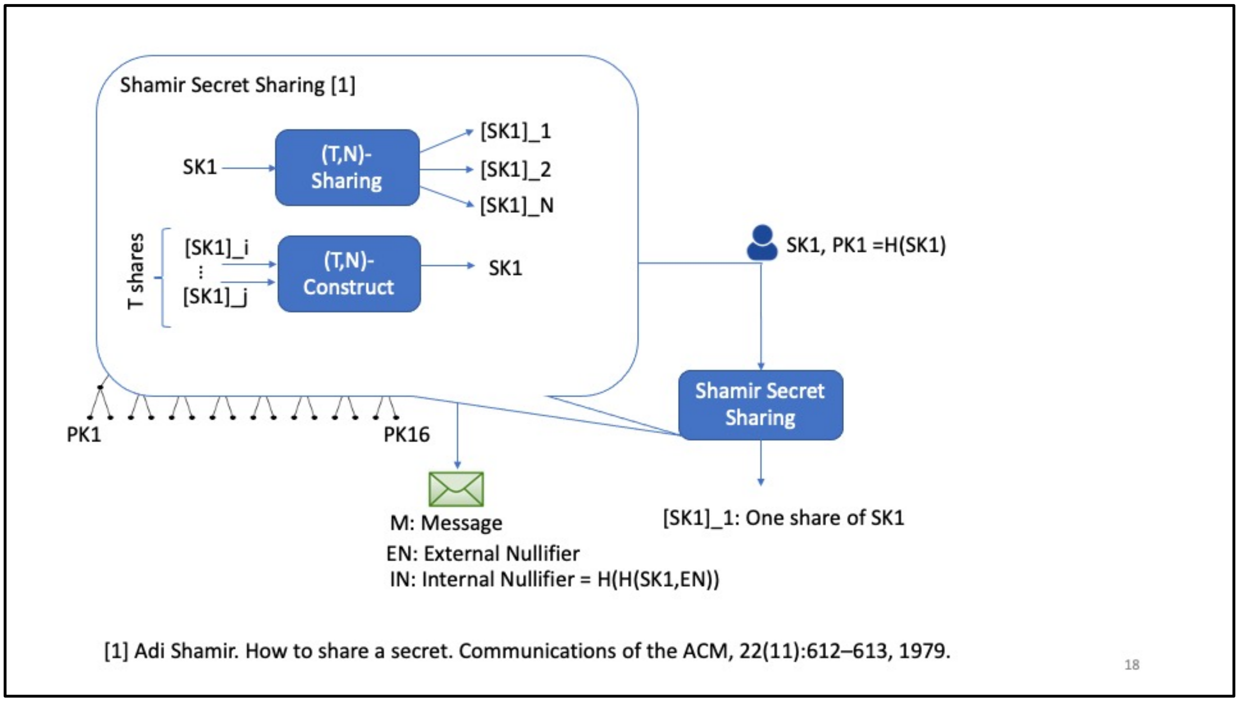
- RLN primitive consists of a merkle tree that represents a group of authorized users,
- each user has a pk registered in this tree,
- the corresponding SK is only known to the user



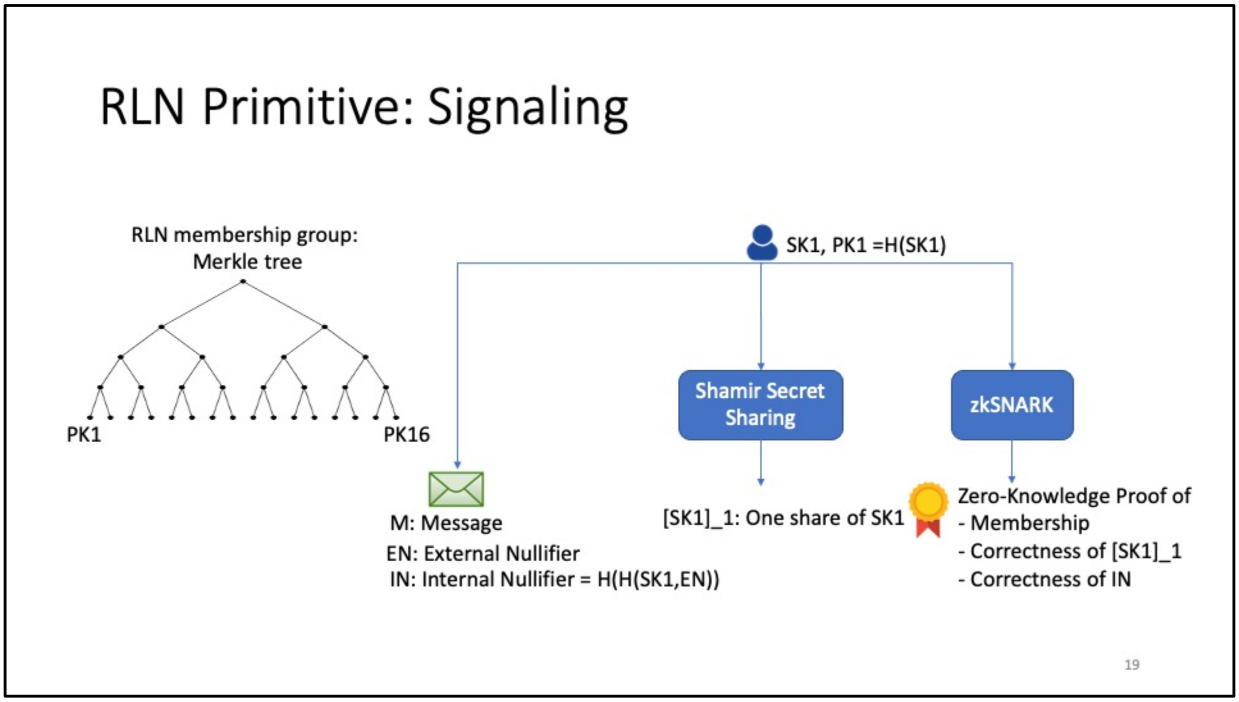
- For the signaling/publishing:
- The user specifies an external nullifier
- as well as an internal nullifier which is derived from the SK and the external nullifier (as you can see in the slide)



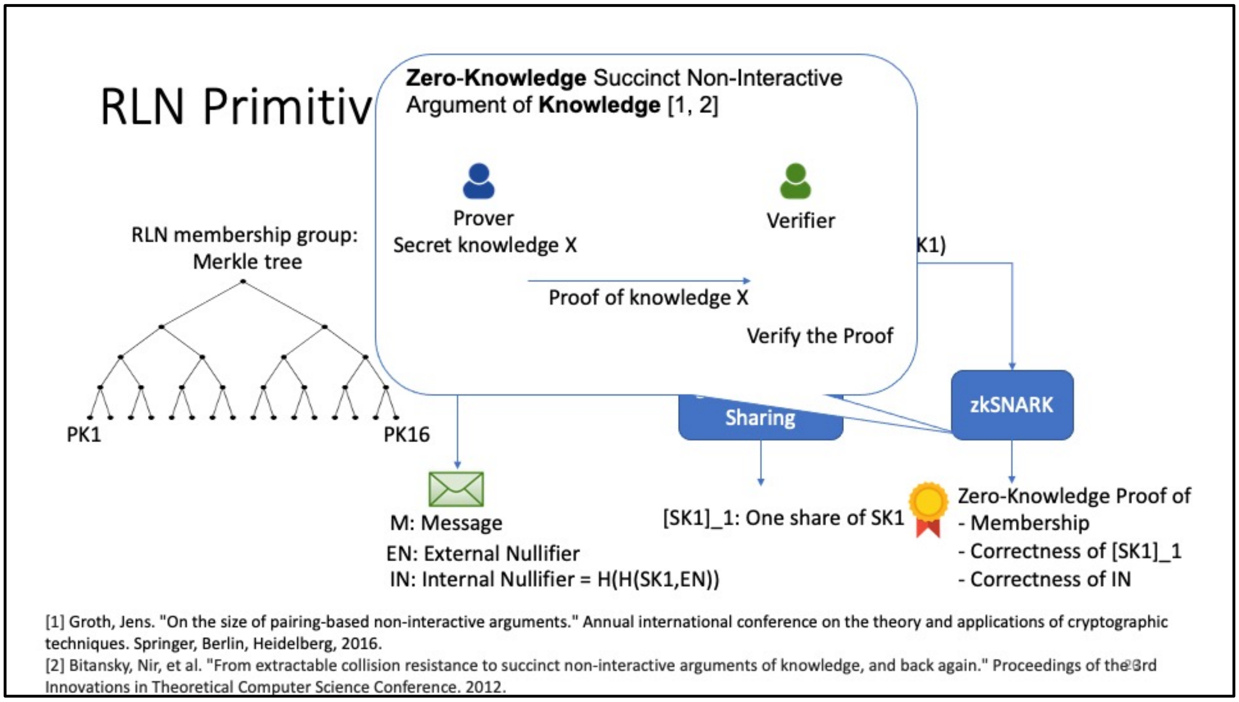
- The user also discloses a share of its secret key using shamir secret sharing scheme
- This share will be used to remove the user from the group in case of double signaling



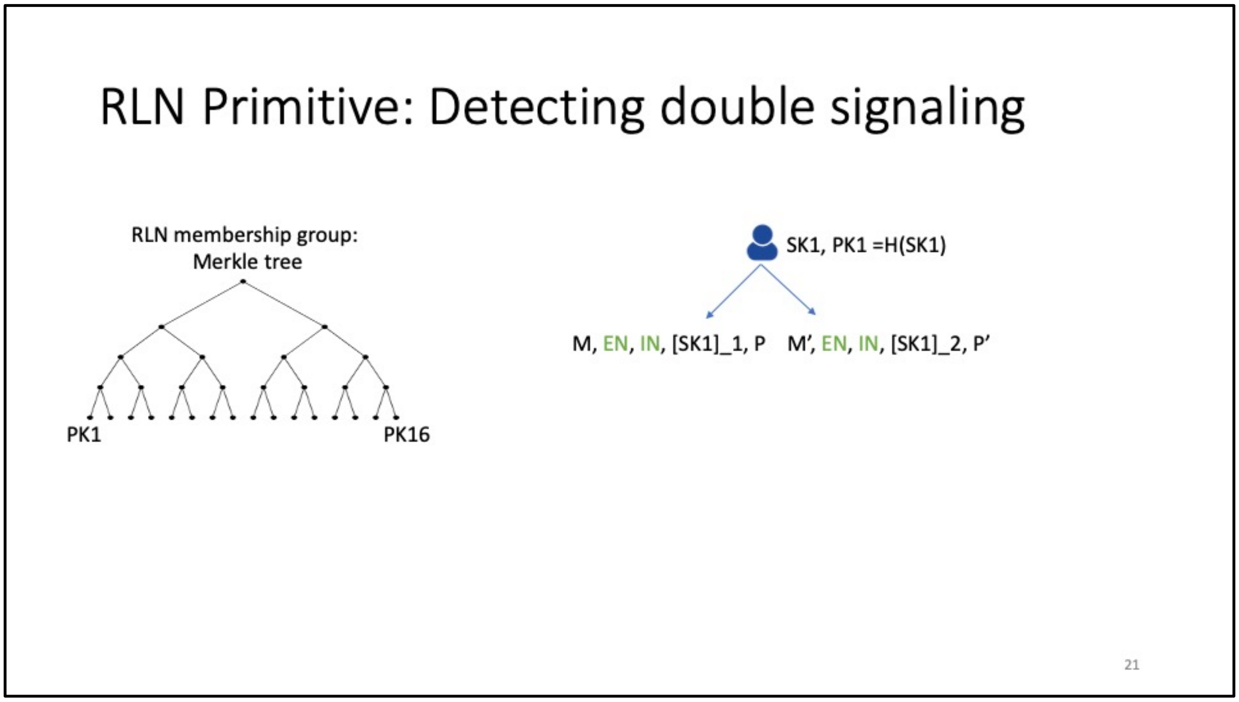
For those who dont know about Shamir secret sharing, the high level idea is that it is a technique that allows to split a secret data sk into N pieces
It is possible to construct the sk back by having a subset of T shares
For this presentation T is 2



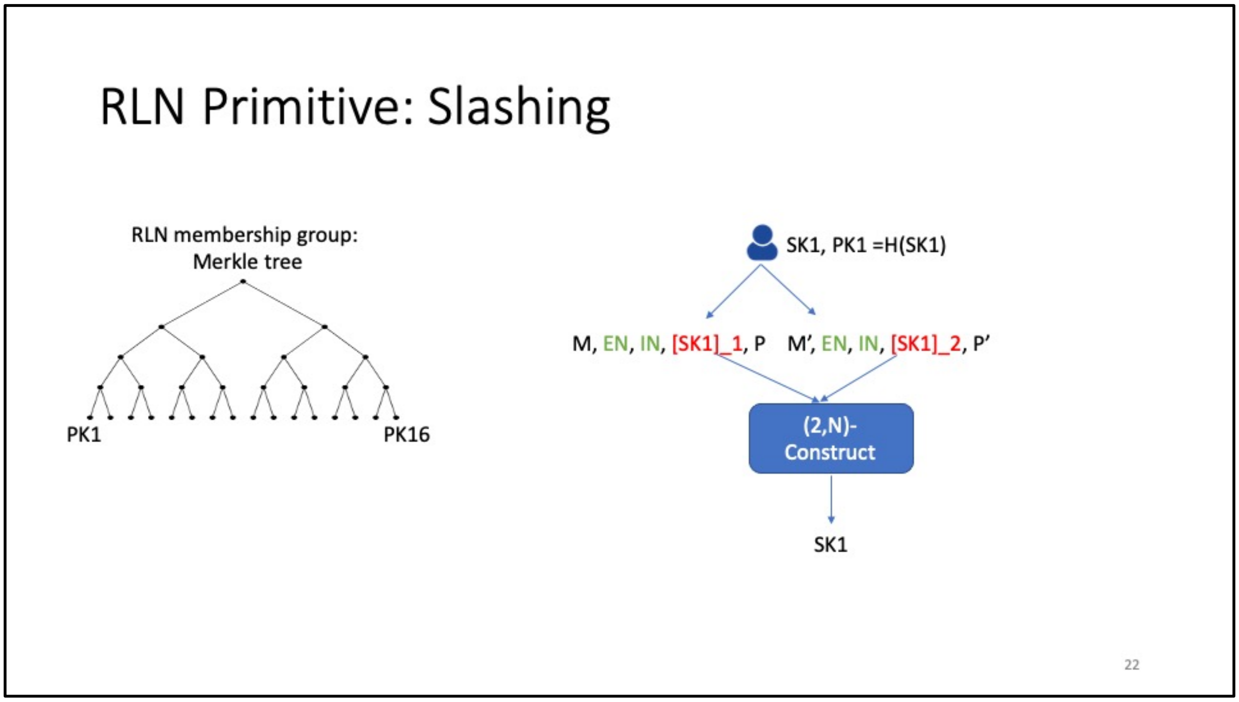
- Finally, the user proves in zero knowledge manner that:
- It is part of the group
- And that has computed the secret share and the Internal nullifier correctly



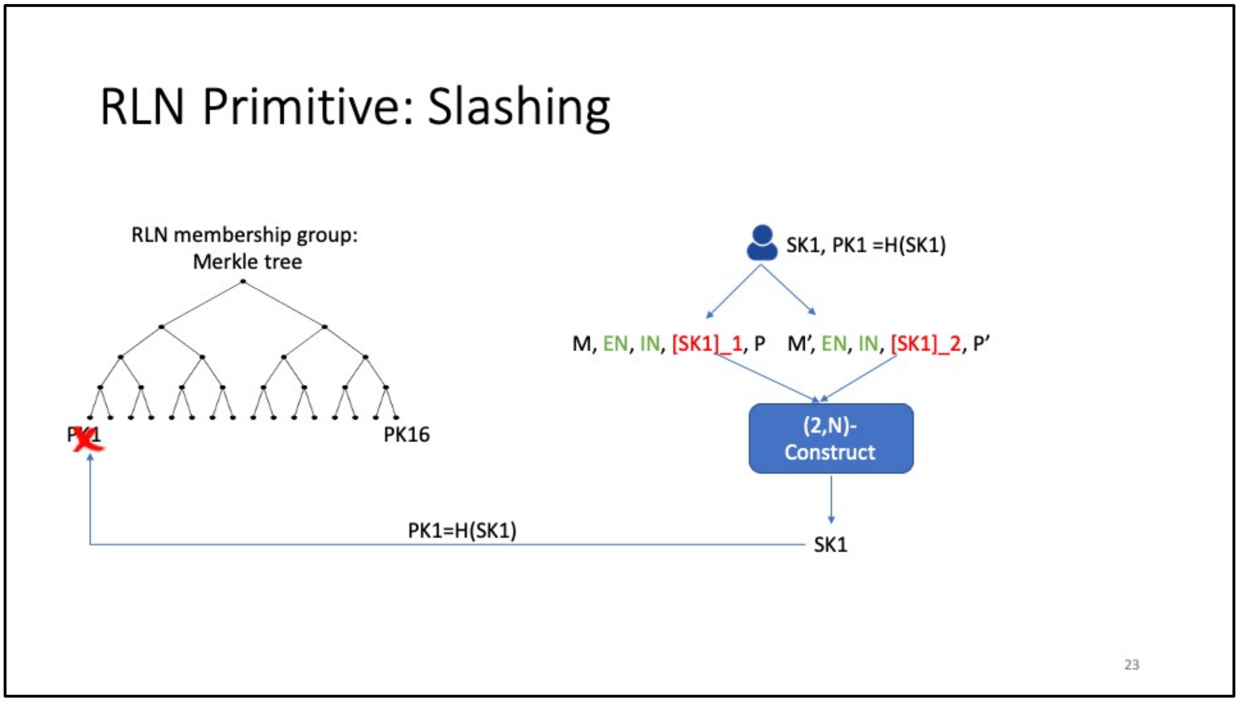
- For those not familiar with zk-SNARK, it stands for "Zero-Knowledge Succinct Non-Interactive Argument of Knowledge."
- And at a high level is is a cryptographic proof that allows one party to prove the possession of certain information to a verifier without revealing that information.



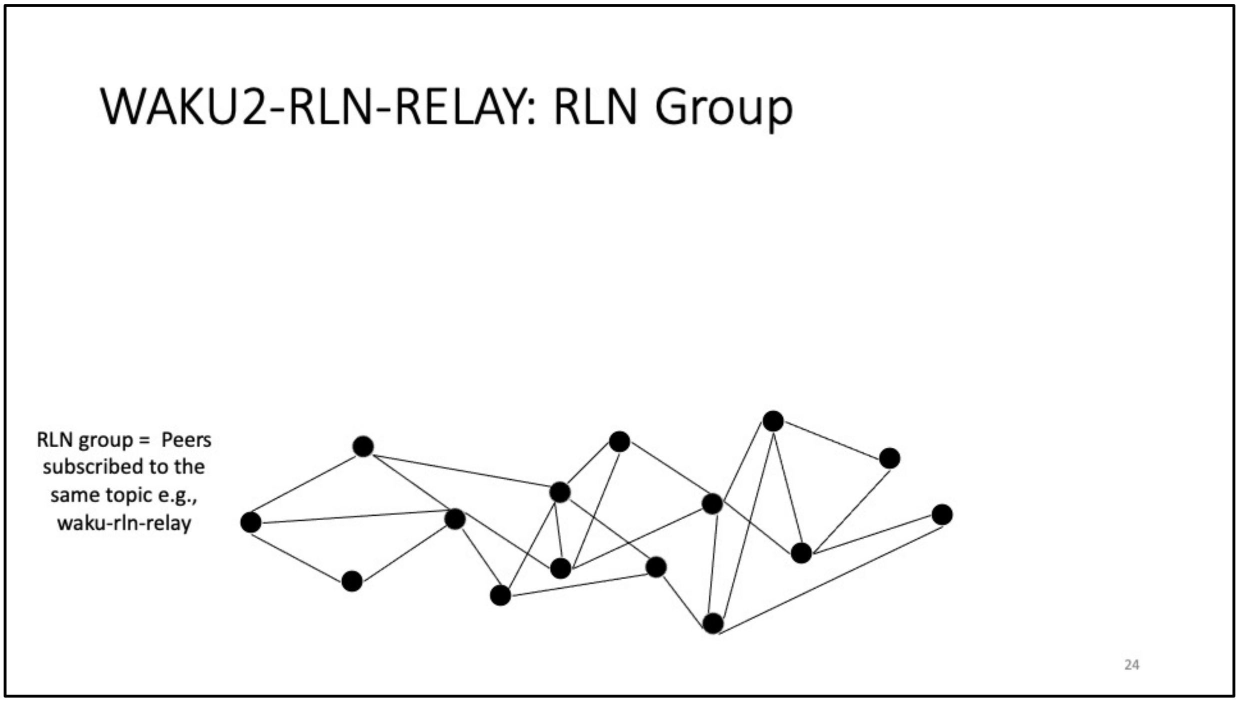
- In RLN construct, Users cannot violate the messaging rate, why? Because If a user publishes more than 1 message for the same external nullifier, it will end up having two messages with the same external and internal nullifiers (remember those are values deterministicly computed from SK and the external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



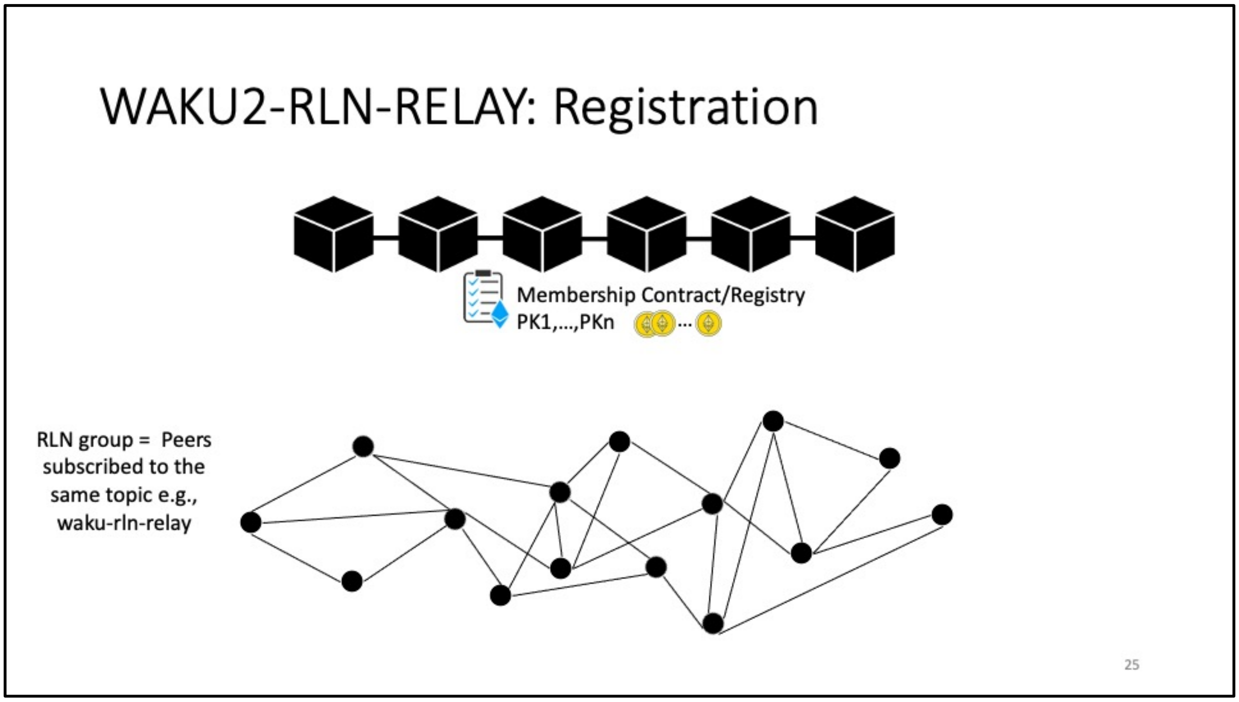
- Moreover, by sending two messages(violating the messaging rate), the user will disclose 2 shares of its SK (one per each message)
- using which the corresponding SK can be reconstructed



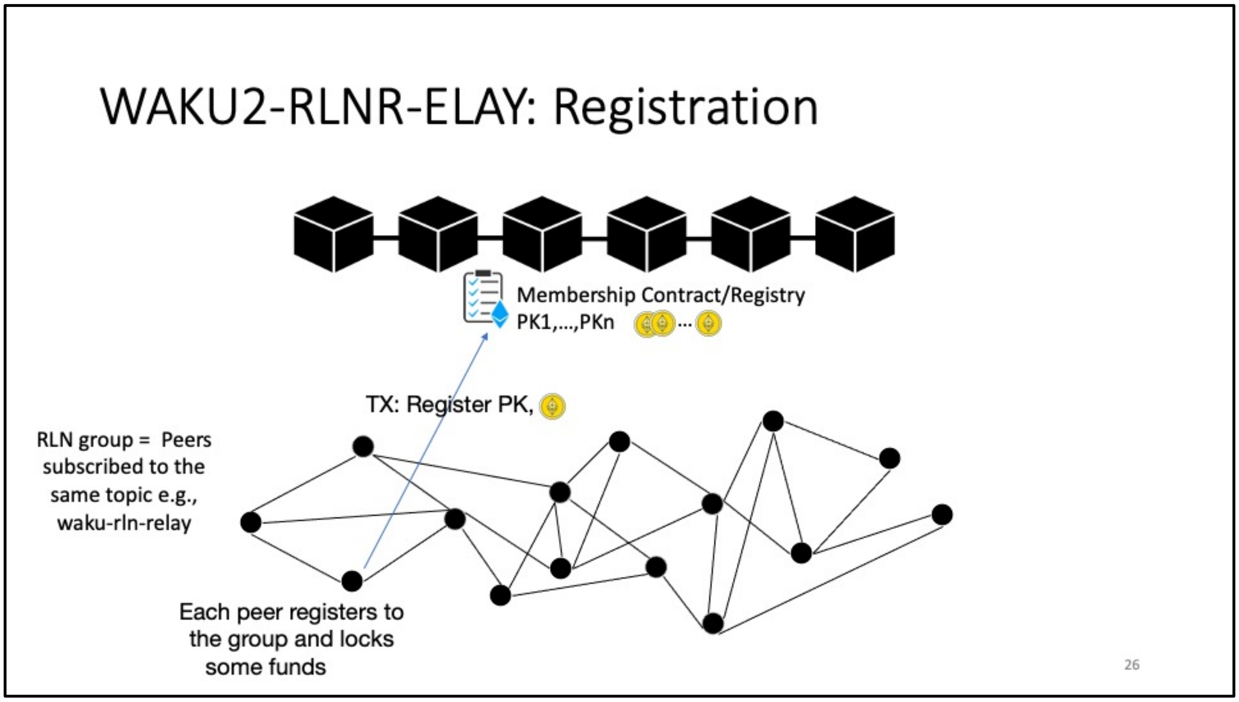
- And the user gets removed from the tree. Thus, it can no longer use that sk for messaging.



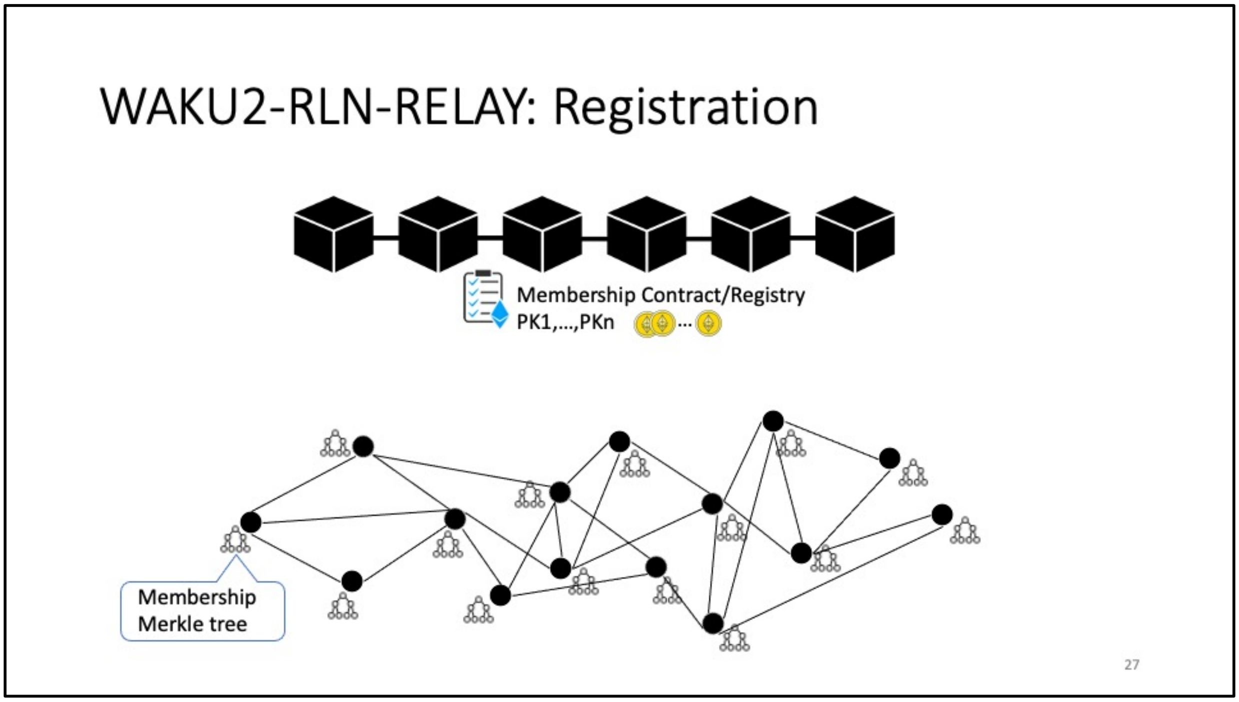
Now lets see the end to end integration of RLN into Waku-rln-relay to achieve spam protection
Here, the rln group consists of the peers that belong to the same GossipSub layer (subscribed to the same topic)



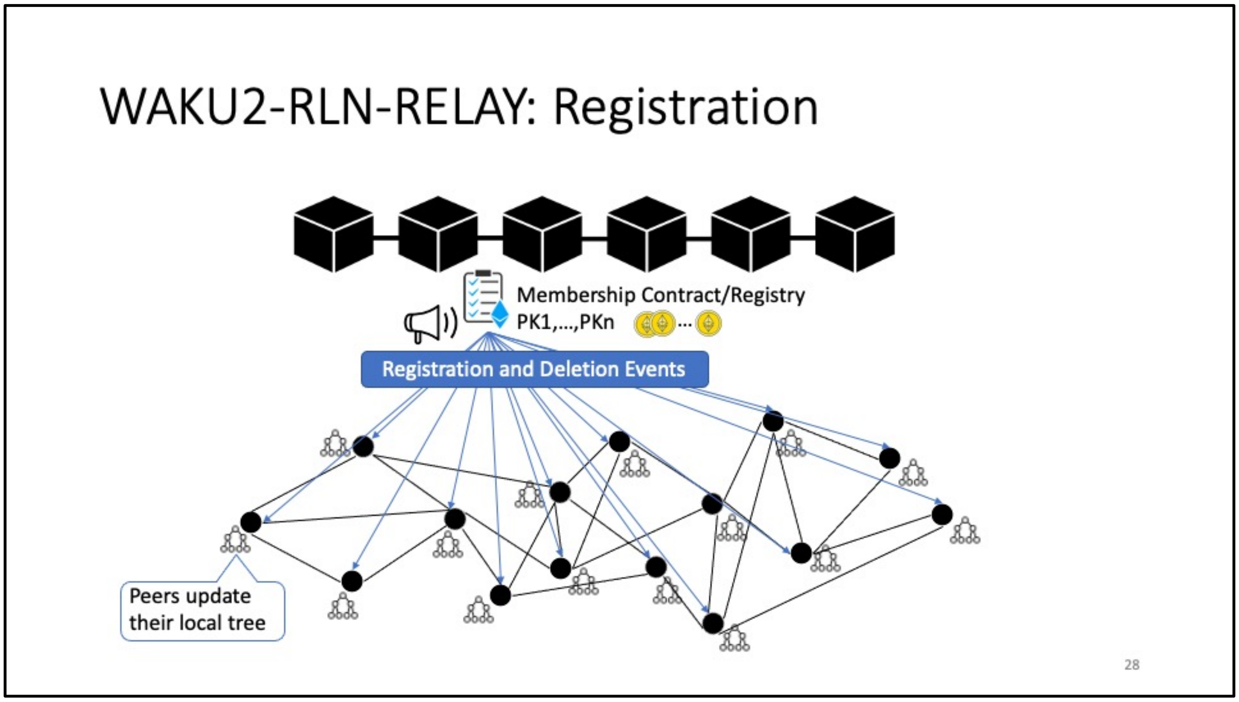
Each peer has a rln pk, and the list of pks is stored in a contract deployed on the Eth blockchain.



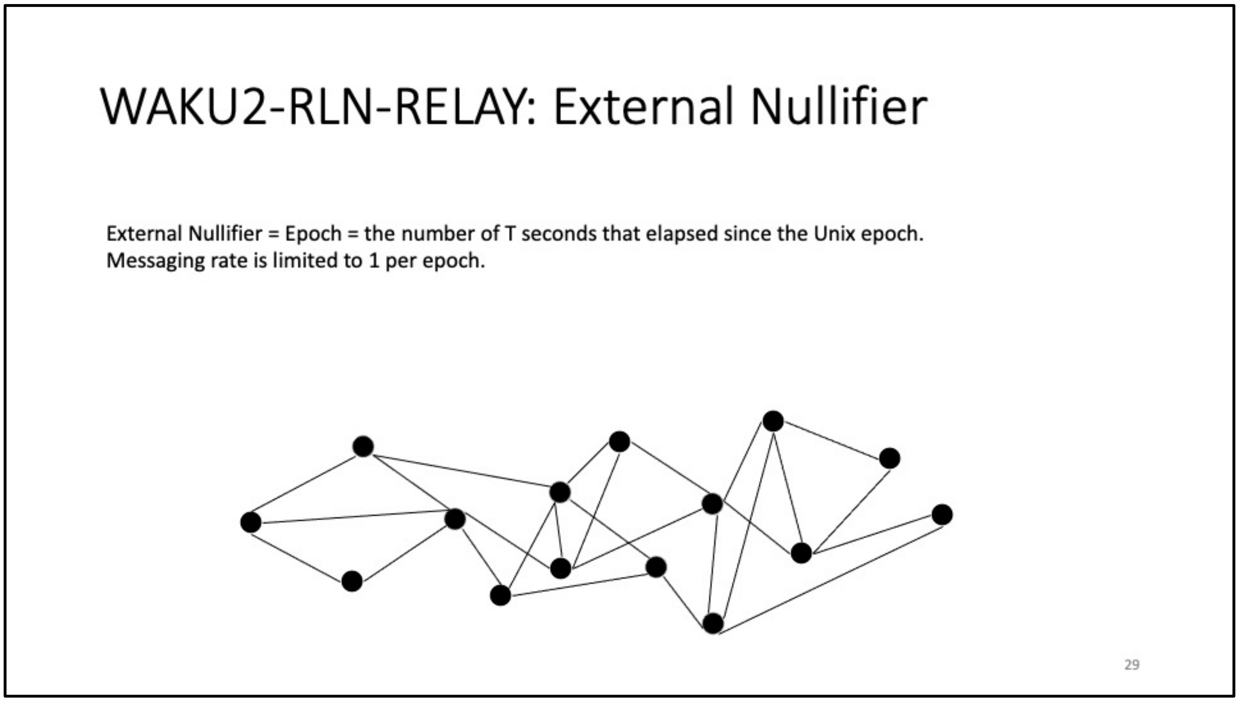
A peer willing to publish a message should register by sending a tx to the contract that contains its rln pk and some amount of Ether. This amount is deposited on the contract to prevent spam activity. Registration is a one time action.



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.

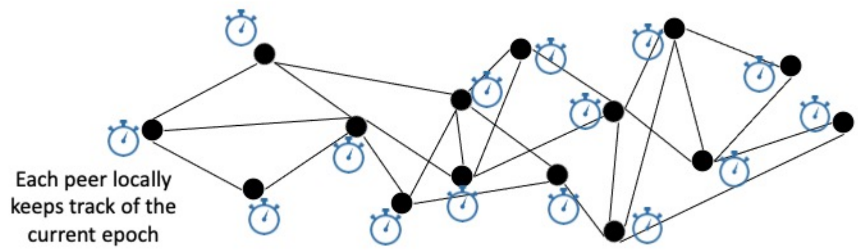


For the external nullifier we denote it by Epoch which is the number of T seconds (where T is a system design parameter) that elapsed since the Unix epoch.

Peers are allowed to publish one message per epoch without being slashed

WAKU2-RLN-RELAY: External Nullifier

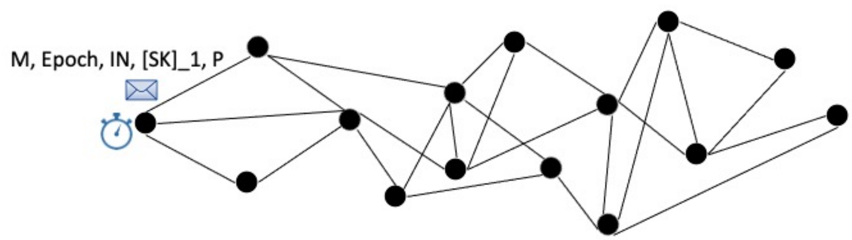
External Nullifier = Epoch = the number of T seconds that elapsed since the Unix epoch.
Messaging rate is limited to 1 per epoch.



30

Each peer locally keeps track of the current epoch.

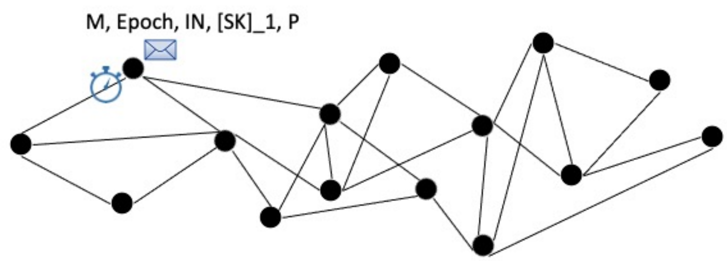
WAKU2-RLN-RELAY: Publishing



31

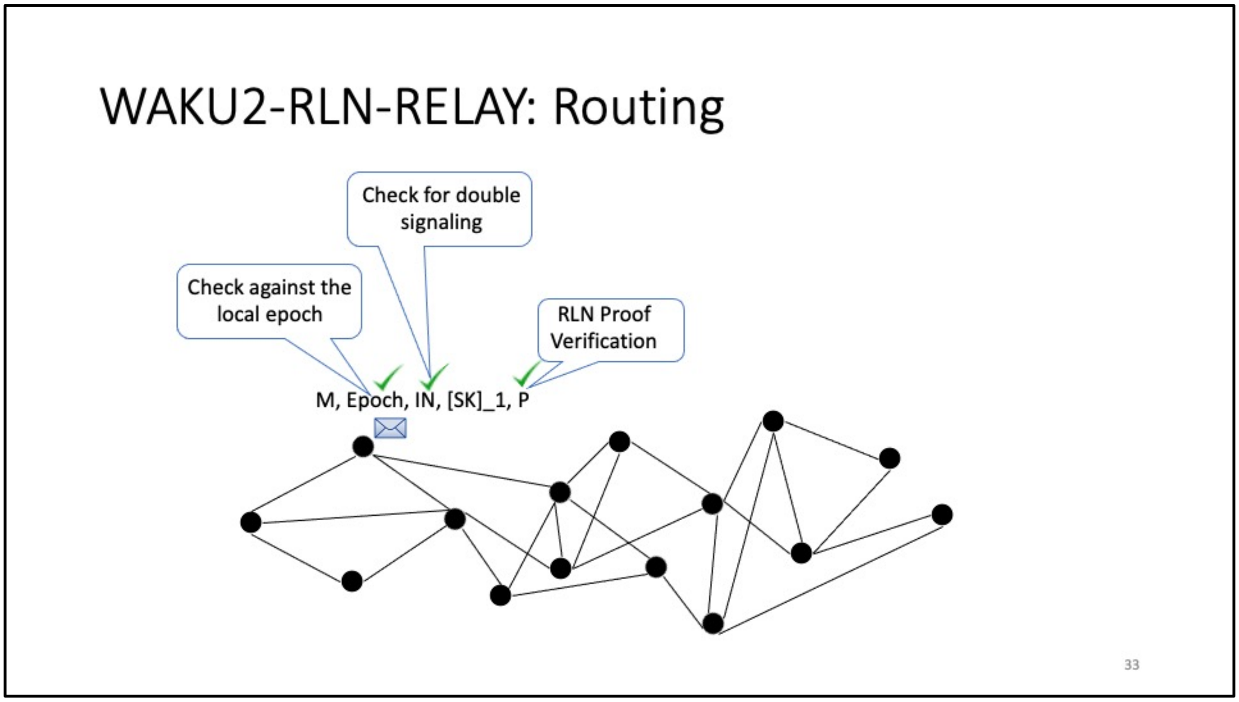
Message publishing in the network is the same as the RLN framework
The message owner, attaches the nullifiers, together with a share of its secret key, and the zero knowledge proof part to the message

WAKU2-RLN-RELAY: Routing



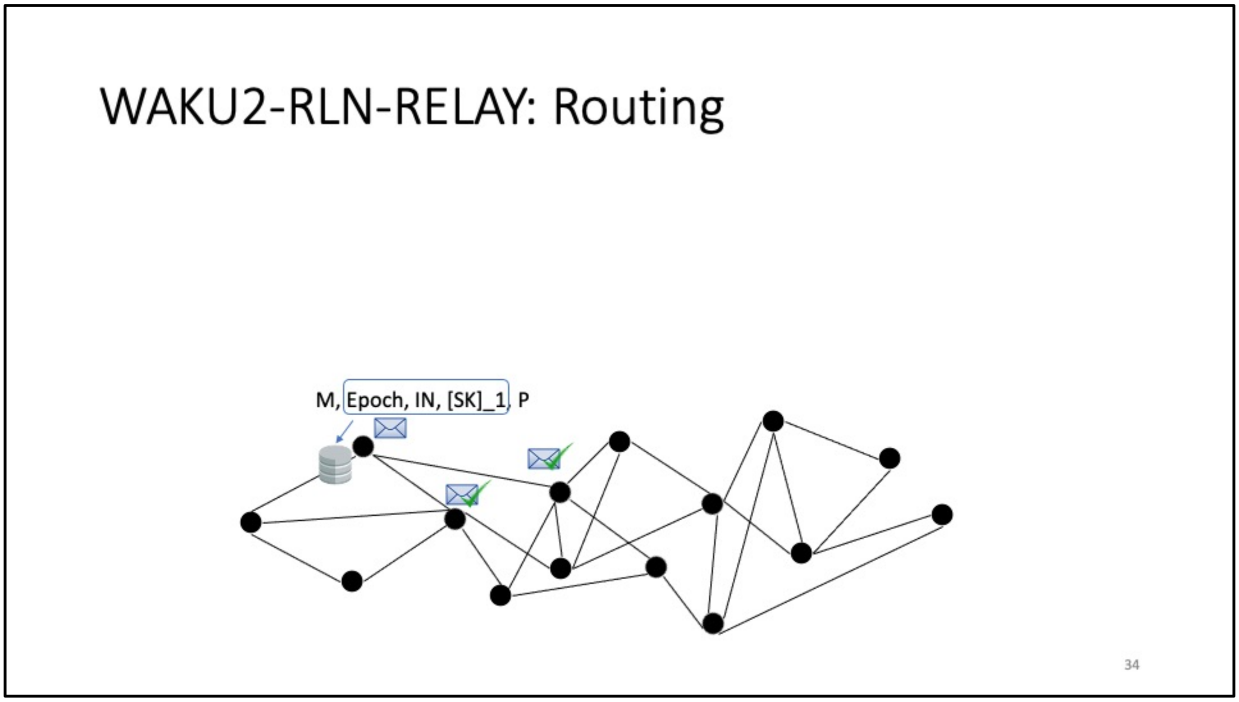
32

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

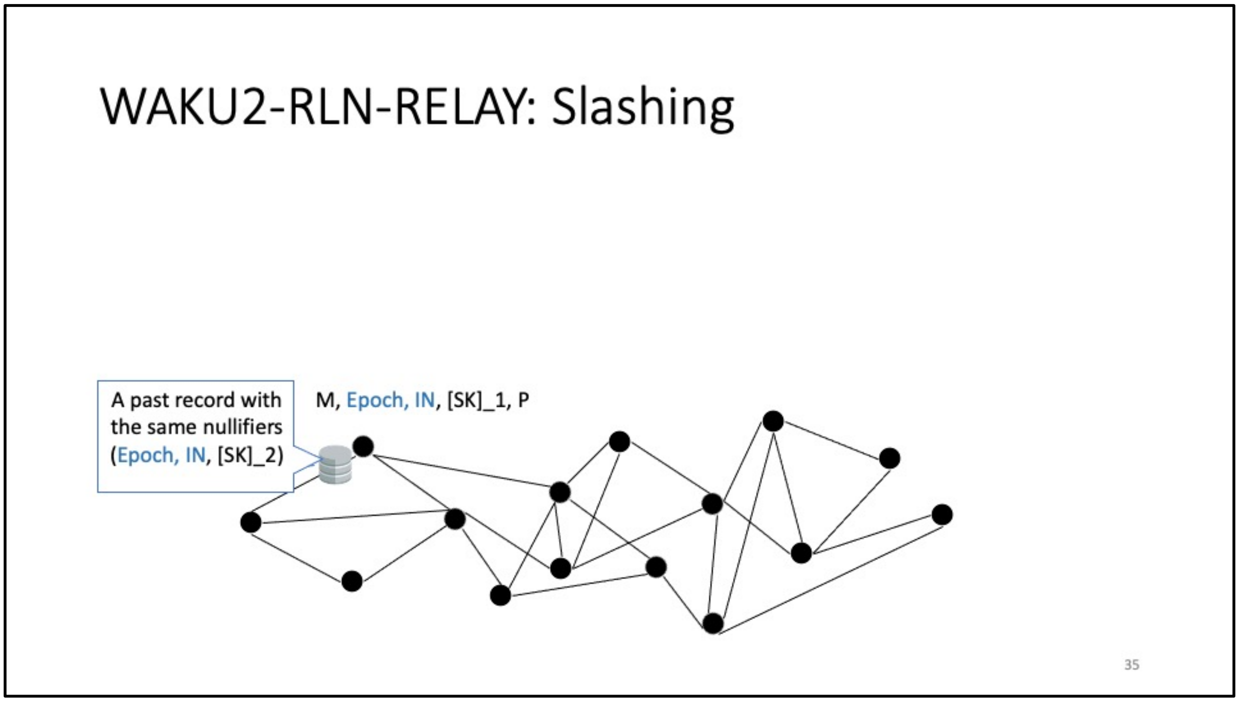


meaning that
It verifies the proof
Also validates the Epoch of the incoming message against its local Epoch to see if there is a huge gap or not

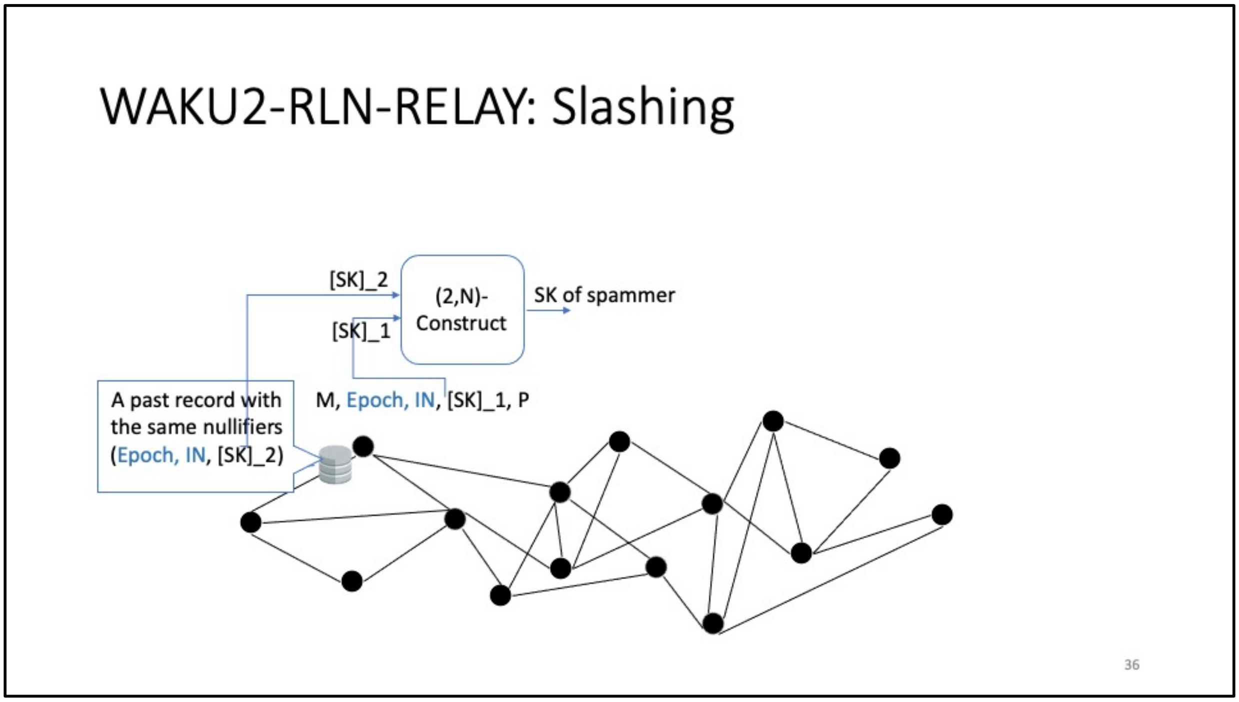
checks the nullifiers to see if double signaling has happened
If all the checks pass, relays the message



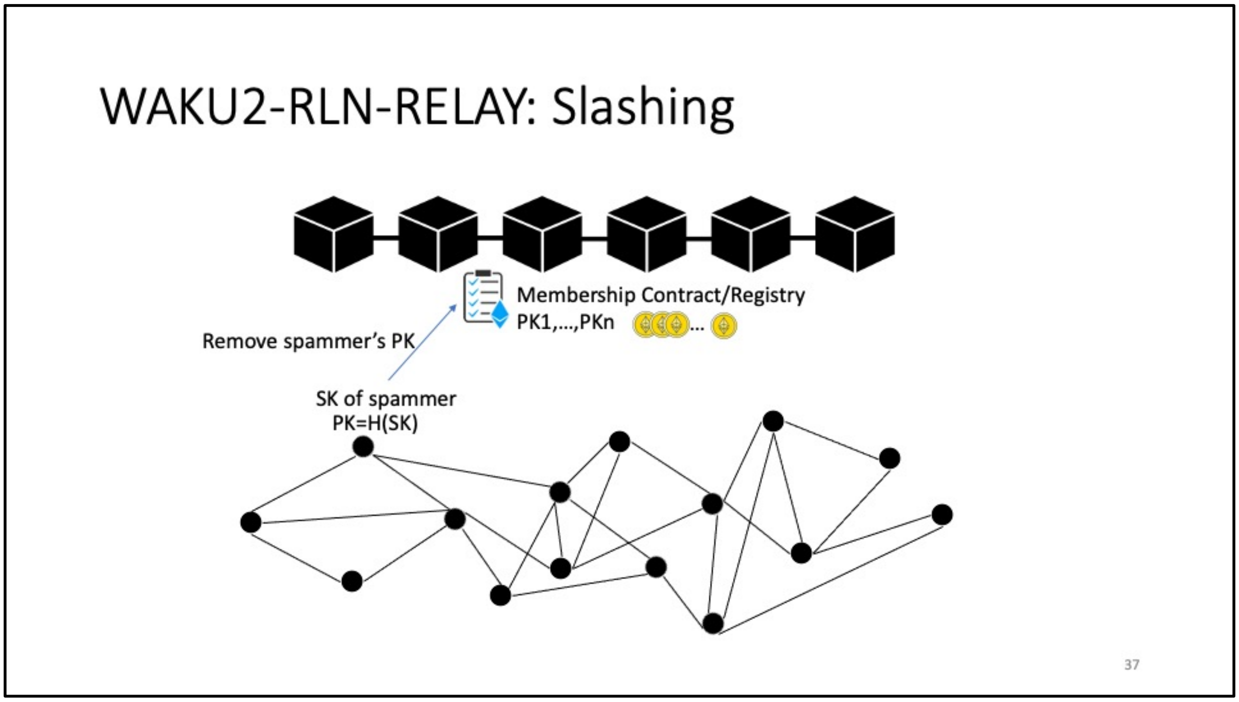
Persists a record of the nullifiers and the secret share of the messages, it is needed to catch double signaling for the future incoming messages



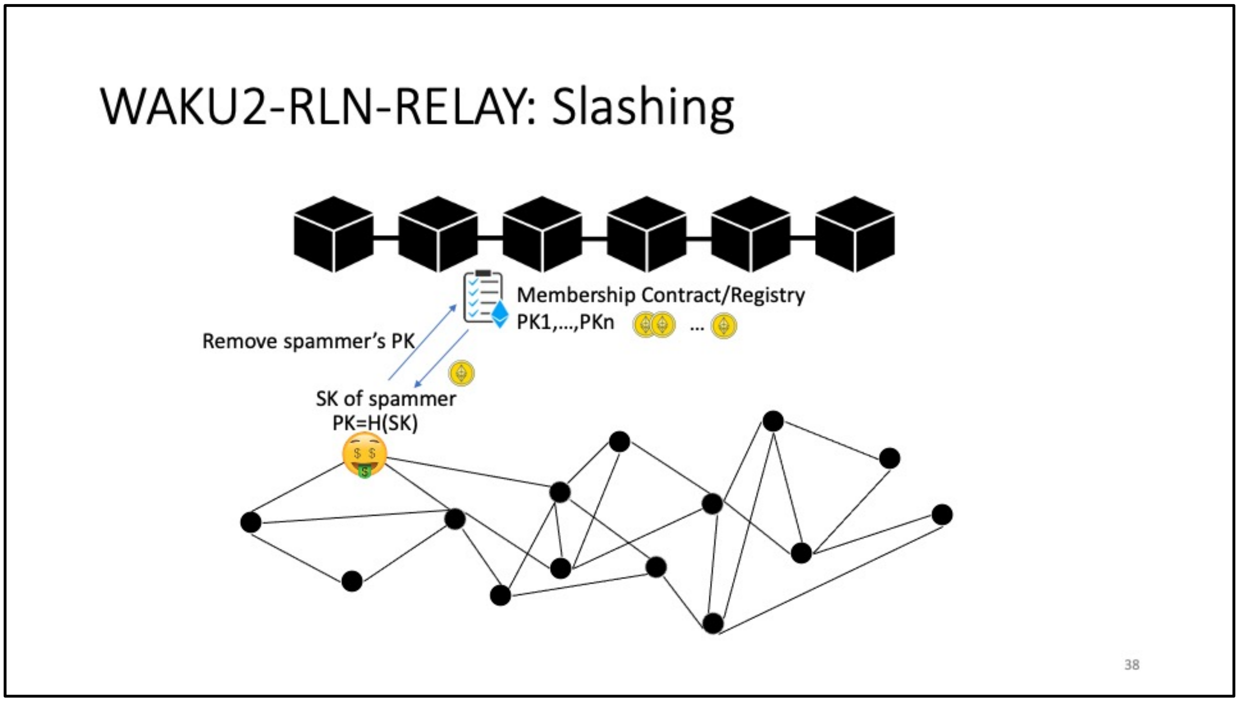
But what if the routing peer finds out that the messaging rate is violated, i.e., there has been an old message whose nullifiers match the new message



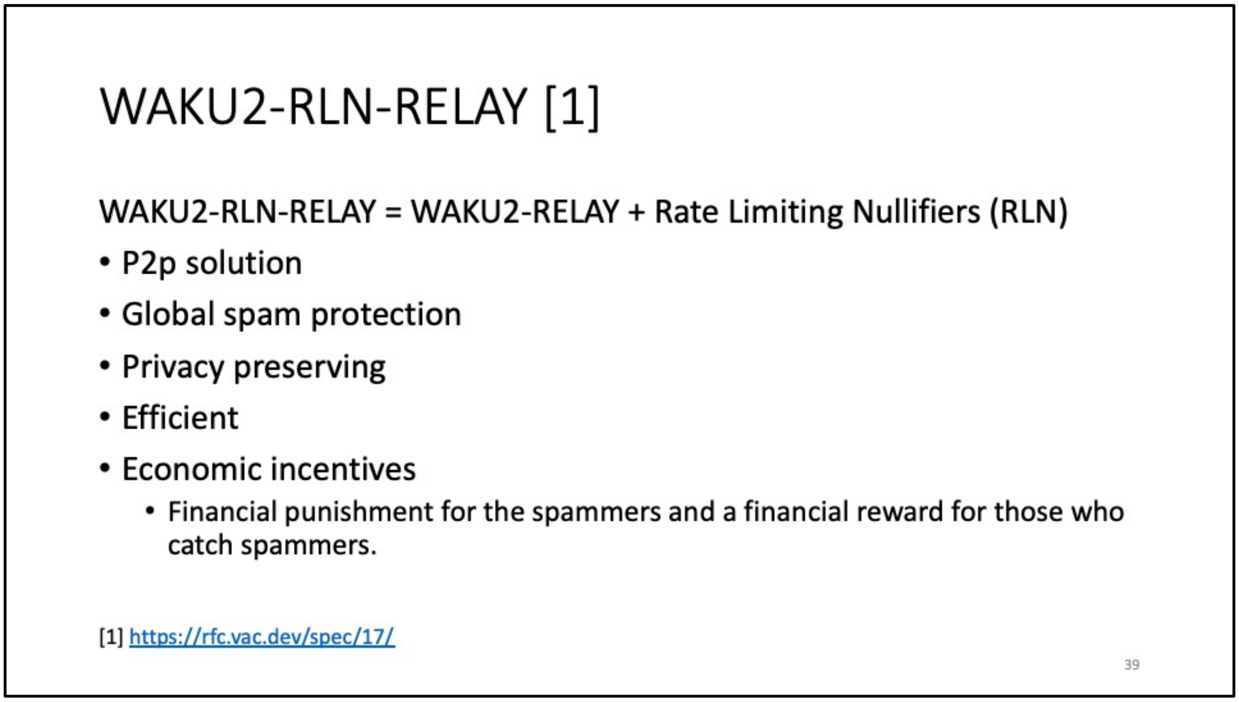
In that case, it reconstructs the sk of the spammer



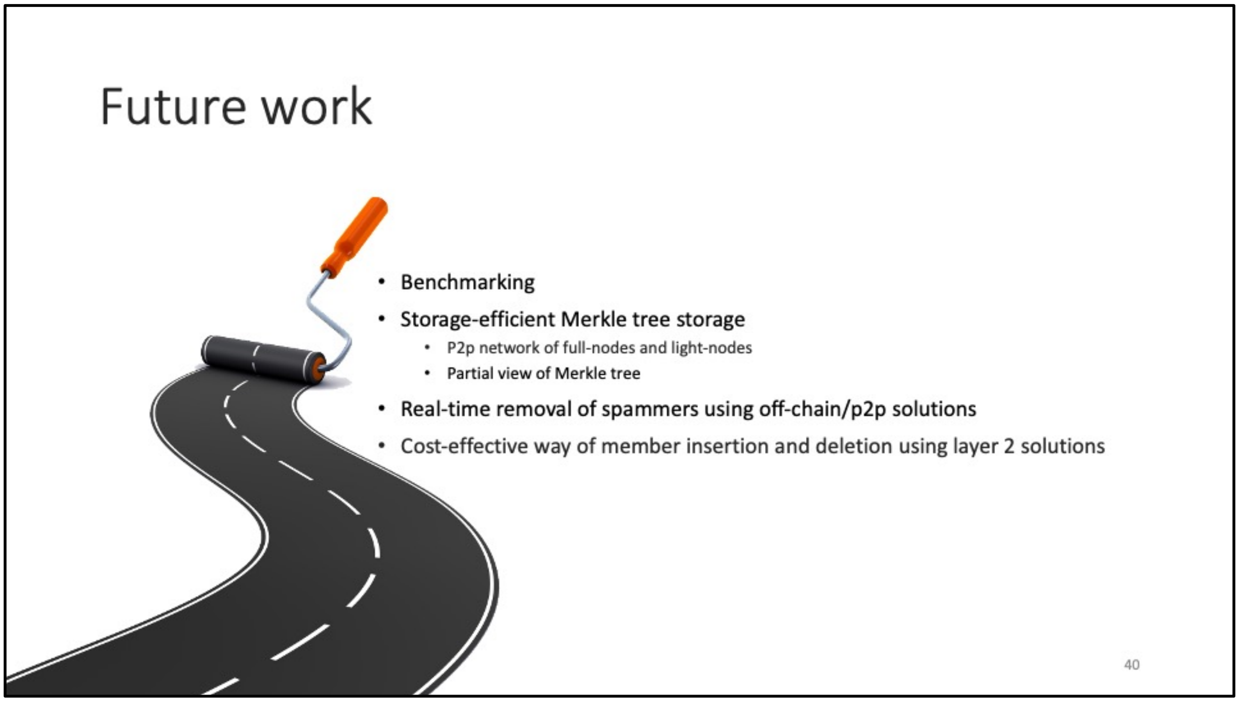
Sends a transaction to the contract and removes the spammer pk from the group



and gets its reward by withdrawing the spammer's deposit.



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-prevention in a p2p messaging system



In the end, I would like to shed light on our future plan
Benchmarking is the first on the roadmap
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

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

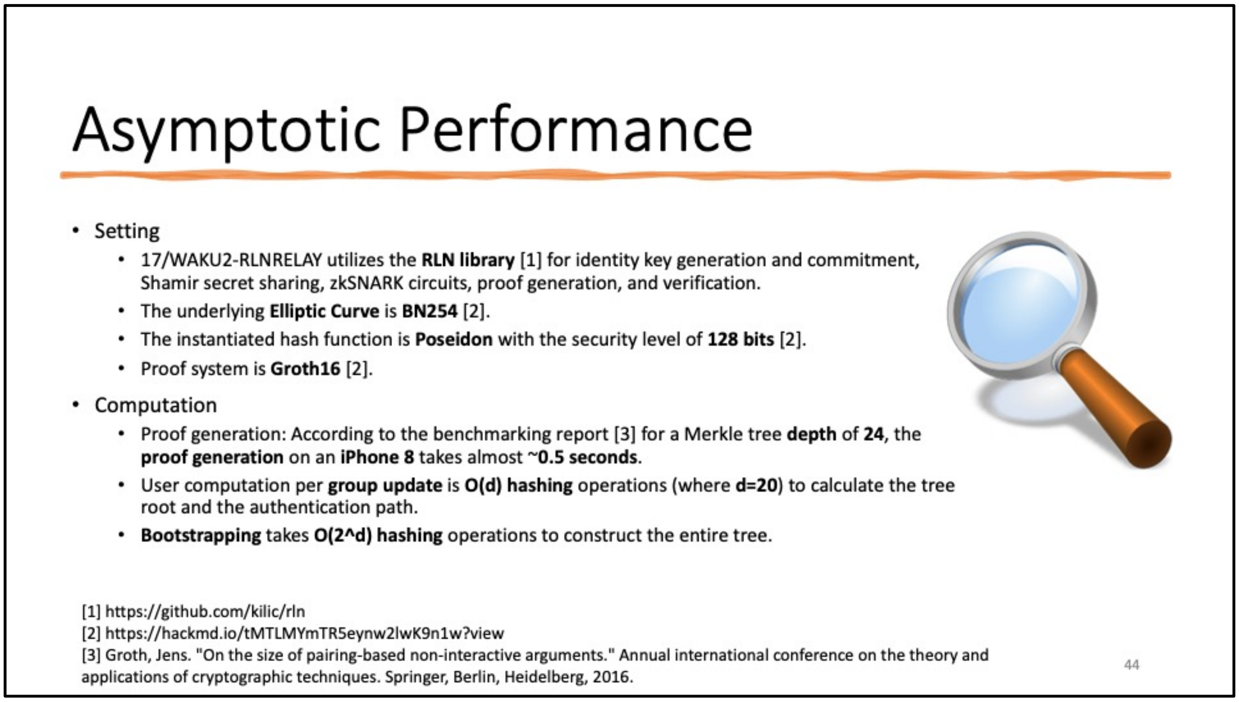
41

References for those interested to read further

Thank you

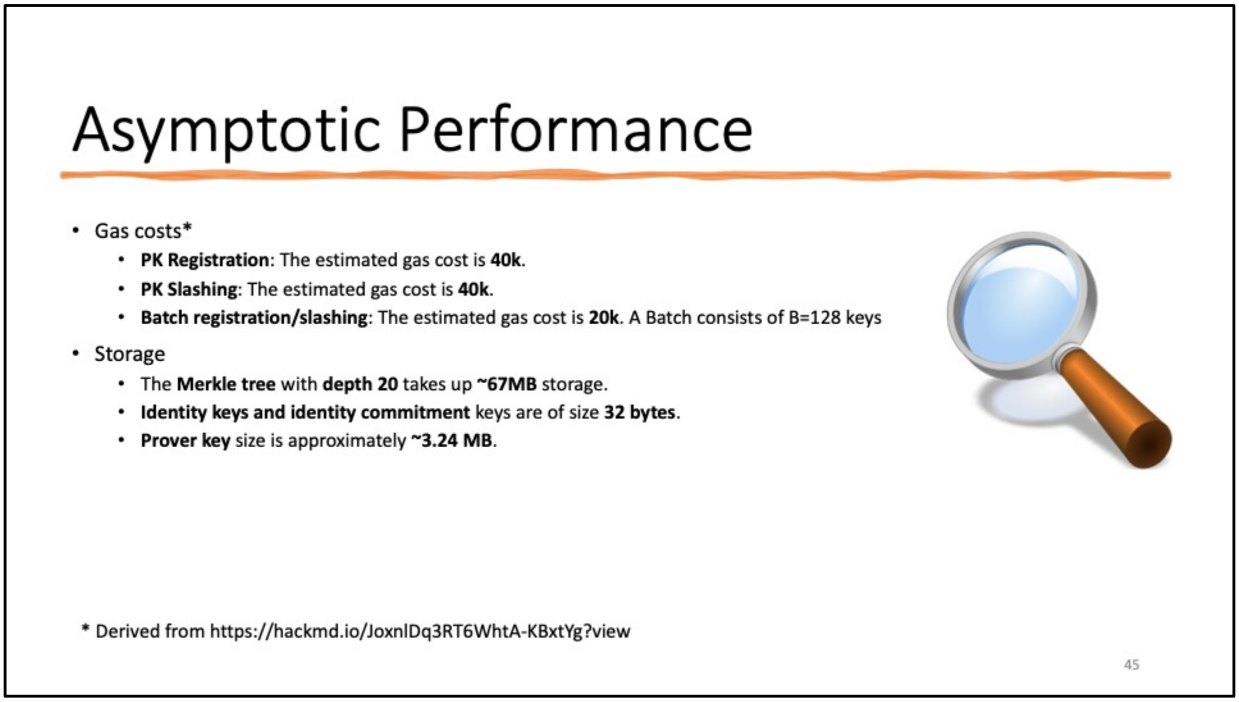
42

43



The proof generation time is sufficiently fast for many messaging applications, but may not be low enough for e.g. real-time communications.

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



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